Virtuozzo Hybrid Infrastructure 4.7

Administrator Command Line Guide

12/27/2021

Table of contents

Introduction

This guide describes the syntax and parameters of the `vinfra` command-line tool that can be used to manage Virtuozzo Hybrid Infrastructure from console and automate such management tasks.

Note
While the following chapters provide information on specific operations that you can perform with `vinfra`, you can also run `vinfra help` to get a list of all supported commands and their descriptions. For help on a specific command, either run `vinfra help <command>` or `vinfra <command> --help`.

In addition, this guide describes how to use the command line to perform operations unsupported by `vinfra` as of now.

Note that the following operations should not be done from the command line:

- Setting custom paths for Virtuozzo Hybrid Infrastructure services, in particular:
  - Creating S3 clusters only in `/mnt/vstorage/vols/s3`
  - Creating iSCSI targets only in `/mnt/vstorage/vols/iscsi`
- Mounting clusters or change cluster mount options
- Configuring firewall with `firewall-cmd`
- Renaming network connections
- Managing metadata and storage services
- Managing partitions, LVMs, or software RAID
- Modifying files in `/mnt/vstorage/vols` and `/mnt/vstorage/webcp/backup` directories
- Setting encoding or replication of cluster root

Providing credentials

The `vinfra` CLI tool requires the following information:

- IP address or hostname of the management node (set to **backend-api.svc.vstoragedomain** by default)
- User name (**admin** by default)
- Password (created during installation of Virtuozzo Hybrid Infrastructure)
- Domain name to authenticate with (**Default** by default)
- Project ID to authenticate with (**admin** by default)

This information can be supplied by using the following command-line parameters with each command:

- `--vinfra-portal <portal>`
- `--vinfra-username <username>`
- `--vinfra-password <password>`
- `--vinfra-domain <domain>`
- `--vinfra-project <project>`

Alternatively, you can supply it by setting the following environment variables (for example, in your `~/.bash_profile`): `VINFRA_PORTAL`, `VINFRA_USERNAME`, `VINFRA_PASSWORD`, `VINFRA_DOMAIN`, and `VINFRA_PROJECT`. In this case, you will be able to run the CLI tool without the aforementioned command-line parameters.

As you typically run `vinfra` from the management node as admin, the only variable you usually need to set is the password. For example:

```
# export VINFRA_PASSWORD=12345
```

If you installed `vinfra` on a remote machine and/or run it as a different system administartor, you will need to set `VINFRA_PORTAL` and/or `VINFRA_USERNAME` on that machine in addition to `VINFRA_PASSWORD`.

In addition, if you want to authenticate within a different project or/and domain, you will need to set two more environment variables: `VINFRA_PROJECT` and/or `VINFRA_DOMAIN`.

Managing tasks

The `vinfra` CLI tool executes some commands immediately, while for other commands (that may take some time to complete) it creates system tasks that are queued. Examples of actions performed via tasks are creating the storage or compute cluster and adding nodes to it.

To keep track of tasks being performed by `vinfra`, use the `vinfra task list` and `vinfra task show` commands. For example:

```
# vinfra task list
+---------------+---------+----------------------------------------+
| task_id       | state   | name                                   |
+---------------+---------+----------------------------------------+
| 8fc27e7a-<...> | success | backend.tasks.cluster.CreateNewCluster  |
| e61377db-<...> | success | backend.tasks.disks.ApplyDiskRoleTask   |
| a005b748-<...> | success | backend.tasks.node.AddNodeInClusterTask |
+---------------+---------+----------------------------------------+
# vinfra task show 8fc27e7a-ba73-471d-9134-e351e1137cf4
+---------+----------------------------------------+
| Field   | Value                                  |
+---------+----------------------------------------+
| args    | - stor1                                |
|         | - 7ffa9540-5a20-41d1-b203-e3f349d62565 |
|         | - null                                 |
|         | - null                                 |
| kwargs  | {}                                     |
| name    | backend.tasks.cluster.CreateNewCluster |
| result  | cluster_id: 1                          |
| state   | success                                |
| task_id | 8fc27e7a-ba73-471d-9134-e351e1137cf4   |
+---------+----------------------------------------+
```

Managing the storage cluster

Managing tokens

vinfra node token show

Display the backend token:

```
usage: vinfra node token show
```

Example:

```
# vinfra node token show
+-------+---------------+
| Field | Value         |
+-------+---------------+
| host  | 10.37.130.101 |
| token | dc56d4d2      |
| ttl   | 86398         |
+-------+---------------+
```

This command shows the details of the current token.

vinfra node token create

Create the backend token:

```
usage: vinfra node token create [--ttl <ttl>]
```

`--ttl <ttl>`

Token TTL, in seconds

Example:

```
# vinfra node token create --ttl 86400
+-------+---------------+
| Field | Value         |
+-------+---------------+
| host  | 10.37.130.101 |
| token | dc56d4d2      |
| ttl   | 86398         |
+-------+---------------+
```

This command creates a new token with the time to live (TTL) of 86400 seconds.

vinfra node token validate

Validate the backend token:

```
usage: vinfra node token validate <token>
```

`<token>`

Token value

Example:

```
# vinfra node token validate dc56d4d2
+--------+-------+
| Field  | Value |
+--------+-------+
| status | valid |
+--------+-------+
```

This command validates the token `dc56d4d2`.

Managing networks

vinfra cluster network create

Create a new network:

```
usage: vinfra cluster network create [--traffic-types <traffic-types>]
                                     [--inbound-allow-list <addresses>]
                                     [--inbound-deny-list <addresses>]
                                     [--outbound-allow-list <rules>]
                                     <network-name>
```

`--traffic-types <traffic-types>`

A comma-separated list of traffic type IDs or names

`--inbound-allow-list <addresses>`

A comma-separated list of IP addresses

`--inbound-deny-list <addresses>`

A comma-separated list of IP addresses

`--outbound-allow-list <rules>`

A comma-separated list of allow rules in the format:
`<address>:<protocol>:<port>:<description>`

`<network-name>`

Network name

Example:

```
# vinfra cluster network create MyNet --traffic-types ssh
+---------------------+----------------------------------------+
```

```
| Field               | Value                                      |
+---------------------+--------------------------------------------+
| id                  | b451c5ed-a553-4214-96c4-d926daa6110e       |
| inbound_allow_list  | []                                         |
| inbound_deny_list   | []                                         |
| name                | MyNet                                      |
| outbound_allow_list | - 0.0.0.0:tcp:8888:Internal management     |
|                     | - 0.0.0.0:tcp:80:HTTP                      |
|                     | - 0.0.0.0:tcp:443:HTTPS                    |
|                     | - 0.0.0.0:udp:53:DNS                       |
|                     | - 0.0.0.0:tcp:53:DNS                       |
|                     | - 0.0.0.0:udp:123:NTP                      |
|                     | - 0.0.0.0:tcp:8443:ABGW registration       |
|                     | - 0.0.0.0:tcp:44445:ABGW Geo-replication   |
|                     | - 0.0.0.0:tcp:9877:Acronis Cyber Protect   |
|                     | - 0.0.0.0:any:0:Allow all                  |
| name                | MyNet                                      |
| traffic_types       | SSH                                        |
| vlan                |                                            |
+---------------------+--------------------------------------------+
```

This command creates a custom network `MyNet` and assigns the traffic type `SSH` to it.

vinfra cluster network list

List available networks:

```
usage: vinfra cluster network list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra cluster network list -c id -c name -c traffic_types
+--------------------------------------+---------+------------------------+
| id                                   | name    | traffic_types          |
+--------------------------------------+---------+------------------------+
| 358bdc39-cd8b-4565-8ebf-e7c12dcd1cf7 | Public  | Backup (ABGW) public,  |
|                                      |         | iSCSI,NFS,S3 public,   |
|                                      |         | SSH,Admin Panel        |
| 6095a997-e5f1-493d-a750-41ddf277153b | Private | Backup (ABGW) private, |
|                                      |         | Internal Management,   |
|                                      |         | OSTOR private,SSH,     |
|                                      |         | Storage                |
+--------------------------------------+---------+------------------------+
```

This command lists all networks in Virtuozzo Hybrid Infrastructure.

vinfra cluster network show

Show details of a network:

```
usage: vinfra cluster network show <network>
```

`<network>`

Network ID or name

Example:

```
# vinfra cluster network show MyNet
+--------------------+-----------------------------------------+
| Field              | Value                                   |
+--------------------+-----------------------------------------+
| id                 | b451c5ed-a553-4214-96c4-d926daa6110e    |
| inbound_allow_list | []                                      |
| inbound_deny_list  | []                                      |
| name               | MyNet                                   |
| outbound_allow_list | - 0.0.0.0:tcp:8888:Internal management  |
|                    | - 0.0.0.0:tcp:80:HTTP                   |
|                    | - 0.0.0.0:tcp:443:HTTPS                 |
|                    | - 0.0.0.0:udp:53:DNS                    |
|                    | - 0.0.0.0:tcp:53:DNS                    |
|                    | - 0.0.0.0:udp:123:NTP                   |
|                    | - 0.0.0.0:tcp:8443:ABGW registration    |
|                    | - 0.0.0.0:tcp:44445:ABGW Geo-replication |
|                    | - 0.0.0.0:tcp:9877:Acronis Cyber Protect |
|                    | - 0.0.0.0:any:0:Allow all               |
| name               | MyNet                                   |
| traffic_types      | SSH                                     |
| vlan               |                                         |
+--------------------+-----------------------------------------+
```

This command shows the details of the custom network `MyNet`.

vinfra cluster network set

Modify network parameters:

```
usage: vinfra cluster network set [--name <network-name>]
                                  [--traffic-types <traffic-types> |
                                  --add-traffic-types <traffic-types> |
                                  --del-traffic-types <traffic-types>]
                                  [--inbound-allow-list <addresses> |
                                  --add-inbound-allow-list <addresses> |
                                  --del-inbound-allow-list <addresses> |
                                  --clear-inbound-allow-list]
                                  [--inbound-deny-list <addresses> |
```

```
                                --add-inbound-deny-list <addresses> |
                                --del-inbound-deny-list <addresses> |
                                --clear-inbound-deny-list]
                                [--outbound-allow-list <rules> |
                                --add-outbound-allow-list <rules> |
                                --del-outbound-allow-list <rules> |
                                --clear-outbound-allow-list |
                                --restore-default-outbound-allow-list]
                                [-y] <network>
```

`--name <network-name>`

Network name

`--traffic-types <traffic-types>`

A comma-separated list of traffic type names (overwrites network's current traffic types)

`--add-traffic-types <traffic-types>`

A comma-separated list of traffic type names (adds the specified traffic types to the network)

`--del-traffic-types <traffic-types>`

A comma-separated list of traffic type names (removes the specified traffic types from the network)

`--inbound-allow-list <addresses>`

A comma-separated list of IP addresses (overwrites the current inbound allow rules)

`--add-inbound-allow-list <addresses>`

A comma-separated list of IP addresses (adds the specified inbound allow rules)

`--del-inbound-allow-list <addresses>`

A comma-separated list of IP addresses (removes the specified inbound allow rules)

`--clear-inbound-allow-list`

Clear all inbound allow rules

`--inbound-deny-list <addresses>`

A comma-separated list of IP addresses (overwrites the current inbound deny rules)

`--add-inbound-deny-list <addresses>`

A comma-separated list of IP addresses (adds the specified inbound deny rules)

`--del-inbound-deny-list <addresses>`

A comma-separated list of IP addresses (removes the specified inbound deny rules)

`--clear-inbound-deny-list <addresses>`

Clear all inbound deny rules

`--outbound-allow-list <rules>`

A comma-separated list of allow rules in the format: `<address>:<protocol>:<port>:<description>` (overwrites the current outbound allow rules)

`--add-outbound-allow-list <rules>`

A comma-separated list of allow rules in the format:
`<address>:<protocol>:<port>:<description>` (adds the specified outbound allow rules)

`--del-outbound-allow-list <rules>`

A comma-separated list of allow rules in the format:
`<address>:<protocol>:<port>:<description>` (removes the specified outbound allow rules)

`--clear-outbound-allow-list <rules>`

Clear all outbound allow rules (needs confirmation)

`--restore-outbound-allow-list <rules>`

Restore default outbound allow rules

`-y, --yes`

Skip yes/no prompt (assumes "yes")

`<network>`

Network ID or name

Example:

```
# vinfra cluster network set MyNet --name MyOtherNet \
--add-traffic-types iscsi,nfs
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | b29f6f66-37d7-47de-b02e-9f4087ad932b |
+---------+--------------------------------------+
```

This command creates a task to rename the network `MyNet` to `MyOtherNet` and assign to it the traffic types `iSCSI` and `NFS`.

Task outcome:

```
# vinfra task show b29f6f66-37d7-47de-b02e-9f4087ad932b
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.network.roles.tasks.RolesSetChangeTask |
| result  | id: 2c030529-8f7e-46cc-8011-6c9d5b46fd49                    |
|         | inbound_allow_list: []                                           |
|         | inbound_deny_list: []                                            |
|         | name: MyOtherNet                                                 |
|         | outbound_allow_list:                                             |
|         | - 0.0.0.0:tcp:8888:Internal management                           |
|         | - 0.0.0.0:tcp:80:HTTP                                            |
|         | - 0.0.0.0:tcp:443:HTTPS                                          |
|         | - 0.0.0.0:udp:53:DNS                                             |
```

```
|         | - 0.0.0.0:tcp:53:DNS                                          |
|         | - 0.0.0.0:udp:123:NTP                                         |
|         | - 0.0.0.0:tcp:8443:ABGW registration                          |
|         | - 0.0.0.0:tcp:44445:ABGW Geo-replication                      |
|         | - 0.0.0.0:tcp:9877:Acronis Cyber Protect                      |
|         | - 0.0.0.0:any:0:Allow all                                     |
|         | roles:                                                        |
|         | - iSCSI                                                       |
|         | - NFS                                                         |
|         | - SSH                                                         |
|         | vlan: null                                                    |
| state   | success                                                       |
| task_id | b29f6f66-37d7-47de-b02e-9f4087ad932b                          |
+---------+---------------------------------------------------------------+
```

vinfra cluster network set-bulk

Modify traffic types of multiple networks:

```
usage: vinfra cluster network set-bulk --network <network>:<traffic-types>
```

`--network <network>:<traffic-types>`

Network configuration in the format:

- `<network>`: network ID or name.
- `<traffic-types>`: a comma-separated list of traffic type names

This option can be used multiple times.

Example:

```
# vinfra cluster network set-bulk --network MyNet1:snmp --network MyNet2:ssh,snmp
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | c774f55d-c45b-42cd-ac9e-16fc196e9283 |
+---------+--------------------------------------+
```

This command creates a task to change the traffic type set of the network `MyNet1` to `SNMP` and that of `MyNet2` to `SSH` and `SNMP`.

Task outcome:

```
# vinfra task show c774f55d-c45b-42cd-ac9e-16fc196e9283
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.network.roles.tasks.RolesSetBulkChangeTask |
| result  | - id: adf49487-9deb-4180-bb0c-08a906257981                       |
```

```
|         |   name: MyNet1                                                  |
|         |   roles:                                                        |
|         |   - SNMP                                                        |
|         |   type: Custom                                                  |
|         | - id: 3f6ff4a3-31bc-440b-a36f-d755c80d5932                      |
|         |   name: MyNet2                                                  |
|         |   roles:                                                        |
|         |   - SNMP                                                        |
|         |   - SSH                                                         |
|         |   type: Custom                                                  |
| state   | success                                                         |
| task_id | c774f55d-c45b-42cd-ac9e-16fc196e9283                            |
+---------+-----------------------------------------------------------------+
```

vinfra cluster network migration start

Important
If you have restricted outbound traffic in your cluster, you need to manually add a rule that will allow outbound traffic on on TCP and UDP ports 60000–60100, as described in "Restricting outbound traffic from cluster nodes" (p. 369).

Start network migration:

```
usage: vinfra cluster network migration start <network> [--subnet <subnet>]
                                              [--netmask <netmask>]
                                              [--gateway <gateway>] [--shutdown]
                                              [--node <node> <address>]
```

`--network <network>`
: Network ID or name

`--subnet <subnet>`
: New network subnet

`--netmask <netmask>`
: New network mask

`--gateway <gateway>`
: New network gateway

`--shutdown`
: Prepare the cluster to be shut down manually for relocation

`--node <node> <address>`
: New node address in the format:

- `<node>`: node ID or hostname

- <address>: IPv4 address

This option can be used multiple times.

Example:

```
# vinfra cluster network migration start --network "Private" \
--subnet 192.168.128.0 --netmask 255.255.255.0 --node node001 192.168.128.11 \
--node node002 192.168.128.12 --node node003 192.168.128.13
+---------------------------+-------------------------------------------------+
| Field                     | Value                                           |
+---------------------------+-------------------------------------------------+
| configuration             | network_id: 3e3619b7-2c93-4e90-a187-135c6f8b9060 |
| link                      | href: /api/v2/network/migration/2d4ec3a9-<...>/  |
|                           | method: GET                                     |
|                           | rel: network-migration-details                  |
| operation                 | network-migration                               |
| progress                  | 0.0                                             |
| single_interface_migration | False                                          |
| state                     | preparing                                       |
| task_id                   | 2d4ec3a9-7714-479d-a03c-1efbe6ffecf5            |
| transitions               | 0                                               |
+---------------------------+-------------------------------------------------+
```

This command starts migration of the `Private` network to the new network configuration.

vinfra cluster network migration show

Display network migration details:

```
usage: vinfra cluster network migration show [--full] [--task-id <task-id>]
```

`--full`

Show full information

`--task-id <task-id>`

The task ID of network migration

Example:

```
# vinfra cluster network migration show
+---------------------------+-------------------------------------------------+
| Field                     | Value                                           |
+---------------------------+-------------------------------------------------+
| link                      | href: /api/v2/network/migration/2d4ec3a9-<...>/ |
|                           | method: GET                                     |
|                           | rel: network-migration-details                  |
| operation                 | network-migration                               |
| progress                  | 1.0                                             |
| single_interface_migration | False                                          |
| state                     | test-passed                                     |
```

```
| task_id                    | 2d4ec3a9-7714-479d-a03c-1efbe6ffecf5           |
| transitions                | 5                                              |
+----------------------------+------------------------------------------------+
```

This command shows the details of the current network migration: the new network configuration has been tested and can be applied.

vinfra cluster network migration apply

Continue network migration to apply the new network configuration:

```
usage: vinfra cluster network migration apply
```

Example:

```
# vinfra cluster network migration apply
+----------------------------+------------------------------------------------+
| Field                      | Value                                          |
+----------------------------+------------------------------------------------+
| link                       | href: /api/v2/network/migration/2d4ec3a9-<...>/ |
|                            | method: GET                                    |
|                            | rel: network-migration-details                 |
| operation                  | network-migration                              |
| progress                   | 1.0                                            |
| single_interface_migration | False                                          |
| state                      | test-passed                                    |
| task_id                    | 2d4ec3a9-7714-479d-a03c-1efbe6ffecf5           |
| transitions                | 5                                              |
+----------------------------+------------------------------------------------+
```

This command continues the network migration and applies the new network configuration.

vinfra cluster network migration retry

Retry an operation for network migration:

```
usage: vinfra cluster network migration retry [--subnet <subnet>]
                                              [--netmask <netmask>]
                                              [--node <node> <address>]
```

`--subnet <subnet>`

New network subnet

`--netmask <netmask>`

New network mask

`--node <node> <address>`

New node address in the format:

- <node>: node ID or hostname
- <address>: IPv4 address

This option can be used multiple times.

Example:

```
# vinfra cluster network migration retry --subnet 192.168.128.0 \
--netmask 255.255.255.0 --node node001 192.168.128.12 --node node002 192.168.128.13 \
--node node003 192.168.128.14
+---------------------------+-------------------------------------------------+
| Field                     | Value                                           |
+---------------------------+-------------------------------------------------+
| link                      | href: /api/v2/network/migration/2d4ec3a9-<...>/ |
|                           | method: GET                                     |
|                           | rel: network-migration-details                  |
| operation                 | network-migration                               |
| progress                  | 0.9                                             |
| single_interface_migration | False                                           |
| state                     | failed-to-apply                                 |
| task_id                   | 2ce42f0e-6401-47c1-a52f-33e7c68d0df4            |
| transitions               | 5                                               |
+---------------------------+-------------------------------------------------+
```

This command retries the failed operation for the network migration with new target IP addresses.

vinfra cluster network migration revert

Revert network migration:

```
usage: vinfra cluster network migration revert
```

Example:

```
# vinfra cluster network migration revert
+---------------------------+-------------------------------------------------+
| Field                     | Value                                           |
+---------------------------+-------------------------------------------------+
| link                      | href: /api/v2/network/migration/2d4ec3a9-<...>/ |
|                           | method: GET                                     |
|                           | rel: network-migration-details                  |
| operation                 | network-migration                               |
| progress                  | 1.0                                             |
| single_interface_migration | False                                           |
| state                     | test-passed                                     |
| task_id                   | 2d4ec3a9-7714-479d-a03c-1efbe6ffecf5            |
| transitions               | 5                                               |
+---------------------------+-------------------------------------------------+
```

This command reverts the network configuration to the previous one.

vinfra cluster network migration resume

Resume network migration after the cluster shutdown:

```
usage: vinfra cluster network migration resume
```

Example:

```
# vinfra cluster network migration resume
+---------------------------+-------------------------------------------------+
| Field                     | Value                                           |
+---------------------------+-------------------------------------------------+
| link                      | href: /api/v2/network/migration/2d4ec3a9-<...>/ |
|                           | method: GET                                     |
|                           | rel: network-migration-details                  |
| operation                 | network-migration                               |
| progress                  | 1.0                                             |
| single_interface_migration | False                                           |
| state                     | test-passed                                     |
| task_id                   | 2d4ec3a9-7714-479d-a03c-1efbe6ffecf5            |
| transitions               | 5                                               |
+---------------------------+-------------------------------------------------+
```

This command resumes the network migration after the cluster has been manually shut down and relocated.

vinfra cluster network reconfiguration show

Display network reconfiguration details:

```
usage: vinfra cluster network reconfiguration show
```

Example:

```
# vinfra cluster network reconfiguration show
+-----------+-----------------------------------------------------------------+
| Field     | Value                                                           |
+-----------+-----------------------------------------------------------------+
| link      | href: /api/v2/network/traffic-type-assignment/285be91b-<...>/ |
|           | method: GET                                                     |
|           | rel: traffic-type-assignment-details                            |
| operation | traffic-type-assignment                                         |
| task_id   | 285be91b-77ee-4f8f-a118-8410ab792148                            |
+-----------+-----------------------------------------------------------------+
```

This command shows the details about the current network reconfiguration: traffic type assignment is in progress.

vinfra cluster network conversion precheck

Check VLAN network interfaces to Open vSwitch VLAN conversion:

```
usage: vinfra cluster network conversion precheck --network <network>
                                                  [--physical-network-name <name>]
```

`--network <network>`

The ID or name of the network, which is connected to a VLAN interface

`--physical-network-name <name>`

The name of a new infrastructure network for trunk network interfaces. Specify this option if your trunk network interfaces are not assigned to any infrastructure network. The new infrastructure network will be automatically created with the given name and assigned to the trunk interfaces.

Example:

```
# vinfra cluster network conversion precheck --network mynet
+--------------------+-------------------------------------------------+
| Field              | Value                                           |
+--------------------+-------------------------------------------------+
| affected_interfaces | - interface: eth0                              |
|                    |   node_id: 13cb6cbf-0b9b-be0f-bb56-8ed6a0e9225c |
|                    |   vlans:                                        |
|                    |   - eth0.1                                      |
|                    | - interface: eth0                               |
|                    |   node_id: 6e5d9e91-5c4e-a874-38cd-fe6f4bef10a4 |
|                    |   vlans:                                        |
|                    |   - eth0.1                                      |
|                    | - interface: eth0                               |
|                    |   node_id: 1053e85b-351c-6113-5623-e0c6c64995e7 |
|                    |   vlans:                                        |
|                    |   - eth0.1                                      |
| affected_networks  | - mynet                                         |
| physical_network   | Public                                          |
+--------------------+-------------------------------------------------+
```

This command checks whether VLAN network interfaces connected to the network `mynet` can be converted to Open vSwitch VLAN.

vinfra cluster network conversion start

Convert VLAN network interfaces to Open vSwitch VLAN, and connect a new network to physical interfaces if they have no assignment:

```
usage: vinfra cluster network conversion start --network <network>
                                               [--physical-network-name <name>]
```

`--network <network>`

The ID or name of the network, which is connected to VLAN interfaces

`--physical-network-name <name>`

The name of a new infrastructure network for trunk network interfaces. Specify this option if your trunk network interfaces are not assigned to any infrastructure network. The new infrastructure network will be automatically created with the given name and assigned to the trunk interfaces.

Example:

```
# vinfra cluster network conversion start --network mynet
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 058fc247-03a8-49fa-90e1-1e073dbafec9 |
+---------+--------------------------------------+
```

This command creates a task to convert VLAN network interfaces connected to the network `mynet` to Open vSwitch VLAN.

Task outcome:

```
# vinfra task show 058fc247-03a8-49fa-90e1-1e073dbafec9
+---------+-------------------------------------------------------------------------+
| Field   | Value                                                                   |
+---------+-------------------------------------------------------------------------+
| details |                                                                         |
| name    | backend.business.models.network.convert2ovs.tasks.Convert2OVSTask |
| result  |                                                                         |
| state   | success                                                                 |
| task_id | 058fc247-03a8-49fa-90e1-1e073dbafec9                                    |
+---------+-------------------------------------------------------------------------+
```

vinfra cluster network conversion status

Get VLAN network interface conversion status:

```
usage: vinfra cluster network conversion status <task>
```

`<task>`

Task ID

Example:

```
# vinfra cluster network conversion status 058fc247-03a8-49fa-90e1-1e073dbafec9
+--------------------+-------------------------------------------------+
| Field              | Value                                           |
+--------------------+-------------------------------------------------+
```

```
| affected_interfaces | - interface: eth0                                  |
|                     |   node_id: 13cb6cbf-0b9b-be0f-bb56-8ed6a0e9225c |
|                     |   vlans:                                           |
|                     |   - eth0.1                                         |
|                     | - interface: eth0                                  |
|                     |   node_id: 6e5d9e91-5c4e-a874-38cd-fe6f4bef10a4 |
|                     |   vlans:                                           |
|                     |   - eth0.1                                         |
|                     | - interface: eth0                                  |
|                     |   node_id: 1053e85b-351c-6113-5623-e0c6c64995e7 |
|                     |   vlans:                                           |
|                     |   - eth0.1                                         |
| flow                | done                                               |
| physical_network    | Public                                             |
| state               | success                                            |
| task_id             | 058fc247-03a8-49fa-90e1-1e073dbafec9               |
+---------------------+----------------------------------------------------+
```

This command shows the details of the VLAN network interface conversion.

vinfra cluster network delete

Delete a network:

```
usage: vinfra cluster network delete <network>
```

`<network>`

Network ID or name

Example:

```
# vinfra cluster network delete MyOtherNet
Operation successful
```

This command deletes the network `MyOtherNet`.

Managing traffic types

vinfra cluster traffic-type create

Create a new traffic type:

```
usage: vinfra cluster traffic-type create --port <port>
                                          [--inbound-allow-list <addresses>]
                                          [--inbound-deny-list <addresses>]
                                          <traffic-type-name>
```

`--port <port>`

Traffic type port

`--inbound-allow-list <addresses>`

A comma-separated list of IP addresses

`--inbound-deny-list <addresses>`

A comma-separated list of IP addresses

`<traffic-type-name>`

Traffic type name

Example:

```
# vinfra cluster traffic-type create "MyTrafficType" --port 6900
+--------------------+---------------+
| Field              | Value         |
+--------------------+---------------+
| exclusive          | False         |
| hidden             | False         |
| inbound_allow_list | []            |
| inbound_deny_list  | []            |
| name               | MyTrafficType |
| port               | 6900          |
| type               | custom        |
+--------------------+---------------+
```

This command creates a custom traffic type `MyTrafficType` on port 6900.

vinfra cluster traffic-type list

List available traffic types:

```
usage: vinfra cluster traffic-type list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra cluster traffic-type list -c name -c type -c exclusive -c port
+----------------------+------------+-----------+------+
| name                 | type       | exclusive | port |
+----------------------+------------+-----------+------+
| Storage              | predefined | True      |      |
| Internal management  | predefined | True      |      |
| OSTOR private        | predefined | True      |      |
| S3 public            | predefined | False     |      |
| iSCSI                | predefined | False     |      |
| NFS                  | predefined | False     |      |
| Backup (ABGW) private | predefined | True      |      |
| Backup (ABGW) public | predefined | False     |      |
| Admin panel          | predefined | False     |      |
```

```
| SSH                   | predefined | False     |      |
| VM public             | predefined | False     |      |
| VM private            | predefined | True      |      |
| Compute API           | predefined | True      |      |
| MyTrafficType         | custom     | False     | 6900 |
+-----------------------+------------+-----------+------+
```

This command lists all traffic types in Virtuozzo Hybrid Infrastructure.

vinfra cluster traffic-type show

Show details of a traffic type:

```
usage: vinfra cluster traffic-type show <traffic-type>
```

`<traffic-type>`

Traffic type name

Example:

```
# vinfra cluster traffic-type show Storage
+--------------------+------------+
| Field              | Value      |
+--------------------+------------+
| exclusive          | True       |
| hidden             | False      |
| inbound_allow_list | []         |
| inbound_deny_list  | []         |
| name               | Storage    |
| port               |            |
| type               | predefined |
+--------------------+------------+
```

This command shows the details of the traffic type `Storage`.

vinfra cluster traffic-type set

Modify traffic type parameters:

```
usage: vinfra cluster traffic-type set [--name <name>] [--port <port>]
                                       [--inbound-allow-list <addresses> |
                                       --add-inbound-allow-list <addresses> |
                                       --del-inbound-allow-list <addresses> |
                                       --clear-inbound-allow-list]
                                       [--inbound-deny-list <addresses> |
                                       --add-inbound-deny-list <addresses> |
                                       --del-inbound-deny-list <addresses> |
                                       --clear-inbound-deny-list]
                                       <traffic-type>
```

`--name <name>`

A new name for the traffic type

`--port <port>`

A new port for the traffic type

`--inbound-allow-list <addresses>`

A comma-separated list of IP addresses (overwrites the current inbound allow rules)

`--add-inbound-allow-list <addresses>`

A comma-separated list of IP addresses (adds the specified inbound allow rules)

`--del-inbound-allow-list <addresses>`

A comma-separated list of IP addresses (removes the specified inbound allow rules)

`--clear-inbound-allow-list`

Clear all inbound allow rules

`--inbound-deny-list <addresses>`

A comma-separated list of IP addresses (overwrites the current inbound deny rules)

`--add-inbound-deny-list <addresses>`

A comma-separated list of IP addresses (adds the specified inbound deny rules)

`--del-inbound-deny-list <addresses>`

A comma-separated list of IP addresses (removes the specified inbound deny rules)

`--clear-inbound-deny-list <addresses>`

Clear all inbound deny rules

`<traffic-type>`

Traffic type name

Example:

```
# vinfra cluster traffic-type set "MyTrafficType" \
--name "MyOtherTrafficType" --port 6901
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 33222e65-eb27-4181-ba23-0ebcd43766d7 |
+---------+--------------------------------------+
```

This command creates a task to rename the traffic type `MyTrafficType` to `MyOtherTrafficType` and change its port to 6901.

Task outcome:

```
# vinfra task show 33222e65-eb27-4181-ba23-0ebcd43766d7
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
```



```
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.presentation.network.roles.tasks.RoleChangeTask |
| result  | exclusive: false                                    |
|         | hidden: false                                       |
|         | inbound_allow_list: []                              |
|         | inbound_deny_list: []                               |
|         | name: MyOtherTrafficType                            |
|         | port: 6901                                          |
|         | type: custom                                        |
| state   | success                                             |
| task_id | 33222e65-eb27-4181-ba23-0ebcd43766d7                |
+---------+-----------------------------------------------------+
```

vinfra cluster traffic-type assignment start

Start traffic type assignment:

```
usage: vinfra cluster traffic-type assignment start --traffic-type <traffic-type>
                                                    --target-network <target-network>
```

`--traffic-type <traffic-type>`

Traffic type name

`--target-network <target-network>`

Target network ID or name

Example:

```
# vinfra cluster traffic-type assignment start --traffic-type Storage \
--target-network Public
+---------------+-------------------------------------------------------------+
| Field         | Value                                                       |
+---------------+-------------------------------------------------------------+
| configuration | target_network: 69ad1db5-512f-4994-ab08-7d643fdb7b39        |
|               | traffic_type: Storage                                       |
| link          | href: /api/v2/network/traffic-type-assignment/285be91b-<...>/ |
|               | method: GET                                                 |
|               | rel: traffic-type-assignment-details                        |
| operation     | traffic-type-assignment                                     |
| progress      | 0.0                                                         |
| state         | preparing                                                   |
| task_id       | 285be91b-77ee-4f8f-a118-8410ab792148                        |
| transitions   | 0                                                           |
+---------------+-------------------------------------------------------------+
```

This command starts assignment of the `Storage` traffic type to the `Public` network.

vinfra cluster traffic-type assignment show

Display traffic type assignment details:

```
usage: vinfra cluster traffic-type assignment show [--full] [--task-id <task-id>]
```

`--full`

Show full information

`-task-id <task-id>`

The task ID of traffic type assignment

Example:

```
# vinfra cluster traffic-type assignment show
+-------------+-----------------------------------------------------------------+
| Field       | Value                                                           |
+-------------+-----------------------------------------------------------------+
| link        | href: /api/v2/network/traffic-type-assignment/285be91b-<...>/ |
|             | method: GET                                                     |
|             | rel: traffic-type-assignment-details                            |
| operation   | traffic-type-assignment                                         |
| progress    | 1.0                                                             |
| state       | test-passed                                                     |
| task_id     | 285be91b-77ee-4f8f-a118-8410ab792148                            |
| transitions | 3                                                               |
+-------------+-----------------------------------------------------------------+
```

This command shows the details of the current traffic type assignment: the new network configuration has been tested and can be applied.

vinfra cluster traffic-type assignment apply

Continue traffic type assignment to apply the new network configuration:

```
usage: vinfra cluster traffic-type assignment apply
```

Example:

```
# vinfra cluster traffic-type assignment apply
+-------------+-----------------------------------------------------------------+
| Field       | Value                                                           |
+-------------+-----------------------------------------------------------------+
| link        | href: /api/v2/network/traffic-type-assignment/285be91b-<...>/ |
|             | method: GET                                                     |
|             | rel: traffic-type-assignment-details                            |
| operation   | traffic-type-assignment                                         |
| progress    | 1.0                                                             |
```

```
| state       | test-passed                                                    |
| task_id     | 285be91b-77ee-4f8f-a118-8410ab792148                           |
| transitions | 3                                                              |
+-------------+----------------------------------------------------------------+
```

This command continues the traffic type assignment and applies the new network configuration.

vinfra cluster traffic-type assignment retry

Retry an operation for traffic type assignment:

```
usage: vinfra cluster traffic-type assignment retry
```

Example:

```
# vinfra cluster traffic-type assignment retry
+-------------+----------------------------------------------------------------+
| Field       | Value                                                          |
+-------------+----------------------------------------------------------------+
| link        | href: /api/v2/network/traffic-type-assignment/f633af90-<...>/ |
|             | method: GET                                                    |
|             | rel: traffic-type-assignment-details                           |
| operation   | traffic-type-assignment                                        |
| progress    | 0.444444444444                                                 |
| state       | test-failed                                                    |
| task_id     | f633af90-302e-4299-8055-d3e400dc0ea7                           |
| transitions | 3                                                              |
+-------------+----------------------------------------------------------------+
```

This command retries the failed operation for the traffic type assignment.

vinfra cluster traffic-type assignment revert

Revert traffic type assignment:

```
usage: vinfra cluster traffic-type assignment revert
```

Example:

```
# vinfra cluster traffic-type assignment revert
+-------------+----------------------------------------------------------------+
| Field       | Value                                                          |
+-------------+----------------------------------------------------------------+
| link        | href: /api/v2/network/traffic-type-assignment/f633af90-<...>/ |
|             | method: GET                                                    |
|             | rel: traffic-type-assignment-details                           |
| operation   | traffic-type-assignment                                        |
| progress    | 1.0                                                            |
| state       | test-passed                                                    |
```

```
| task_id     | f633af90-302e-4299-8055-d3e400dc0ea7                          |
| transitions | 5                                                             |
+-------------+---------------------------------------------------------------+
```

This command reverts the traffic type assignment to the previous network configuration.

vinfra cluster traffic-type delete

Delete a traffic type:

```
usage: vinfra cluster traffic-type delete <traffic-type>
```

`<traffic-type>`

Traffic type name

Example:

```
# vinfra cluster traffic-type delete "MyOtherTrafficType"
Operation successful
```

This command deletes the custom traffic type `MyOtherTrafficType`.

Managing storage nodes

vinfra node join

Join a node to the storage cluster:

```
usage: vinfra node join [--disk <disk>:<role>[:<key=value,…>]] <node>
```

`--disk <disk>:<role> [:<key=value,…>]`

Disk configuration in the format:

- `<disk>`: disk device ID or name
- `<role>`: disk role (`cs`, `mds`, `journal`, `mds-journal`, `mds-system`, `cs-system`, `system`)
- comma-separated `key=value` pairs with keys (optional):
  - `tier`: disk tier (0, 1, 2 or 3)
  - `journal-tier`: journal (cache) disk tier (0, 1, 2 or 3)
  - `journal-type`: journal (cache) disk type (`no_cache`, `inner_cache` or `external_cache`)
  - `journal-disk`: journal (cache) disk ID or device name
  - `bind-address`: bind IP address for the metadata service

Example: `sda:cs:tier=0,journal-type=inner_cache`.

This option can be used multiple times.

<node>

Node ID or hostname

Example:

```
# vinfra node join f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4 \
--disk sda:mds-system \
--disk sdb:cs \
--disk sdc:cs
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | a2713068-9544-4ea1-8ec8-69a068cf86f2 |
+---------+--------------------------------------+
```

This command creates a task to add the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4` to the storage cluster and assigns roles to disks: `mds-system` to `sda`, `cs` to `sdb` and `sdc`.

Task outcome:

```
# vinfra task show a2713068-9544-4ea1-8ec8-69a068cf86f2
+---------+------------------------------------------+
| Field   | Value                                    |
+---------+------------------------------------------+
| args    | - f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4   |
|         | - 1                                      |
| kwargs  | disks:                                   |
|         | - id: 85F32403-94A9-465A-9E6C-C1A2B41294FC |
|         |   role: mds-system                       |
|         |   service_params: {}                     |
|         | - id: FE0B5876-E054-489B-B0FD-72429BEFD46A |
|         |   role: cs                               |
|         |   service_params: {}                     |
|         | - id: D3BEF4BB-AA3B-4DB6-9376-BC7CDA636700 |
|         |   role: cs                               |
|         |   service_params: {}                     |
| name    | backend.tasks.node.AddNodeInClusterTask  |
| result  | {}                                       |
| state   | success                                  |
| task_id | a2713068-9544-4ea1-8ec8-69a068cf86f2     |
+---------+------------------------------------------+
```

vinfra node list

List storage nodes:

```
usage: vinfra node list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra node list
+--------------+--------------+------------+-----------+-------------+----------+
| id           | host         | is_primary | is_online | is_assigned | is_in_ha |
+--------------+--------------+------------+-----------+-------------+----------+
| 09bb6b8<...> | node001<...> | True       | True      | True        | False    |
| 187edb1<...> | node002<...> | False      | True      | True        | False    |
| e6255ae<...> | node003<...> | False      | True      | True        | False    |
+--------------+--------------+------------+-----------+-------------+----------+
```

This command lists all nodes registered in Virtuozzo Hybrid Infrastructure (both unassigned and used in the storage cluster).

vinfra node show

Show storage node details:

```
usage: vinfra node show <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra node show 4f96acf5-3bc8-4094-bcb6-4d1953be7b55
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| cpu_cores     | 2                                    |
| host          | node001.vstoragedomain               |
| id            | 4f96acf5-3bc8-4094-bcb6-4d1953be7b55 |
| ipaddr        | node001.vstoragedomain               |
| is_assigned   | False                                |
| is_in_ha      | False                                |
| is_installing | False                                |
| is_online     | True                                 |
| is_primary    | True                                 |
| is_virt       | True                                 |
| mem_total     | 8201310208                           |
| roles         | management:                          |
|               |   is_primary: true                   |
| tasks         |                                      |
+---------------+--------------------------------------+
```

This command shows the details of the node with the ID `4f96acf5-3bc8-4094-bcb6-4d1953be7b55`.

vinfra node maintenance precheck

Start node maintenance precheck:

```
usage: vinfra node maintenance precheck <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra node maintenance precheck 9dcc9632-911c-4cc5-9a89-5a6fa5db2314
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 7c7f0afa-10f4-41b7-9b2e-973f3d392178 |
+---------+--------------------------------------+
```

This command creates a task to start maintenance precheck for the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`.

Task outcome:

```
# vinfra task show 7c7f0afa-10f4-41b7-9b2e-973f3d392178
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.business.models.maintenance.tasks.MaintenancePrecheckTask |
| result  |                                                                        |
| state   | success                                                                |
| task_id | 7c7f0afa-10f4-41b7-9b2e-973f3d392178                                   |
+---------+------------------------------------------------------------------------+
```

vinfra node maintenance start

Start node maintenance:

```
usage: vinfra node maintenance start [--iscsi-mode <mode>]
                                     [--compute-mode <mode>] [--s3-mode <mode>]
                                     [--storage-mode <mode>] [--alua-mode <mode>]
                                     [--nfs-mode <mode>] <node>
```

`--iscsi-mode <mode>`

Ignore ISCSI evacuation during maintenance (`ignore`).

`--compute-mode <mode>`

Ignore compute evacuation during maintenance (`ignore`).

`--s3-mode <mode>`

Ignore S3 evacuation during maintenance (`ignore`).

`--storage-mode <mode>`

Ignore storage evacuation during maintenance (`ignore`).

`--alua-mode <mode>`

Ignore Block Storage target groups during maintenance (`ignore`).

`--nfs-mode <mode>`

Ignore NFS evacuation during maintenance (`ignore`).

`<node>`

Node ID or hostname

Example:

```
# vinfra node maintenance start 9dcc9632-911c-4cc5-9a89-5a6fa5db2314 \
--iscsi-mode ignore --compute-mode ignore
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 3d4c23b6-9f62-412c-ad7c-ec9537a36fa7 |
+---------+--------------------------------------+
```

This command creates a task to start maintenance for the node with the ID `9dcc9632-911c-4cc5-9a89-5a6fa5db2314` without evacuating its iSCSI and compute services.

Task outcome:

```
# vinfra task show 3d4c23b6-9f62-412c-ad7c-ec9537a36fa7
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.business.models.maintenance.tasks.MaintenanceStartTask |
| result  |                                                                  |
| state   | success                                                          |
| task_id | 3d4c23b6-9f62-412c-ad7c-ec9537a36fa7                             |
+---------+------------------------------------------------------------------+
```

vinfra node maintenance status

Show node maintenance details:

```
usage: vinfra node maintenance status <node>
```

`<node>`

Node ID or hostname

Example:

```
# vinfra node maintenance status 9dcc9632-911c-4cc5-9a89-5a6fa5db2314
+-----------+----------------------------------------------+
| Field     | Value                                        |
```

```
+-----------+-----------------------------------------+
| node_id   | 9dcc9632-911c-4cc5-9a89-5a6fa5db2314    |
| params    | compute_mode: ignore                    |
|           | iscsi_mode: ignore                      |
|           | nfs_mode: evacuate                      |
|           | s3_mode: evacuate                       |
|           | storage_mode: suspend                   |
| precheck  |                                         |
| resources | compute:                                |
|           |   failed: []                            |
|           |   handled: []                           |
|           |   initial:                              |
|           |   - id: fa69e3f1-461c-4f4a-b442-3ffb356130c1 |
|           |     status: ACTIVE                      |
|           |   - id: f336f631-cc82-43e6-a582-8dcfcd24a722 |
|           |     status: ACTIVE                      |
|           |   - id: 48864c3f-d9bf-4c32-abdc-901395c3b5f9 |
|           |     status: ACTIVE                      |
|           |   - id: 5fd82e2a-3fef-4171-bfa4-67daa99ae64f |
|           |     status: ACTIVE                      |
|           |   untouched: []                         |
|           | iscsi:                                  |
|           |   failed: []                            |
|           |   handled: []                           |
|           |   initial: []                           |
|           | nfs: null                               |
| state     | suspended_complete                      |
| task      | flow: complete                          |
|           | id: fe8e5ff5-48c1-4a23-a533-28233aaae4db |
|           | state: success                          |
|           | updated_at: '2019-11-20T13:52:15.849492' |
+-----------+-----------------------------------------+
```

This command shows maintenance details for the node with the ID `9dcc9632-911c-4cc5-9a89-5a6fa5db2314`.

vinfra node maintenance stop

Return node to operation:

```
usage: vinfra node maintenance stop <node> [--ignore-compute]
```

<node>

Node ID or hostname

`--ignore-compute`

Ignore compute resources while returning a node to operation

Example:

```
# vinfra node maintenance stop 9dcc9632-911c-4cc5-9a89-5a6fa5db2314
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 34e0b546-aa2c-466c-93fe-7dff28c543c6 |
+---------+--------------------------------------+
```

This command creates a task to stop maintenance for the node with the ID `9dcc9632-911c-4cc5-9a89-5a6fa5db2314`.

Task outcome:

```
# vinfra task show 34e0b546-aa2c-466c-93fe-7dff28c543c6
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.business.models.maintenance.tasks.MaintenanceStopTask |
| result  |                                                                  |
| state   | success                                                          |
| task_id | 34e0b546-aa2c-466c-93fe-7dff28c543c6                             |
+---------+------------------------------------------------------------------+
```

vinfra node release

Release a node from the storage cluster. Start data migration from the node as well as cluster replication and rebalancing to meet the configured redundancy level:

```
usage: vinfra node release [--force] <node>
```

`--force`

Release node without data migration

<node>

Node ID or hostname

Example:

```
# vinfra node release f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | c2a653a2-8991-4b3a-8bdf-5c0872aa75b3 |
+---------+--------------------------------------+
```

This command creates a task to release the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4` from the storage cluster with migration of data to maintain the set redundancy mode.

Task outcome:


```
# vinfra task show c2a653a2-8991-4b3a-8bdf-5c0872aa75b3
+---------+----------------------------------------+
| Field   | Value                                  |
+---------+----------------------------------------+
| args    | - f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4 |
|         | - false                                |
| kwargs  | {}                                     |
| name    | backend.tasks.node.ReleaseNodeTask     |
| state   | success                                |
| task_id | c2a653a2-8991-4b3a-8bdf-5c0872aa75b3   |
+---------+----------------------------------------+
```

vinfra node forget

Remove a node from the storage cluster:

```
usage: vinfra node forget <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra node forget fd1e46de-6e17-4571-bf6b-1ac34ec1c225
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 0eac3b74-e8f5-4974-9efe-a9070187d83c |
+---------+--------------------------------------+
```

This command creates a task to unregister the node with the ID `fd1e46de-6e17-4571-bf6b-1ac34ec1c225` from Virtuozzo Hybrid Infrastructure.

Task outcome:

```
# vinfra task show 0eac3b74-e8f5-4974-9efe-a9070187d83c
+---------+----------------------------------------+
| Field   | Value                                  |
+---------+----------------------------------------+
| args    | - fd1e46de-6e17-4571-bf6b-1ac34ec1c225 |
| kwargs  | {}                                     |
| name    | backend.tasks.node.DeleteNodeTask      |
| state   | success                                |
| task_id | 0eac3b74-e8f5-4974-9efe-a9070187d83c   |
+---------+----------------------------------------+
```

Managing failure domains

vinfra failure domain list

Show available failure domains:

```
usage: vinfra failure domain list
```

Example:

```
# vinfra failure domain list
+----+----------+--------+
| id | singular | plural |
+----+----------+--------+
|  0 | disk     | disks  |
|  1 | host     | hosts  |
|  2 | rack     | racks  |
|  3 | row      | rows   |
|  4 | room     | rooms  |
+----+----------+--------+
```

This command lists the failure domains.

vinfra failure domain rename

Set names for failure domain levels, which define the storage location. These four levels are 1=host, 2=rack, 3=row, 4=room. The names for levels 2, 3 and 4 can be changed.

```
usage: vinfra failure domain rename [-h] {2,3,4} <singular-name> <plural-name>
```

`{2,3,4}`
: Failure domain ID.

`<singular-name>`
: Singular name of the specified failure domain.

`<plural-name>`
: Plural name of the specified failure domain.

Example:

```
# vinfra failure domain rename 2 chassis chassis
Operation successful.
```

This command renames the failure domain `2` to `chassis`.

Task outcome:

```
# vinfra failure domain list
+----+----------+---------+
| id | singular | plural  |
+----+----------+---------+
|  0 | disk     | disks   |
|  1 | host     | hosts   |
|  2 | chassis  | chassis |
|  3 | row      | rows    |
|  4 | room     | rooms   |
+----+----------+---------+
```

Note
If you use a name other than `zone`, `enclosure`, `chassis`, `blade server`, it will be replaced with **location** in the admin panel.

Managing node location

vinfra location list

List locations of the specified failure domain:

```
usage: vinfra location list --fd <fd>
```

`--fd <fd>`

Failure domain ID

Example:

```
# vinfra location list --fd 4
+----+--------------+----------+
| id | name         | children |
+----+--------------+----------+
|  0 | Default room | - 0      |
+----+--------------+----------+
```

This command lists the locations.

Note
The `children` column lists the ID of children, not their quantity.

vinfra location create

Create a new child location of the specified failure domain within the parent location identified by ID:

```
usage: vinfra location create --fd <fd> --name <location-name>
                              [--parent-id <parent-id>]
```

`--fd <fd>`

Failure domain ID

`--name <location-name>`

Name of the location to be created.

`--parent-id <parent-id>`

ID of the parent location where the child location should be created in.

Example:

```
# vinfra location create --fd 3 --name row2 --parent-id 0
+----------+-------+
| Field    | Value |
+----------+-------+
| children | []    |
| id       | 1     |
| name     | row2  |
| parent   | 0     |
+----------+-------+
```

This command creates the location `row2`.

Note

To create a location of level 4 (room), do not use the `--parent-id` argument.

vinfra location rename

Change the name of the location of the specified failure domain and identified by ID:

```
usage: vinfra location rename --fd <fd> --id <location-id>
                              --name <location-name>
```

`--fd <fd>`

Failure domain ID.

`--id <location-id>`

ID of the location to rename.

`--name <location-name>`

The new location name.

Example:

```
# vinfra location rename --fd 3 --id 2 --name row_renamed
Operation successful.
```

```
```

This command renames the location `2` to `row_renamed`.

Task outcome:

```
# vinfra location list --fd 3
+----+-------------+--------+----------+
| id | name        | parent | children |
+----+-------------+--------+----------+
|  0 | Default row |      0 | - 0      |
|  2 | row_renamed |      1 | []       |
+----+-------------+--------+----------+
```

vinfra location show

Show the location of the specified failure domain and identified by ID:

```
usage: vinfra location show --fd <fd> --id <location-id>
```

`--fd <fd>`

Failure domain ID.

`--id <location-id>`

ID of the location to show.

Example:

```
# vinfra location show --fd 3 --id 2
+----------+-------------+
| Field    | Value       |
+----------+-------------+
| children | []          |
| id       | 2           |
| name     | row_renamed |
| parent   | 1           |
+----------+-------------+
```

This command shows the location `2`.

vinfra location move

Move locations identified by IDs to the parent location of the specified failure domain and identified by ID:

```
usage: vinfra location move --children <children> [<children> …]
                            --parent-fd <parent-fd> --parent-id <parent-id>
```

`--children <children> [<children> …]`

IDs of locations to be moved to the parent location.

`--parent-fd <parent-fd>`

The failure domain of the parent location.

`--parent-id <parent-id>`

ID of the parent location.

Example:

```
# vinfra location move --children 2 --parent-fd 4 --parent-id 1
Operation successful.
```

This command moves the location (row) 2 to the failure domain 4 (room) with ID 1.

vinfra location delete

Delete the location of specified failure domain and identified by ID.

```
usage: vinfra location delete --fd <fd> --id <location-id>
```

`--fd <fd>`

Failure domain ID.

`--id <location-id>`

ID of the location to delete.

Example:

```
# vinfra location delete --fd 3 --id 1
Operation successful.
```

This command deletes the location 1.

Managing node network interfaces

vinfra node iface list

List node network interfaces:

```
usage: vinfra node iface list [--long] [-a | --node <node>]
```

`--long`

Enable access and listing of all fields of objects.

`-a, --all`

List all network interfaces on all nodes

`--node <node>`

Node ID or hostname to list network interfaces on (default: `node001.vstoragedomain`)

Example:

This command shows network interfaces of the node with the ID `4f96acf5-3bc8-4094-bcb6-4d1953be7b55`.

```
# vinfra node iface list --node 4f96acf5-3bc8-4094-bcb6-4d1953be7b55
+------+---------------+--------------------+-------+---------+
| name | node_id       | ipv4               | state | network |
+------+---------------+--------------------+-------+---------+
| eth0 | 4f96acf5-<...> | - 10.94.29.218/16  | up    | Public  |
| eth1 | 4f96acf5-<...> | - 10.37.130.101/24 | up    | Private |
+------+---------------+--------------------+-------+---------+
```

vinfra node iface show

Show details of a network interface:

```
usage: vinfra node iface show [--node <node>] <iface>
```

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<iface>`

Network interface name

Example:

```
# vinfra node iface show eth0 --node 4f96acf5-3bc8-4094-bcb6-4d1953be7b55
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| contained_in         |                                      |
| dhcp4                | 10.94.29.218                         |
| dhcp4_enabled        | True                                 |
| dhcp6                | fe80::21c:42ff:fe2a:4fdf             |
| dhcp6_enabled        | True                                 |
| dns4                 | - 127.0.0.1                          |
| dns6                 | []                                   |
| duplex               |                                      |
| gw4                  | 10.94.0.1                            |
| gw6                  |                                      |
| ignore_auto_dns_v4   | False                                |
| ignore_auto_dns_v6   | False                                |
| ignore_auto_routes_v4 | False                                |
| ignore_auto_routes_v6 | False                                |
| ipv4                 | - 10.94.29.218/16                    |
| ipv6                 | - fe80::21c:42ff:fe2a:4fdf/64        |
```

```
| mac_addr              | 00:1c:42:2a:4f:df                    |
| mtu                   | 1500                                 |
| multicast             | True                                 |
| name                  | eth0                                 |
| node_id               | 4f96acf5-3bc8-4094-bcb6-4d1953be7b55 |
| plugged               | True                                 |
| roles_set             | 237e58dd-6c10-49c1-be7f-7ddf7de2efd1 |
| rx_bytes              | 1844502614                           |
| rx_dropped            | 0                                    |
| rx_errors             | 0                                    |
| rx_overruns           | 0                                    |
| rx_packets            | 11543284                             |
| speeds                | current: null                        |
|                       | max: null                            |
| state                 | up                                   |
| tx_bytes              | 28477979                             |
| tx_dropped            | 0                                    |
| tx_errors             | 0                                    |
| tx_overruns           | 0                                    |
| tx_packets            | 107649                               |
| type                  | iface                                |
+-----------------------+--------------------------------------+
```

This command shows the details of the network interface `eth0` located on the node with the ID `4f96acf5-3bc8-4094-bcb6-4d1953be7b55`.

vinfra node iface up

Bring a network interface up:

```
usage: vinfra node iface up [--node <node>] <iface>
```

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<iface>`

Network interface name

Example:

```
# vinfra node iface up eth2 --node 4f96acf5-3bc8-4094-bcb6-4d1953be7b55
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| contained_in          |                                      |
| dhcp4                 | 10.37.130.138                        |
| dhcp4_enabled         | True                                 |
| dhcp6                 | fe80::21c:42ff:fef8:5b90             |
| dhcp6_enabled         | True                                 |
| dns4                  | - 127.0.0.1                          |
```

```
| dns6                  | []                                       |
| duplex                |                                          |
| gw4                   | 10.94.0.1                                |
| gw6                   |                                          |
| ignore_auto_dns_v4    | False                                    |
| ignore_auto_dns_v6    | False                                    |
| ignore_auto_routes_v4 | False                                    |
| ignore_auto_routes_v6 | False                                    |
| ipv4                  | - 10.37.130.138/24                       |
| ipv6                  | - fe80::21c:42ff:fef8:5b90/64            |
| mac_addr              | 00:1c:42:f8:5b:90                        |
| mtu                   | 1500                                     |
| multicast             | True                                     |
| name                  | eth2                                     |
| node_id               | 4f96acf5-3bc8-4094-bcb6-4d1953be7b55 |
| plugged               | True                                     |
| roles_set             |                                          |
| rx_bytes              | 97632                                    |
| rx_dropped            | 0                                        |
| rx_errors             | 0                                        |
| rx_overruns           | 0                                        |
| rx_packets            | 1258                                     |
| speeds                | current: null                            |
|                       | max: null                                |
| state                 | up                                       |
| tx_bytes              | 1116                                     |
| tx_dropped            | 0                                        |
| tx_errors             | 0                                        |
| tx_overruns           | 0                                        |
| tx_packets            | 8                                        |
| type                  | iface                                    |
+-----------------------+------------------------------------------+
```

This command brings up the network interface `eth2` located on the node with the ID `4f96acf5-3bc8-4094-bcb6-4d1953be7b55`.

vinfra node iface down

Bring a network interface down:

```
usage: vinfra node iface down [--node <node>] <iface>
```

`--node <node>`

Node ID or hostname

`<iface>`

Network interface name

Example:

```
# vinfra node iface down eth2 --node 4f96acf5-3bc8-4094-bcb6-4d1953be7b55
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| contained_in         |                                      |
| dhcp4                |                                      |
| dhcp4_enabled        | True                                 |
| dhcp6                |                                      |
| dhcp6_enabled        | True                                 |
| dns4                 | - 127.0.0.1                          |
| dns6                 | []                                   |
| duplex               |                                      |
| gw4                  | 10.94.0.1                            |
| gw6                  |                                      |
| ignore_auto_dns_v4   | False                                |
| ignore_auto_dns_v6   | False                                |
| ignore_auto_routes_v4 | False                                |
| ignore_auto_routes_v6 | False                                |
| ipv4                 | []                                   |
| ipv6                 | []                                   |
| mac_addr             | 00:1c:42:f8:5b:90                    |
| mtu                  | 1500                                 |
| multicast            | True                                 |
| name                 | eth2                                 |
| node_id              | 4f96acf5-3bc8-4094-bcb6-4d1953be7b55 |
| plugged              | False                                |
| roles_set            |                                      |
| rx_bytes             | 97984                                |
| rx_dropped           | 0                                    |
| rx_errors            | 0                                    |
| rx_overruns          | 0                                    |
| rx_packets           | 1264                                 |
| speeds               | current: null                        |
|                      | max: null                            |
| state                | down                                 |
| tx_bytes             | 1116                                 |
| tx_dropped           | 0                                    |
| tx_errors            | 0                                    |
| tx_overruns          | 0                                    |
| tx_packets           | 8                                    |
| type                 | iface                                |
+----------------------+--------------------------------------+
```

This command brings down the network interface `eth2` located on the node with the ID `4f96acf5-3bc8-4094-bcb6-4d1953be7b55`.

vinfra node iface set

Modify network interface parameters (overwrite the omitted options with the default values for the interface):


```
usage: vinfra node iface set [--ipv4 <ipv4>] [--ipv6 <ipv6>]
                             [--gw4 <gw4>] [--gw6 <gw6>] [--mtu <mtu>]
                             [--dhcp4 | --no-dhcp4] [--dhcp6 | --no-dhcp6]
                             [--auto-routes-v4 | --ignore-auto-routes-v4]
                             [--auto-routes-v6 | --ignore-auto-routes-v6]
                             [--network <network> | --no-network]
                             [--connected-mode | --datagram-mode]
                             [--ifaces <ifaces>] [--bond-type <bond-type>]
                             [--node <node>] <iface>
```

`--ipv4 <ipv4>`

A comma-separated list of IPv4 addresses

`--ipv6 <ipv6>`

A comma-separated list of IPv6 addresses

`--gw4 <gw4>`

Gateway IPv4 address

`--gw6 <gw6>`

Gateway IPv6 address

`--mtu <mtu>`

MTU interface value

`--dhcp4`

Enable DHCPv4

`--no-dhcp4`

Disable DHCPv4

`--dhcp6`

Enable DHCPv6

`--no-dhcp6`

Disable DHCPv6

`--auto-routes-v4`

Enable automatic IPv4 routes

`--ignore-auto-routes-v4`

Ignore automatic IPv4 routes

`--auto-routes-v6`

Enable automatic IPv6 routes

`--ignore-auto-routes-v6`

Ignore automatic IPv6 routes

`--network <network>`

Network ID or name

`--no-network`

Remove a network from the interface

`--connected-mode`

Enable connected mode (InfiniBand interfaces only)

`--datagram-mode`

Enable datagram mode (InfiniBand interfaces only)

`--ifaces <ifaces>`

A comma-separated list of network interface names, for example, `iface1,iface2,...,ifaceN`

`--bond-type <bond-type>`

Bond type (`balance-rr`, `balance-xor`, `broadcast`, `802.3ad`, `balance-tlb`, `balance-alb`)

Bond type for an OVS interface (`balance-tcp`, `active-backup`)

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<iface>`

Network interface name

Example 1. Assigning a network to a network interface:

```
# vinfra node iface set eth2 --network MyNet --node node002
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 8a378098-6760-4fe9-ac20-1f18a8ed9d2e |
+---------+--------------------------------------+
```

This command creates a task to assign the network `MyNet` to the network interface `eth2` located on the node `node002`.

Task outcome:

```
# vinfra task show 8a378098-6760-4fe9-ac20-1f18a8ed9d2e
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.network.tasks.NetworkInterfaceChangeTask |
| result  | contained_in: null                                               |
|         | dhcp4: null                                                      |
|         | dhcp4_enabled: false                                             |
|         | dhcp6: null                                                      |
|         | dhcp6_enabled: false                                             |
|         | duplex: null                                                     |
|         | gw4: null                                                        |
|         | gw6: null                                                        |
```



```
|         | ignore_auto_routes_v4: true                                    |
|         | ignore_auto_routes_v6: true                                    |
|         | ipv4: []                                                       |
|         | ipv6: []                                                       |
|         | mac_addr: fa:16:3e:a7:fa:bc                                    |
|         | mtu: 1450                                                      |
|         | multicast: true                                                |
|         | name: eth2                                                     |
|         | node_id: db83dc60-e34a-43d3-06fe-0caeb5ddaae2                  |
|         | plugged: true                                                  |
|         | roles_set: 7577efe8-33b5-44da-b54e-ab8f4419125b                |
|         | rx_bytes: 7038                                                 |
|         | rx_dropped: 0                                                  |
|         | rx_errors: 0                                                   |
|         | rx_overruns: 0                                                 |
|         | rx_packets: 90                                                 |
|         | speeds:                                                        |
|         |   current: null                                                |
|         |   max: null                                                    |
|         | state: up                                                      |
|         | tx_bytes: 356                                                  |
|         | tx_dropped: 0                                                  |
|         | tx_errors: 0                                                   |
|         | tx_overruns: 0                                                 |
|         | tx_packets: 4                                                  |
|         | type: iface                                                    |
| state   | success                                                        |
| task_id | 8a378098-6760-4fe9-ac20-1f18a8ed9d2e                           |
+---------+----------------------------------------------------------------+
```

Example 2. Unassigning a network from a network interface:

```
# vinfra node iface set eth2 --node node002 --no-network
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | c47837c4-e7a8-40d0-ab77-67c65375b86d |
+---------+--------------------------------------+
```

This command creates a task to unassign a network from the network interface `eth2` located on the node `node002`.

Task outcome:

```
# vinfra task show c47837c4-e7a8-40d0-ab77-67c65375b86d
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.network.tasks.NetworkInterfaceChangeTask |
| result  | contained_in: null                                               |
```



```
|         | dhcp4: null                                                      |
|         | dhcp4_enabled: false                                             |
|         | dhcp6: null                                                      |
|         | dhcp6_enabled: false                                             |
|         | duplex: null                                                     |
|         | gw4: null                                                        |
|         | gw6: null                                                        |
|         | ignore_auto_routes_v4: true                                      |
|         | ignore_auto_routes_v6: true                                      |
|         | ipv4: []                                                         |
|         | ipv6: []                                                         |
|         | mac_addr: fa:16:3e:a7:fa:bc                                      |
|         | mtu: 1450                                                        |
|         | multicast: true                                                  |
|         | name: eth2                                                       |
|         | node_id: db83dc60-e34a-43d3-06fe-0caeb5ddaae2                    |
|         | plugged: true                                                    |
|         | roles_set: ''                                                    |
|         | rx_bytes: 7038                                                   |
|         | rx_dropped: 0                                                    |
|         | rx_errors: 0                                                     |
|         | rx_overruns: 0                                                   |
|         | rx_packets: 90                                                   |
|         | speeds:                                                          |
|         |   current: null                                                  |
|         |   max: null                                                      |
|         | state: up                                                        |
|         | tx_bytes: 356                                                    |
|         | tx_dropped: 0                                                    |
|         | tx_errors: 0                                                     |
|         | tx_overruns: 0                                                   |
|         | tx_packets: 4                                                    |
|         | type: iface                                                      |
| state   | success                                                          |
| task_id | c47837c4-e7a8-40d0-ab77-67c65375b86d                             |
+---------+------------------------------------------------------------------+
```

Example 3. Enabling DHCP for a network interface:

```
# vinfra node iface set eth2 --node node002 --dhcp4
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 077ef0c2-de0b-4e6c-84d0-d7cafd390606 |
+---------+--------------------------------------+
```

This command creates a task to enable IP address allocation via DCHP for the network interface `eth2` located on the node `node002`.

Task outcome:


```
# vinfra task show 077ef0c2-de0b-4e6c-84d0-d7cafd390606
+---------+-------------------------------------------------------------+
| Field   | Value                                                       |
+---------+-------------------------------------------------------------+
| details |                                                             |
| name    | backend.presentation.network.tasks.NetworkInterfaceChangeTask |
| result  | contained_in: null                                          |
|         | dhcp4: 192.168.30.192/24                                    |
|         | dhcp4_enabled: true                                         |
|         | dhcp6: null                                                 |
|         | dhcp6_enabled: true                                         |
|         | duplex: null                                                |
|         | gw4: null                                                   |
|         | gw6: null                                                   |
|         | ignore_auto_routes_v4: true                                 |
|         | ignore_auto_routes_v6: false                                |
|         | ipv4:                                                       |
|         | - 192.168.30.192/24                                         |
|         | ipv6: []                                                    |
|         | mac_addr: fa:16:3e:a7:fa:bc                                 |
|         | mtu: 1450                                                   |
|         | multicast: true                                             |
|         | name: eth2                                                  |
|         | node_id: db83dc60-e34a-43d3-06fe-0caeb5ddaae2               |
|         | plugged: true                                               |
|         | roles_set: ''                                               |
|         | rx_bytes: 8080                                              |
|         | rx_dropped: 0                                               |
|         | rx_errors: 0                                                |
|         | rx_overruns: 0                                              |
|         | rx_packets: 93                                              |
|         | speeds:                                                     |
|         |   current: null                                             |
|         |   max: null                                                 |
|         | state: up                                                   |
|         | tx_bytes: 1570                                              |
|         | tx_dropped: 0                                               |
|         | tx_errors: 0                                                |
|         | tx_overruns: 0                                              |
|         | tx_packets: 13                                              |
|         | type: iface                                                 |
| state   | success                                                     |
| task_id | 077ef0c2-de0b-4e6c-84d0-d7cafd390606                        |
+---------+-------------------------------------------------------------+
```

Example 4. Disabling DHCP and manually setting the IP address for a network interface:

```
# vinfra node iface set eth2 --node node002 --no-dhcp4 --ipv4 192.168.30.20/24
+---------+-------------------------------------+
| Field   | Value                               |
+---------+-------------------------------------+
```

```
| task_id | 95ab841c-3ce8-4ada-ab61-60ddcfc90d79 |
+---------+--------------------------------------+
```

This command creates a task to disable DHCP and set the IP address 192.168.30.20/24 for the network interface `eth2` located on the node `node002`.

Task outcome:

```
# vinfra task show 95ab841c-3ce8-4ada-ab61-60ddcfc90d79
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.network.tasks.NetworkInterfaceChangeTask |
| result  | contained_in: null                                               |
|         | dhcp4: null                                                      |
|         | dhcp4_enabled: false                                             |
|         | dhcp6: null                                                      |
|         | dhcp6_enabled: false                                             |
|         | duplex: null                                                     |
|         | gw4: null                                                        |
|         | gw6: null                                                        |
|         | ignore_auto_routes_v4: true                                      |
|         | ignore_auto_routes_v6: true                                      |
|         | ipv4:                                                            |
|         | - 192.168.30.20/24                                               |
|         | ipv6: []                                                         |
|         | mac_addr: fa:16:3e:a7:fa:bc                                      |
|         | mtu: 1450                                                        |
|         | multicast: true                                                  |
|         | name: eth2                                                       |
|         | node_id: db83dc60-e34a-43d3-06fe-0caeb5ddaae2                    |
|         | plugged: true                                                    |
|         | roles_set: ''                                                    |
|         | rx_bytes: 8164                                                   |
|         | rx_dropped: 0                                                    |
|         | rx_errors: 0                                                     |
|         | rx_overruns: 0                                                   |
|         | rx_packets: 95                                                   |
|         | speeds:                                                          |
|         |   current: null                                                  |
|         |   max: null                                                      |
|         | state: up                                                        |
|         | tx_bytes: 1962                                                   |
|         | tx_dropped: 0                                                    |
|         | tx_errors: 0                                                     |
|         | tx_overruns: 0                                                   |
|         | tx_packets: 19                                                   |
|         | type: iface                                                      |
| state   | success                                                          |
```

```
| task_id | 95ab841c-3ce8-4ada-ab61-60ddcfc90d79                          |
+---------+-------------------------------------------------------------+
```

Example 5. Changing the bond type for a network bond:

```
# vinfra node iface set bond0 --node node002 --bond-type balance-xor
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 3a21b5b8-fe5e-432a-b143-43f80ec51b70 |
+---------+--------------------------------------+
```

This command creates a task to change the bond type of the network bond `bond0` located on the node `node002` to `balance-xor`.

Task outcome:

```
# vinfra task show 3a21b5b8-fe5e-432a-b143-43f80ec51b70
+---------+-------------------------------------------------------------------+
| Field   | Value                                                             |
+---------+-------------------------------------------------------------------+
| details |                                                                   |
| name    | backend.presentation.network.tasks.NetworkInterfaceChangeTask |
| result  | bond_type: balance-xor                                            |
|         | dhcp4: null                                                       |
|         | dhcp4_enabled: false                                              |
|         | dhcp6: null                                                       |
|         | dhcp6_enabled: false                                              |
|         | duplex: null                                                      |
|         | gw4: null                                                         |
|         | gw6: null                                                         |
|         | ifaces:                                                           |
|         | - eth2                                                            |
|         | - eth3                                                            |
|         | ignore_auto_routes_v4: true                                       |
|         | ignore_auto_routes_v6: true                                       |
|         | ipv4:                                                             |
|         | - 192.168.30.20/24                                                |
|         | ipv6: []                                                          |
|         | mac_addr: fa:16:3e:a7:fa:bc                                       |
|         | mtu: 1450                                                         |
|         | multicast: true                                                   |
|         | name: bond0                                                       |
|         | node_id: db83dc60-e34a-43d3-06fe-0caeb5ddaae2                     |
|         | plugged: true                                                     |
|         | roles_set: ''                                                     |
|         | rx_bytes: 1326                                                    |
|         | rx_dropped: 0                                                     |
|         | rx_errors: 0                                                      |
|         | rx_overruns: 0                                                    |
|         | rx_packets: 17                                                    |
```

```
|         | speeds:                                                          |
|         |   current: null                                                  |
|         |   max: null                                                      |
|         | state: up                                                        |
|         | tx_bytes: 1586                                                   |
|         | tx_dropped: 0                                                    |
|         | tx_errors: 0                                                     |
|         | tx_overruns: 0                                                   |
|         | tx_packets: 21                                                   |
|         | type: bonding                                                    |
| state   | success                                                          |
| task_id | 3a21b5b8-fe5e-432a-b143-43f80ec51b70                             |
+---------+------------------------------------------------------------------+
```

vinfra node iface create-bond

Create a network bonding:

```
usage: vinfra node iface create-bond [--ipv4 <ipv4>] [--ipv6 <ipv6>]
                                     [--gw4 <gw4>] [--gw6 <gw6>] [--mtu <mtu>]
                                     [--dhcp4 | --no-dhcp4]
                                     [--dhcp6 | --no-dhcp6]
                                     [--auto-routes-v4 | --ignore-auto-routes-v4]
                                     [--auto-routes-v6 | --ignore-auto-routes-v6]
                                     [--bonding-opts <bonding_opts>]
                                     [--network <network>] [--node <node>]
                                     --bond-type <bond-type> --ifaces <ifaces>
```

`--ipv4 <ipv4>`

A comma-separated list of IPv4 addresses

`--ipv6 <ipv6>`

A comma-separated list of IPv6 addresses

`--gw4 <gw4>`

Gateway IPv4 address

`--gw6 <gw6>`

Gateway IPv6 address

`--mtu <mtu>`

MTU interface value

`--dhcp4`

Enable DHCPv4

`--no-dhcp4`

Disable DHCPv4

`--dhcp6`

Enable DHCPv6

`--no-dhcp6`

Disable DHCPv6

`--auto-routes-v4`

Enable automatic IPv4 routes

`--ignore-auto-routes-v4`

Ignore automatic IPv4 routes

`--auto-routes-v6`

Enable automatic IPv6 routes

`--ignore-auto-routes-v6`

Ignore automatic IPv6 routes

`--network <network>`

Network ID or name

`--bonding-opts <bonding_opts>`

Additional bonding options

`--bond-type <bond-type>`

Bond type (`balance-rr`, `active-backup`, `balance-xor`, `broadcast`, `802.3ad`, `balance-tlb`, `balance-alb`)

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`--ifaces <ifaces>`

A comma-separated list of network interface names, for example, `iface1,iface2,…,iface<N>`

Example:

```
# vinfra node iface create-bond --ifaces eth2,eth3 --bond-type balance-xor \
--dhcp4 --node fd1e46de-6e17-4571-bf6b-1ac34ec1c225
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | becf96ad-9e39-4bec-b82c-4e1219a196de |
+---------+--------------------------------------+
```

This command creates a task to bond network interfaces `eth2` and `eth3` into `bond0` of the type `balance-xor` on the node with the ID `fd1e46de-6e17-4571-bf6b-1ac34ec1c225`.

Task outcome:

```
# vinfra task show becf96ad-9e39-4bec-b82c-4e1219a196de
+---------+------------------------------------------------------------------+
```

```
| Field   | Value                                                                |
+---------+----------------------------------------------------------------------+
| args    | - fd1e46de-6e17-4571-bf6b-1ac34ec1c225                               |
| kwargs  | bond_type: balance-xor                                               |
|         | ifaces:                                                              |
|         | - eth2                                                               |
|         | - eth3                                                               |
|         | registration_token: 3102ed1a                                         |
| name    | backend.presentation.network.tasks.NetworkInterfaceCreateBonding     |
| result  | bond_type: balance-xor                                               |
|         | dhcp4: 10.37.130.117                                                 |
|         | dhcp4_enabled: true                                                  |
|         | dhcp6: fe80::21c:42ff:fe81:27d0                                      |
|         | dhcp6_enabled: true                                                  |
|         | duplex: null                                                         |
|         | gw4: 10.94.0.1                                                       |
|         | gw6: null                                                            |
|         | ignore_auto_routes_v4: false                                         |
|         | ignore_auto_routes_v6: false                                         |
|         | ipv4:                                                                |
|         | - 10.37.130.117/24                                                   |
|         | ipv6:                                                                |
|         | - fe80::21c:42ff:fe81:27d0/64                                        |
|         | mac_addr: 00:1c:42:81:27:d0                                          |
|         | mtu: 1500                                                            |
|         | multicast: true                                                      |
|         | name: bond0                                                          |
|         | node_id: fd1e46de-6e17-4571-bf6b-1ac34ec1c225                        |
|         | plugged: true                                                        |
|         | roles_set: ''                                                        |
|         | rx_bytes: 3048                                                       |
|         | rx_dropped: 0                                                        |
|         | rx_errors: 0                                                         |
|         | rx_overruns: 0                                                       |
|         | rx_packets: 22                                                       |
|         | speeds:                                                              |
|         |   current: null                                                      |
|         |   max: null                                                          |
|         | state: up                                                            |
|         | tx_bytes: 1782                                                       |
|         | tx_dropped: 0                                                        |
|         | tx_errors: 0                                                         |
|         | tx_overruns: 0                                                       |
|         | tx_packets: 13                                                       |
|         | type: bonding                                                        |
| state   | success                                                              |
| task_id | becf96ad-9e39-4bec-b82c-4e1219a196de                                 |
+---------+----------------------------------------------------------------------+
```

vinfra node iface create-vlan

Create a VLAN:

```
usage: vinfra node iface create-vlan [--ipv4 <ipv4>] [--ipv6 <ipv6>]
                                     [--gw4 <gw4>] [--gw6 <gw6>] [--mtu <mtu>]
                                     [--dhcp4 | --no-dhcp4]
                                     [--dhcp6 | --no-dhcp6]
                                     [--auto-routes-v4 | --ignore-auto-routes-v4]
                                     [--auto-routes-v6 | --ignore-auto-routes-v6]
                                     [--network <network>] [--node <node>]
                                     --iface <iface> --tag <tag>
```

`--ipv4 <ipv4>`
: A comma-separated list of IPv4 addresses

`--ipv6 <ipv6>`
: A comma-separated list of IPv6 addresses

`--gw4 <gw4>`
: Gateway IPv4 address

`--gw6 <gw6>`
: Gateway IPv6 address

`--mtu <mtu>`
: MTU interface value

`--dhcp4`
: Enable DHCPv4

`--no-dhcp4`
: Disable DHCPv4

`--dhcp6`
: Enable DHCPv6

`--no-dhcp6`
: Disable DHCPv6

`--auto-routes-v4`
: Enable automatic IPv4 routes

`--ignore-auto-routes-v4`
: Ignore automatic IPv4 routes

`--auto-routes-v6`
: Enable automatic IPv6 routes

`--ignore-auto-routes-v6`
: Ignore automatic IPv6 routes

`--network <network>`
: Network ID or name

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`--iface <iface>`

Interface name

`--tag <tag>`

VLAN tag number

Example:

```
# vinfra node iface create-vlan --iface eth2 --tag 100 --dhcp4 \
--node fd1e46de-6e17-4571-bf6b-1ac34ec1c225
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 0b978acd-367b-47ad-8572-4f4e6ffb8877 |
+---------+--------------------------------------+
```

This command creates a task to create a VLAN with the tag `100` on the network interface `eth2` on the node with the ID `fd1e46de-6e17-4571-bf6b-1ac34ec1c225`.

Task outcome:

```
# vinfra task show 0b978acd-367b-47ad-8572-4f4e6ffb8877
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| args    | - fd1e46de-6e17-4571-bf6b-1ac34ec1c225                                 |
| kwargs  | iface: eth2                                                            |
|         | tag: 100                                                               |
| name    | backend.presentation.network.tasks.NetworkInterfaceCreateVlanTask |
| result  | built_on: eth2                                                         |
|         | dhcp4: null                                                            |
|         | dhcp4_enabled: false                                                   |
|         | dhcp6: null                                                            |
|         | dhcp6_enabled: false                                                   |
|         | duplex: null                                                           |
|         | gw4: null                                                              |
|         | gw6: null                                                              |
|         | ignore_auto_routes_v4: true                                            |
|         | ignore_auto_routes_v6: true                                            |
|         | ipv4: []                                                               |
|         | ipv6:                                                                  |
|         | - fe80::21c:42ff:fe81:27d0/64                                          |
|         | mac_addr: 00:1c:42:81:27:d0                                            |
|         | mtu: 1500                                                              |
|         | multicast: true                                                        |
|         | name: eth2.100                                                         |
|         | node_id: fd1e46de-6e17-4571-bf6b-1ac34ec1c225                          |
|         | plugged: true                                                          |
```

```
|         | roles_set: ''                                                                |
|         | rx_bytes: 0                                                                  |
|         | rx_dropped: 0                                                                |
|         | rx_errors: 0                                                                 |
|         | rx_overruns: 0                                                               |
|         | rx_packets: 0                                                                |
|         | speeds:                                                                      |
|         |   current: null                                                              |
|         |   max: null                                                                  |
|         | state: up                                                                    |
|         | tag: 100                                                                     |
|         | tx_bytes: 738                                                                |
|         | tx_dropped: 0                                                                |
|         | tx_errors: 0                                                                 |
|         | tx_overruns: 0                                                               |
|         | tx_packets: 7                                                                |
|         | type: vlan                                                                   |
| state   | success                                                                      |
| task_id | 0b978acd-367b-47ad-8572-4f4e6ffb8877                                         |
+---------+------------------------------------------------------------------------+
```

vinfra node iface delete

Delete a network interface:

```
usage: vinfra node iface delete [--node <node>] <iface>
```

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<iface>`

Network interface name

Example:

```
# vinfra node iface delete --node fd1e46de-6e17-4571-bf6b-1ac34ec1c225 eth2.100
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 16503616-6c1c-48f9-999a-9d87b617d9ee |
+---------+--------------------------------------+
```

This command creates a task to delete a VLAN interface `eth1.100` from the node with the ID `fd1e46de-6e17-4571-bf6b-1ac34ec1c225`.

Task outcome:

```
# vinfra task show 16503616-6c1c-48f9-999a-9d87b617d9ee
+---------+------------------------------------------------------------------+
```

```
| Field   | Value                                                          |
+---------+----------------------------------------------------------------+
| details |                                                                |
| name    | backend.presentation.network.tasks.NetworkInterfaceRemoveTask |
| result  |                                                                |
| state   | success                                                        |
| task_id | 16503616-6c1c-48f9-999a-9d87b617d9ee                           |
+---------+----------------------------------------------------------------+
```

Managing node disks

vinfra node disk list

List node disks:

```
usage: vinfra node disk list [--long] [-a | --node <node>]
```

`--long`

Enable access and listing of all fields of objects.

`-a, --all`

List disks on all nodes

`--node <node>`

Node ID or hostname to list disks on (default: `node001.vstoragedomain`)

Example:

```
# vinfra node disk list --node 94d58604-6f30-4339-8578-adb7903b7277 \
-c id -c node_id -c device -c used -c size -c role
+----------------+----------------+--------+--------+-----------+------------+
| id             | node_id        | device | used   | size      | role       |
+----------------+----------------+--------+--------+-----------+------------+
| E0B7CE6F-<...> | 94d58604-<...> | sda    | 5.5GiB | 239.1GiB  | mds-system |
| EAC7DF5D-<...> | 94d58604-<...> | sdb    | 2.1GiB | 1007.8GiB | cs         |
| 49D792CA-<...> | 94d58604-<...> | sdc    | 2.1GiB | 1007.8GiB | cs         |
+----------------+----------------+--------+--------+-----------+------------+
```

This command lists disks on the node with the ID `94d58604-6f30-4339-8578-adb7903b7277`.

vinfra node disk show

Show details of a disk:

```
usage: vinfra node disk show [--node <node>] <disk>
```

`--node <node>`

Node ID or hostname

`<disk>`

Disk ID or device name (default: `node001.vstoragedomain`)

Example:

```
# vinfra node disk show EAC7DF5D-9E60-4444-85F7-5CA5738399CC \
--node 94d58604-6f30-4339-8578-adb7903b7277
+--------------------+--------------------------------------+
| Field              | Value                                |
+--------------------+--------------------------------------+
| being_released     | False                                |
| device             | sdb                                  |
| disk_status        | ok                                   |
| encryption         |                                      |
| id                 | EAC7DF5D-9E60-4444-85F7-5CA5738399CC |
| is_blink_available | False                                |
| is_blinking        | False                                |
| latency            |                                      |
| lun_id             |                                      |
| model              | Vz_HARDDISK2                         |
| mountpoint         | /vstorage/33aac2d5                   |
| node_id            | 94d58604-6f30-4339-8578-adb7903b7277 |
| role               | cs                                   |
| rpm                |                                      |
| serial_number      | 45589b5823ce4c188b55                 |
| service_id         | 1026                                 |
| service_params     | journal_type: inner_cache            |
|                    | tier: 0                              |
| service_status     | ok                                   |
| slot               |                                      |
| smart_status       | not_supported                        |
| space              | full_size: 1099511627776             |
|                    | size: 1082101518336                  |
|                    | used: 2246164480                     |
| tasks              |                                      |
| temperature        | 0.0                                  |
| transport          |                                      |
| type               | hdd                                  |
+--------------------+--------------------------------------+
```

This command shows the details of the disk with the ID `EAC7DF5D-9E60-4444-85F7-5CA5738399CC` attached to the node with the ID `94d58604-6f30-4339-8578-adb7903b7277`.

vinfra node disk assign

Add multiple disks to the storage cluster:

```
usage: vinfra node disk assign --disk <disk>:<role>[:<key=value,…>]
                               [--node <node>]
```

`--disk <disk>:<role> [:<key=value,…>]`

Disk configuration in the format:

- `<disk>`: disk device ID or name
- `<role>`: disk role (`cs`, `mds`, `journal`, `mds-journal`, `mds-system`, `cs-system`, `system`)
- comma-separated `key=value` pairs with keys (optional):
  - `tier`: disk tier (0, 1, 2 or 3)
  - `journal-tier`: journal (cache) disk tier (0, 1, 2 or 3)
  - `journal-type`: journal (cache) disk type (`no_cache`, `inner_cache` or `external_cache`)
  - `journal-disk`: journal (cache) disk ID or device name
  - `bind-address`: bind IP address for the metadata service

Example: `sda:cs:tier=0,journal-type=inner_cache`.

This option can be used multiple times.

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

Example:

```
# vinfra node disk assign --disk sdc:cs --node f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 080337ba-0508-44a0-9363-eddcd9df9f0d |
+---------+--------------------------------------+
```

This command creates a task to assign the role `cs` to the disk `sdc` on the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`

Task outcome:

```
# vinfra task show 080337ba-0508-44a0-9363-eddcd9df9f0d
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| args    | []                                                               |
| kwargs  | cluster_id: 1                                                    |
|         | disks:                                                           |
|         | - id: D3BEF4BB-AA3B-4DB6-9376-BC7CDA636700                       |
|         |   role: cs                                                       |
|         |   service_params: {}                                             |
|         | logger:                                                          |
|         |   __classname: backend.logger.tracer.TracingLogger               |
|         |   __dict:                                                        |
|         |     prefix: POST /api/v2/1/nodes/f59dabdb-bd1c-<...>/disks/      |
|         |     token: '3215629651314950'                                    |
|         | node_id: f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4                    |
| name    | backend.tasks.disks.BulkAssignDiskTask                           |
```

```
| result  | {}                                                                |
| state   | success                                                           |
| task_id | 080337ba-0508-44a0-9363-eddcd9df9f0d                              |
+---------+-------------------------------------------------------------------+
```

vinfra node disk release

Release a disk from the storage cluster. Start data migration from the node as well as cluster replication and rebalancing to meet the configured redundancy level:

```
usage: vinfra node disk release [--force] [--node <node>] <disk>
```

`--force`

Release without data migration

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<disk>`

Disk ID or device name

Example:

```
# vinfra node disk release sdc --node f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 587a936d-3953-481c-a2cd-b1223b890bec |
+---------+--------------------------------------+
```

This command creates a task to release the role `cs` from the disk `sdc` on the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`.

Task outcome:

```
# vinfra task show 587a936d-3953-481c-a2cd-b1223b890bec
+---------+-------------------------------------------------------------------+
| Field   | Value                                                             |
+---------+-------------------------------------------------------------------+
| args    | []                                                                |
| kwargs  | cluster_id: 1                                                     |
|         | disk_id: 43EF3400-EA95-43DE-B624-3D7ED0F9DDDD                     |
|         | force: false                                                      |
|         | logger:                                                           |
|         |   __classname: backend.logger.tracer.TracingLogger                |
|         |   __dict:                                                         |
|         |     prefix: POST /api/v2/1/nodes/f59dabdb-                        |
|         | bd1c-4944-8af2-26b8fe9ff8d4/disks/43EF3400-EA95-<...>/release/ |
|         |     token: '3217122839314940'                                     |
```

```
|         | node_id: f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4                  |
| name    | backend.tasks.disks.ReleaseDiskTask                            |
| state   | success                                                        |
| task_id | 587a936d-3953-481c-a2cd-b1223b890bec                           |
+---------+----------------------------------------------------------------+
```

vinfra node disk blink on

Start blinking the specified disk bay to identify disk for maintenance purposes:

```
usage: vinfra node disk blink on [--node <node>] <disk>
```

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<disk>`

Disk ID or device name

Example:

```
# vinfra node disk blink on sda --node f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4
```

This command starts blinking the disk `sda` on the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`.

vinfra node disk blink off

Stop blinking the specified disk bay:

```
usage: vinfra node disk blink off [--node <node>] <disk>
```

`--node <node>`

Node ID or hostname (default: `node001.vstoragedomain`)

`<disk>`

Disk ID or device name

Example:

```
# vinfra node disk blink off sda --node f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4
```

This command stops blinking the disk `sda` on the node with the ID `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`.

vinfra node iscsi target add

Important
If you have restricted outbound traffic in your cluster, you need to manually add a rule that will allow outbound traffic on the port used by the added iSCSI device, as described in "Restricting outbound traffic from cluster nodes" (p. 369).

Add an iSCSI target as a disk to a node:

```
usage: vinfra node iscsi target add [--auth-username <auth-username>]
                                    [--auth-password <auth-password>]
                                    --portal <portal> --node <node> <target-name>
```

`--auth-username <auth-username>`
: User name

`--auth-password <auth-password>`
: User password

`--portal <portal>`
: Portal IP address in the format `IP:port` (this option can be specified multiple times)

`--node <node>`
: Node ID or hostname

`<target-name>`
: Target name

Example:

```
# vinfra node iscsi target add iqn.2014-06.com.vstorage:target1 \
--portal 172.16.24.244:3260 --node f1931be7-0a01-4977-bfef-51a392adcd94
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | c42bfbe5-7292-41c2-91cb-446795535ab9 |
+---------+--------------------------------------+
```

This command creates a task to connect a remote iSCSI target `iqn.2014-06.com.vstorage:target1` with the IP address `172.16.24.244` and port `3260` to the node with the ID `f1931be7-0a01-4977-bfef-51a392adcd94`.

Task outcome:

```
# vinfra task show c42bfbe5-7292-41c2-91cb-446795535ab9
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
```

```
+---------+-------------------------------------------------------------+
| args    | - f1931be7-0a01-4977-bfef-51a392adcd94                      |
| kwargs  | portals:                                                    |
|         | - address: 172.16.24.244                                    |
|         |   port: 3260                                                |
|         | target_name: iqn.2014-06.com.vstorage:target1               |
| name    | backend.presentation.nodes.iscsi_initiators.tasks.ConnectTask |
| result  | connected: true                                             |
|         | portals:                                                    |
|         | - address: 172.16.24.244                                    |
|         |   port: 3260                                                |
|         | state: connected                                            |
|         | target_name: iqn.2014-06.com.vstorage:target1               |
| state   | success                                                     |
| task_id | c42bfbe5-7292-41c2-91cb-446795535ab9                        |
+---------+-------------------------------------------------------------+
```

vinfra node iscsi target delete

Delete an iSCSI target from a node:

```
usage: vinfra node iscsi target delete --node <node> <target-name>
```

`--node <node>`

Node ID or hostname

`<target-name>`

Target name

Example:

```
# vinfra node iscsi target delete iqn.2014-06.com.vstorage:target1 \
--node f1931be7-0a01-4977-bfef-51a392adcd94
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | c8dc74ee-86d6-4b89-8b6f-153ff1e78cb7 |
+---------+--------------------------------------+
```

This command creates a task to disconnect a remote iSCSI target `iqn.2014-06.com.vstorage:target1` from the node with the ID `f1931be7-0a01-4977-bfef-51a392adcd94`.

Task outcome:

```
# vinfra task show c8dc74ee-86d6-4b89-8b6f-153ff1e78cb7
+---------+-------------------------------------------------------------------+
| Field   | Value                                                             |
+---------+-------------------------------------------------------------------+
| args    | - f1931be7-0a01-4977-bfef-51a392adcd94                            |
```

```
| kwargs  | target_name: iqn.2014-06.com.vstorage:target1                    |
| name    | backend.presentation.nodes.iscsi_initiators.tasks.DisconnectTask |
| state   | success                                                          |
| task_id | c8dc74ee-86d6-4b89-8b6f-153ff1e78cb7                             |
+---------+------------------------------------------------------------------+
```

Showing RAM reservation details

vinfra node ram-reservation list

Show RAM reservation details for all of the cluster nodes:

```
usage: vinfra node ram-reservation list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra node ram-reservation list -c host -c reservations -c total
+---------------+-------------------------------------------------------+-------+
| host          | reservations                                          | total |
+---------------+-------------------------------------------------------+-------+
| node001.<...> | - reserved_ram_mb: 3750                               | 17978 |
|               |   service_name: file_cache                           |       |
|               |   slice_name: ''                                     |       |
|               | - reserved_ram_mb: 1024                               |       |
|               |   service_name: fuse                                 |       |
|               |   slice_name: system.slice                           |       |
|               | - reserved_ram_mb: 1024                               |       |
|               |   service_name: management                           |       |
|               |   slice_name: system.slice                           |       |
|               | - reserved_ram_mb: 512                                |       |
|               |   service_name: user                                 |       |
|               |   slice_name: user.slice                             |       |
|               | - reserved_ram_mb: 7700                               |       |
|               |   service_name: compute                              |       |
|               |   slice_name: vstorage.slice/vstorage-compute.slice  |       |
|               | - reserved_ram_mb: 2048                               |       |
|               |   service_name: cses                                 |       |
|               |   slice_name: vstorage.slice/vstorage-services.slice |       |
|               | - reserved_ram_mb: 256                                |       |
|               |   service_name: mdses                                |       |
|               |   slice_name: vstorage.slice/vstorage-services.slice |       |
|               | - reserved_ram_mb: 128                                |       |
|               |   service_name: agent                                |       |
|               |   slice_name: vstorage.slice/vstorage-ui.slice       |       |
|               | - reserved_ram_mb: 1536                               |       |
|               |   service_name: management                           |       |
```

```
|               |   slice_name: vstorage.slice/vstorage-ui.slice       |       |
| <...>         |                                                      |       |
+---------------+------------------------------------------------------+-------+
```

This command lists all cluster nodes with their RAM reservation details.

vinfra node ram-reservation show

Show RAM reservation details for a cluster node:

```
usage: vinfra node ram-reservation show <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra node ram-reservation show ed39298c-dc1f-f057-0b78-bcf4281eda73
+--------------+-----------------------------------------------------+
| Field        | Value                                               |
+--------------+-----------------------------------------------------+
| host         | node001.vstoragedomain                              |
| id           | ed39298c-dc1f-f057-0b78-bcf4281eda73                |
| reservations | - reserved_ram_mb: 3749                             |
|              |   service_name: file_cache                          |
|              |   slice_name: ''                                    |
|              | - reserved_ram_mb: 1024                             |
|              |   service_name: fuse                                |
|              |   slice_name: system.slice                          |
|              | - reserved_ram_mb: 1024                             |
|              |   service_name: management                          |
|              |   slice_name: system.slice                          |
|              | - reserved_ram_mb: 512                              |
|              |   service_name: user                                |
|              |   slice_name: user.slice                            |
|              | - reserved_ram_mb: 7700                             |
|              |   service_name: compute                             |
|              |   slice_name: vstorage.slice/vstorage-compute.slice |
|              | - reserved_ram_mb: 2048                             |
|              |   service_name: cses                                |
|              |   slice_name: vstorage.slice/vstorage-services.slice|
|              | - reserved_ram_mb: 256                              |
|              |   service_name: mdses                               |
|              |   slice_name: vstorage.slice/vstorage-services.slice|
|              | - reserved_ram_mb: 128                              |
|              |   service_name: agent                               |
|              |   slice_name: vstorage.slice/vstorage-ui.slice      |
|              | - reserved_ram_mb: 1536                             |
|              |   service_name: management                          |
|              |   slice_name: vstorage.slice/vstorage-ui.slice      |
```

```
| total        | 17977                                                |
+--------------+------------------------------------------------------+
```

This command shows RAM reservation details for the node with the ID `ed39298c-dc1f-f057-0b78-bcf4281eda73`.

vinfra node ram-reservation total

Show total RAM reservation details:

```
usage: vinfra node ram-reservation total
```

Example:

```
# vinfra node ram-reservation total
+--------------+------------------------------------------------------+
| Field        | Value                                                |
+--------------+------------------------------------------------------+
| reservations | - reserved_ram_mb: 11868                             |
|              |   service_name: file_cache                           |
|              |   slice_name: ''                                     |
|              | - reserved_ram_mb: 4096                              |
|              |   service_name: fuse                                 |
|              |   slice_name: system.slice                           |
|              | - reserved_ram_mb: 1024                              |
|              |   service_name: management                           |
|              |   slice_name: system.slice                           |
|              | - reserved_ram_mb: 2048                              |
|              |   service_name: user                                 |
|              |   slice_name: user.slice                             |
|              | - reserved_ram_mb: 9760                              |
|              |   service_name: compute                              |
|              |   slice_name: vstorage.slice/vstorage-compute.slice  |
|              | - reserved_ram_mb: 7168                              |
|              |   service_name: cses                                 |
|              |   slice_name: vstorage.slice/vstorage-services.slice |
|              | - reserved_ram_mb: 1024                              |
|              |   service_name: mdses                                |
|              |   slice_name: vstorage.slice/vstorage-services.slice |
|              | - reserved_ram_mb: 512                               |
|              |   service_name: agent                                |
|              |   slice_name: vstorage.slice/vstorage-ui.slice       |
|              | - reserved_ram_mb: 1536                              |
|              |   service_name: management                           |
|              |   slice_name: vstorage.slice/vstorage-ui.slice       |
| total        | 39036                                                |
+--------------+------------------------------------------------------+
```

This command shows total RAM reservation details in the cluster.

Creating and deleting the storage cluster

vinfra cluster create

Create a storage cluster:

```
usage: vinfra cluster create [--disk <disk>:<role>[:<key=value,…>]]
                             [--tier-encryption {0,1,2,3}] --node <node>
                             <cluster-name>
```

`--disk <disk>:<role> [:<key=value,…>]`

Disk configuration in the format:

- `<disk>`: disk device ID or name
- `<role>`: disk role (`cs`, `mds`, `journal`, `mds-journal`, `mds-system`, `cs-system`, `system`)
- comma-separated `key=value` pairs with keys (optional):
  - `tier`: disk tier (0, 1, 2 or 3)
  - `journal-tier`: journal (cache) disk tier (0, 1, 2 or 3)
  - `journal-type`: journal (cache) disk type (`no_cache`, `inner_cache` or `external_cache`)
  - `journal-disk`: journal (cache) disk ID or device name
  - `bind-address`: bind IP address for the metadata service

Example: `sda:cs:tier=0,journal-type=inner_cache`.
This option can be used multiple times.

`--tier-encryption {0,1,2,3}`

Enable encryption for storage cluster tiers. Encryption is disabled by default. This option can be used multiple times.

`--node <node>`

Node ID or hostname

`<cluster-name>`

Storage cluster name

Example:

```
# vinfra cluster create stor1 --node 94d58604-6f30-4339-8578-adb7903b7277
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | d9ca8e1d-8ac8-4459-898b-2d803efd7bc6 |
+---------+--------------------------------------+
```

This command creates a task to create the storage cluster `stor1` on the node with the ID `94d58604-6f30-4339-8578-adb7903b7277`. As disk roles are not explicitly specified, they are assigned automatically: `mds-system` to the system disk, and `cs` to all other disks.

Task outcome:

```
# vinfra task show d9ca8e1d-8ac8-4459-898b-2d803efd7bc6
+---------+----------------------------------------+
| Field   | Value                                  |
+---------+----------------------------------------+
| args    | - stor1                                |
|         | - 94d58604-6f30-4339-8578-adb7903b7277 |
|         | - null                                 |
|         | - null                                 |
| kwargs  | {}                                     |
| name    | backend.tasks.cluster.CreateNewCluster |
| result  | cluster_id: 1                          |
| state   | success                                |
| task_id | d9ca8e1d-8ac8-4459-898b-2d803efd7bc6   |
+---------+----------------------------------------+
```

vinfra cluster delete

Delete the storage cluster:

```
usage: vinfra cluster delete
```

Example:

```
# vinfra cluster delete
Operation waiting (timeout=600s) [Elapsed Time: 0:01:09] ... |
Operation successful
```

This command releases all nodes from the storage cluster.

Showing storage cluster overview and details

vinfra cluster overview

Show storage cluster overview:

```
usage: vinfra cluster overview
```

Example:

```
# vinfra cluster overview
+------------------+-------------------------+
```

```
| Field             | Value                     |
+-------------------+---------------------------+
| active_cses       | '0': 5                    |
| active_nodes      | '0': 5                    |
| chunks            | blocked: 0                |
|                   | degraded: 0               |
|                   | deleting: 0               |
|                   | healthy: 2                |
|                   | offline: 0                |
|                   | overcommitted: 0          |
|                   | pending: 0                |
|                   | replicating: 0            |
|                   | standby: 0                |
|                   | total: 2                  |
|                   | unique: 2                 |
|                   | urgent: 0                 |
|                   | void: 0                   |
| cs                | failed: 0                 |
|                   | total: 5                  |
| fs_stat           | chunk_maps: 2             |
|                   | chunk_nodes: 2            |
|                   | file_maps: 2              |
|                   | files: 9                  |
|                   | inodes: 9                 |
|                   | used_size: 11335680       |
| id                | 1                         |
| license           | capacity: 1099511627776   |
|                   | expiration_ts: null       |
|                   | keynumber: null           |
|                   | status: 0                 |
|                   | used_size: 11335680       |
| logic_space       | free: 1099500292096       |
|                   | total: 1099511627776      |
|                   | used: 11335680            |
| mds               | failed: 0                 |
|                   | total: 5                  |
| name              | cluster1                  |
| repl              | eta: null                 |
|                   | reads: 0                  |
|                   | writes: 0                 |
| resistance        | to_lose: 0                |
|                   | total: 1                  |
| space_per_service | abgw: null                |
|                   | compute: null             |
|                   | iscsi: null               |
|                   | nfs: null                 |
|                   | other: 11335680           |
|                   | s3: null                  |
| status            | healthy                   |
| tiers             | - id: 0                   |
|                   |   phys_space:             |
|                   |     free: 2164191700992   |
```

```
|                   |     total: 2164203036672 |
|                   |     used: 11335680       |
+-------------------+--------------------------+
```

This command shows an overview of the cluster.

vinfra cluster show

Show cluster details:

```
usage: vinfra cluster show
```

Example:

```
# vinfra cluster show
+-------+-------------------------------------------+
| Field | Value                                     |
+-------+-------------------------------------------+
| id    | 1                                         |
| name  | stor1                                     |
| nodes | - host: node004.vstoragedomain            |
|       |   id: 4b83a87d-9adf-472c-91f0-782c47b2d5f1 |
|       |   is_installing: false                    |
|       |   is_releasing: false                     |
|       | - host: node003.vstoragedomain            |
|       |   id: 7d7d37b8-4c06-4f1a-b3a6-4b54257d70ce |
|       |   is_installing: false                    |
|       |   is_releasing: false                     |
|       | - host: node005.vstoragedomain            |
|       |   id: fd1e46de-6e17-4571-bf6b-1ac34ec1c225 |
|       |   is_installing: false                    |
|       |   is_releasing: false                     |
|       | - host: node001.vstoragedomain            |
|       |   id: 94d58604-6f30-4339-8578-adb7903b7277 |
|       |   is_installing: false                    |
|       |   is_releasing: false                     |
|       | - host: node001.vstoragedomain            |
|       |   id: f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4 |
|       |   is_installing: false                    |
|       |   is_releasing: false                     |
+-------+-------------------------------------------+
```

This command shows cluster details.

Managing the compute cluster

Creating and deleting the compute cluster

vinfra service compute create

Create a compute cluster:

```
usage: vinfra service compute create [--public-network <network>]
                                     [--subnet cidr=CIDR[,key=value,…]]
                                     [--cpu-model <cpu-model>] [--force]
                                     [--enable-k8saas] [--enable-lbaas]
                                     [--enable-metering] --nodes <nodes>
                                     [--notification-forwarding <transport-url>]
                                     [--disable-notification-forwarding]
                                     [--endpoint-hostname <hostname>]
                                     [--vlan-id <vlan-id>]
                                     [--pci-passthrough-config <path>]
                                     [--custom-param <service_name> <config_file>
                                     <section> <property> <value>]
                                     [--nova-scheduler-ram-weight-multiplier <value>]
                                     [--nova-compute-ram-allocation-ratio <value>]
                                     [--neutron-openvswitch-vxlan-port <value>]
                                     [--nova-scheduler-host-subset-size <value>]
                                     [--nova-compute-cpu-allocation-ratio <value>]
```

`--public-network <network>`

An infrastructure network to connect the compute physical network to. It must include the 'VM public' traffic type.

`--subnet cidr=CIDR[,key=value,…]`

Subnet for IP address management in the compute physical network (the `--public-network` option is required):

- `cidr`: subnet range in CIDR notation;
- comma-separated `key=value` pairs with keys (optional):
  - `gateway`: gateway IP address.
  - `dhcp`: enable/disable the virtual DHCP server.
  - `allocation-pool`: allocation pool of IP addresses from CIDR in the format `ip1-ip2`, where `ip1` and `ip2` are starting and ending IP addresses. Specify the key multiple times to create multiple IP pools.
  - `dns-server`: DNS server IP address, specify multiple times to set multiple DNS servers.

Example: `--subnet cidr=192.168.5.0/24,dhcp=enable`.

`--cpu-model <cpu-model>`

CPU model for virtual machines. View the list of available CPU models by using `vinfra service compute show`.

`--force`

Skip checks for minimal hardware requirements.

`--enable-k8saas`

Enable Kubernetes-as-a-Service services.

`--enable-lbaas`

Enable Load-Balancing-as-a-Service services.

`--enable-metering`

Enable metering services.

`--notification-forwarding <transport-url>`

Enable notification forwarding through the specified transport URL in the format `driver://[user:pass@]host:port[,[userN:passN@]hostN:portN]?query`, where

- `driver` is the supported transport driver (`kafka`)
- `user:pass` are the username and password used for authentication with the messaging broker
- `host:port` specifies the hostname or IP address and port number of the messaging broker
- `query` are parameters that override those from the broker configuration file:
  - `topic` specifies the topic name
  - `driver` is the messaging driver: `messaging`, `messagingv2`, `routing`, `log`, `test`, `noop`

Example: `kafka://10.10.10.10:9092?topic=notifications`

`--disable-notification-forwarding`

Disable notification forwarding

`--endpoint-hostname <hostname>`

Use the given hostname for a public endpoint. Specify an empty value in quotes to use the raw IP.

`--vlan-id <vlan-id>`

Create VLAN-based physical network by the given VLAN ID.

`--pci-passthrough-config <path>`

Path to the PCI passthrough configuration file (refer to "Configuring the compute cluster for PCI passthrough" (p. 377))

`--custom-param <service_name> <config_file> <section> <property> <value>`

Set custom parameters for OpenStack configuration files:

- `service_name` is the service name: `nova-scheduler`, `nova-compute`, or `neutron-openvswitch-agent`

- `config_file` specifies the service configuration file: `nova.conf` for `nova-scheduler` and `nova-compute`, or `ml2_conf.ini` for `neutron-openvswitch-agent`
- `section` specifies the section in the service configuration file where the parameter is defined: `DEFAULT` in `nova.conf` or `agent` in `ml2_conf.ini`
- `property` is the parameter to be changed: `ram_weight_multiplier`, `ram_allocation_ratio`, `scheduler_host_subset`, and `cpu_allocation_ratio` in `nova.conf`; `vxlan_udp_port` in `ml2_conf.ini`
- `value` is a new parameter value

`--nova-scheduler-ram-weight-multiplier <value>`

Shortcut for `--custom-param nova-scheduler nova.conf DEFAULT ram_weight_multiplier <value>`

`--nova-compute-ram-allocation-ratio <value>`

Shortcut for `--custom-param nova-compute nova.conf DEFAULT ram_allocation_ratio <value>`

`--neutron-openvswitch-vxlan-port <value>`

Shortcut for `--custom-param neutron-openvswitch-agent ml2_conf.ini agent vxlan_udp_port <value>`

`--nova-scheduler-host-subset-size <value>`

Shortcut for `--custom-param nova-scheduler nova.conf DEFAULT scheduler_host_subset_size <value>`

`--nova-compute-cpu-allocation-ratio <value>`

Shortcut for `--custom-param nova-scheduler nova.conf DEFAULT cpu_allocation_ratio <value>`

`--nodes <nodes>`

A comma-separated list of node IDs or hostnames.

Example:

```
# vinfra service compute create --nodes 7ffa9540-5a20-41d1-b203-e3f349d62565,\
02ff64ae-5800-4090-b958-18b1fe8f5060,6e8afc28-7f71-4848-bdbe-7c5de64c5013,\
37c70bfb-c289-4794-8be4-b7a40c2b6d95,827a1f4e-56e5-404f-9113-88748c18f0c2 \
--public-network Public --subnet cidr=10.94.0.0/16,dhcp=enable,\
gateway=10.94.0.1,allocation-pool=10.94.129.64-10.94.129.79,\
dns-server=10.30.0.27,dns-server=10.30.0.28
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | be517afa-fae0-457e-819c-f4d6399f3ae2 |
+---------+--------------------------------------+
```

This command creates a task to create the compute cluster from five nodes specified by ID. It also specifies the physical network for VMs, the gateway, the allocation pool of IP addresses to assign to VMs, and the DNS servers to use.

Task outcome:


```
# vinfra task show be517afa-fae0-457e-819c-f4d6399f3ae2
+----------+------------------------------------------------------------+
| Field    | Value                                                      |
+----------+------------------------------------------------------------+
| details  |                                                            |
| name     | backend.presentation.compute.tasks.DeployComputeClusterTask |
| progress | 100                                                        |
| result   |                                                            |
| state    | success                                                    |
| task_id  | be517afa-fae0-457e-819c-f4d6399f3ae2                       |
+----------+------------------------------------------------------------+
```

vinfra service compute delete

Delete all nodes from the compute cluster:

```
usage: vinfra service compute delete
```

Example:

```
# vinfra service compute delete
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 063e8a15-fcfe-4629-865f-b5e5fa44b38f |
+---------+--------------------------------------+
```

This command creates a task to release nodes from the compute cluster.

Task outcome:

```
# vinfra task show 063e8a15-fcfe-4629-865f-b5e5fa44b38f
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.presentation.compute.tasks.DestroyComputeClusterTask |
| result  |                                                            |
| state   | success                                                    |
| task_id | 063e8a15-fcfe-4629-865f-b5e5fa44b38f                       |
+---------+------------------------------------------------------------+
```

Showing compute cluster details and overview

vinfra service compute show

Display compute cluster details:

```
usage: vinfra service compute show
```

Example:

```
# vinfra service compute show
+--------------+-------------------------------------+
| Field        | Value                               |
+--------------+-------------------------------------+
| capabilities | cpu_models:                         |
|              | - EPYC-IBPB                         |
|              | - Nehalem                           |
|              | - Nehalem-IBRS                      |
|              | - SandyBridge                       |
|              | - SandyBridge-IBRS                  |
|              | - IvyBridge                         |
|              | - IvyBridge-IBRS                    |
|              | - Haswell                           |
|              | - Haswell-IBRS                      |
|              | - Haswell-noTSX                     |
|              | - Haswell-noTSX-IBRS                |
|              | - Broadwell                         |
|              | - Broadwell-IBRS                    |
|              | - Broadwell-noTSX                   |
|              | - Broadwell-noTSX-IBRS              |
|              | - Skylake-Client                    |
|              | - Skylake-Client-IBRS               |
|              | - Skylake-Server                    |
|              | - Skylake-Server-IBRS               |
|              | - HostPassthrough                   |
|              | os_distributions:                   |
|              | - id: linux                         |
|              |   os_type: linux                    |
|              |   title: Generic Linux              |
|              | - id: centos8                       |
|              |   os_type: linux                    |
|              |   title: CentOS 8                   |
|              | - id: centos7                       |
|              |   os_type: linux                    |
|              |   title: CentOS 7                   |
|              | - id: centos6                       |
|              |   os_type: linux                    |
|              |   title: CentOS 6                   |
|              | - id: rhel8                         |
|              |   os_type: linux                    |
|              |   title: Red Hat Enterprise Linux 8 |
|              | - id: rhel7                         |
|              |   os_type: linux                    |
|              |   title: Red Hat Enterprise Linux 7 |
|              | - id: ubuntu20.04                   |
|              |   os_type: linux                    |
|              |   title: Ubuntu 20.04               |
```

```
|              | - id: ubuntu18.04                   |
|              |   os_type: linux                    |
|              |   title: Ubuntu 18.04               |
|              | - id: ubuntu16.04                   |
|              |   os_type: linux                    |
|              |   title: Ubuntu 16.04               |
|              | - id: debian10                      |
|              |   os_type: linux                    |
|              |   title: Debian 10                  |
|              | - id: debian9                       |
|              |   os_type: linux                    |
|              |   title: Debian 9                   |
|              | - id: windows                       |
|              |   os_type: windows                  |
|              |   title: Generic Windows            |
|              | - id: win2k19                       |
|              |   os_type: windows                  |
|              |   title: Windows Server 2019        |
|              | - id: win2k16                       |
|              |   os_type: windows                  |
|              |   title: Windows Server 2016        |
|              | - id: win2k12r2                     |
|              |   os_type: windows                  |
|              |   title: Windows Server 2012 R2     |
|              | - id: win2k12                       |
|              |   os_type: windows                  |
|              |   title: Windows Server 2012        |
|              | - id: win2k8r2                      |
|              |   os_type: windows                  |
|              |   title: Windows Server 2008 R2     |
|              | - id: win2k8                        |
|              |   os_type: windows                  |
|              |   title: Windows Server 2008        |
|              | - id: win10                         |
|              |   os_type: windows                  |
|              |   title: Windows 10                 |
|              | - id: win8.1                        |
|              |   os_type: windows                  |
|              |   title: Windows 8.1                |
|              | - id: win7                          |
|              |   os_type: windows                  |
|              |   title: Windows 7                  |
| options      | cpu_model: ''                       |
|              | custom_params: []                   |
|              | notification_forwarding: disabled   |
| status       | active                              |
+--------------+-------------------------------------+
```

This command shows the status and capabilities of the compute cluster.

vinfra service compute stat

Display compute cluster statistics:

```
usage: vinfra service compute stat
```

Example:

```
# vinfra service compute stat
+----------+-----------------------------------------------+
| Field    | Value                                         |
+----------+-----------------------------------------------+
| compute  | block_capacity: 1073741824                    |
|          | block_usage: 268435456                        |
|          | cpu_allocation_ratio: 8                       |
|          | cpu_usage: 0.09                               |
|          | mem_total: 536870912                          |
|          | mem_usage: 176398336                          |
|          | vcpus: 1                                      |
|          | vcpus_free: 47                                |
| datetime | 2020-05-01T16:16:08.120482                    |
| fenced   | compute_mem_total: 0                          |
|          | physical_cpu_cores: 0                         |
|          | physical_cpu_usage: 0                         |
|          | physical_mem_total: 0                         |
|          | reserved_memory: 0                            |
|          | vcpus: 0                                      |
| physical | block_capacity: 1099511627776                 |
|          | block_free: 1099213661363                     |
|          | cpu_cores: 12                                 |
|          | cpu_usage: 8.31                               |
|          | mem_total: 49967353856                        |
|          | vcpus_total: 96                               |
| reserved | cpus: 6                                       |
|          | memory: 26135298048                           |
|          | vcpus: 48                                     |
| servers  | count: 1                                      |
|          | error: 0                                      |
|          | in_progress: 0                                |
|          | running: 1                                    |
|          | stopped: 0                                    |
|          | top:                                          |
|          |   disk:                                       |
|          |   - id: 32b0f95d-477f-46b5-86d6-e150360ea673 |
|          |     name: vm1                                 |
|          |     size: 268435456                           |
|          |   memory:                                     |
|          |   - id: 32b0f95d-477f-46b5-86d6-e150360ea673 |
|          |     name: vm1                                 |
|          |     size: 176398336                           |
```

```
|          |   vcpus:                                      |
|          |   - count: 0.01                               |
|          |     id: 32b0f95d-477f-46b5-86d6-e150360ea673 |
|          |     name: vm1                                 |
+----------+-----------------------------------------------+
```

This command shows the overview of the compute cluster.

Changing compute cluster parameters

Change compute cluster parameters:

```
usage: vinfra service compute set [--cpu-model <cpu-model>] [--enable-k8saas]
                                  [--enable-lbaas] [--enable-metering]
                                  [--notification-forwarding <transport-url>]
                                  [--disable-notification-forwarding]
                                  [--endpoint-hostname <hostname>] [--force]
                                  [--pci-passthrough-config <path>]
                                  [--custom-param <service_name> <config_file>
                                  <section> <property> <value>]
                                  [--nova-scheduler-ram-weight-multiplier <value>]
                                  [--nova-compute-ram-allocation-ratio <value>]
                                  [--neutron-openvswitch-vxlan-port <value>]
                                  [--nova-scheduler-host-subset-size <value>]
                                  [--nova-compute-cpu-allocation-ratio <value>]
```

`--cpu-model <cpu-model>`

Set the default CPU model for virtual machines. View the list of available CPU models by using "vinfra service compute show" (p. 85).

`--enable-k8saas`

Enable Kubernetes-as-a-Service services.

`--enable-lbaas`

Enable Load-Balancing-as-a-Service services.

`--enable-metering`

Enable metering services.

`--notification-forwarding <transport-url>`

Enable notification forwarding through the specified transport URL in the format `driver://[user:pass@]host:port[,[userN:passN@]hostN:portN]?query`, where

- `driver` is the supported transport driver (`kafka`)
- `user:pass` are the username and password used for authentication with the messaging broker
- `host:port` specifies the hostname or IP address and port number of the messaging broker

- `query` are parameters that override those from the broker configuration file:
  - `topic` specifies the topic name
  - `driver` is the messaging driver: `messaging`, `messagingv2`, `routing`, `log`, `test`, `noop`

  Example: `kafka://10.10.10.10:9092?topic=notifications`

`--disable-notification-forwarding`

Disable notification forwarding

`--endpoint-hostname <hostname>`

Use the given hostname for a public endpoint. Specify an empty value in quotes to use the raw IP.

`--force`

Skip checks for minimal hardware requirements.

`--pci-passthrough-config <path>`

Path to the PCI passthrough configuration file (refer to "Configuring the compute cluster for PCI passthrough" (p. 377))

`--custom-param <service_name> <config_file> <section> <property> <value>`

Set custom parameters for OpenStack configuration files:

- `service_name` is the service name: `nova-scheduler`, `nova-compute`, or `neutron-openvswitch-agent`
- `config_file` specifies the service configuration file: `nova.conf` for `nova-scheduler` and `nova-compute`, or `ml2_conf.ini` for `neutron-openvswitch-agent`
- `section` specifies the section in the service configuration file where the parameter is defined: `DEFAULT` in `nova.conf` or `agent` in `ml2_conf.ini`
- `property` is the parameter to be changed: `ram_weight_multiplier`, `ram_allocation_ratio`, `scheduler_host_subset`, and `cpu_allocation_ratio` in `nova.conf`; `vxlan_udp_port` in `ml2_conf.ini`
- `value` is a new parameter value

`--nova-scheduler-ram-weight-multiplier <value>`

Shortcut for `--custom-param nova-scheduler nova.conf DEFAULT ram_weight_multiplier <value>`

`--nova-compute-ram-allocation-ratio <value>`

Shortcut for `--custom-param nova-compute nova.conf DEFAULT ram_allocation_ratio <value>`

`--neutron-openvswitch-vxlan-port <value>`

Shortcut for `--custom-param neutron-openvswitch-agent ml2_conf.ini agent vxlan_udp_port <value>`

`--nova-scheduler-host-subset-size <value>`

Shortcut for `--custom-param nova-scheduler nova.conf DEFAULT scheduler_host_subset_size <value>`

`--nova-compute-cpu-allocation-ratio <value>`

Shortcut for `--custom-param nova-scheduler nova.conf DEFAULT cpu_allocation_ratio <value>`

Example:

```
# vinfra service compute set --cpu-model Haswell --nova-scheduler-cpu-allocation-ratio 3
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | be02e41d-18a5-44ee-8c76-333ebd92bc0d |
+---------+--------------------------------------+
```

This command creates a task to change the default CPU model for VMs to `Haswell` and the `cpu_allocation_ratio` parameter in `/etc/kolla/nova-scheduler/nova.conf` to 3.

Task outcome:

```
# vinfra task show be02e41d-18a5-44ee-8c76-333ebd92bc0d
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.compute.tasks.ReconfigureComputeClusterTask |
| result  |                                                                  |
| state   | success                                                          |
| task_id | be02e41d-18a5-44ee-8c76-333ebd92bc0d                             |
+---------+------------------------------------------------------------------+
```

Managing compute nodes

vinfra service compute node add

Add a node to the compute cluster:

```
usage: vinfra service compute node add [--compute] [--controller] [--force] <node>
```

`--compute`

Compute node role

`--controller`

Compute controller node role

`--force`

Skip checks for minimal hardware requirements

<node>

Node ID or hostname

Example:

```
# vinfra service compute node add 827a1f4e-56e5-404f-9113-88748c18f0c2 --compute
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 4c58e63c-31b6-406a-8070-9197445ec794 |
+---------+--------------------------------------+
```

This command creates a task to add the node with the ID `827a1f4e-56e5-404f-9113-88748c18f0c2` to the compute cluster with the compute role.

Task outcome:

```
# vinfra task show 4c58e63c-31b6-406a-8070-9197445ec794
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.presentation.compute.tasks.AddComputeNodesTask |
| result  |                                                            |
| state   | success                                                    |
| task_id | 4c58e63c-31b6-406a-8070-9197445ec794                       |
+---------+------------------------------------------------------------+
```

vinfra service compute node list

List compute nodes:

```
usage: vinfra service compute node list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute node list
+--------------------------------------+-----------------------+-------+-----+
| id                                   | host                  | state | vms |
+--------------------------------------+-----------------------+-------+-----+
| 7ffa9540-5a20-41d1-b203-e3f349d62565 | node001.vstoragedomain | up    |   1 |
| 6e8afc28-7f71-4848-bdbe-7c5de64c5013 | node002.vstoragedomain | up    |   1 |
| 02ff64ae-5800-4090-b958-18b1fe8f5060 | node003.vstoragedomain | up    |   1 |
| 827a1f4e-56e5-404f-9113-88748c18f0c2 | node004.vstoragedomain | up    |   0 |
| 37c70bfb-c289-4794-8be4-b7a40c2b6d95 | node005.vstoragedomain | up    |   1 |
+--------------------------------------+-----------------------+-------+-----+
```

This command lists nodes in the compute cluster.

vinfra service compute node show

Display compute node details:

```
usage: vinfra service compute node show <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra service compute node show 7ffa9540-5a20-41d1-b203-e3f349d62565
+---------------+-----------------------------------------+
| Field         | Value                                   |
+---------------+-----------------------------------------+
| host          | node001.vstoragedomain                  |
| host_ip       | 10.37.130.101                           |
| hypervisor    | id: 86f1ca2c-71c7-47a0-9c7f-bb9dd705e67e |
|               | state: up                               |
|               | status: enabled                         |
|               | vms: 0                                  |
| id            | 7ffa9540-5a20-41d1-b203-e3f349d62565    |
| orig_hostname | node001                                 |
| placements    | []                                      |
| roles         | - controller                            |
|               | - compute                               |
| services      | - name: cinder-scheduler                |
|               |   state: healthy                        |
|               | - name: cinder-volume                   |
|               |   state: healthy                        |
|               | - name: neutron-dhcp-agent              |
|               |   state: healthy                        |
|               | - name: neutron-l3-agent                |
|               |   state: healthy                        |
|               | - name: neutron-metadata-agent          |
|               |   state: healthy                        |
|               | - name: neutron-openvswitch-agent       |
|               |   state: healthy                        |
|               | - name: nova-compute                    |
|               |   state: healthy                        |
|               | - name: nova-conductor                  |
|               |   state: healthy                        |
|               | - name: nova-scheduler                  |
|               |   state: healthy                        |
| state         | healthy                                 |
+---------------+-----------------------------------------+
```

This command shows the details of the compute node with the ID `7ffa9540-5a20-41d1-b203-e3f349d62565`.

vinfra service compute node fence

Fence a compute node:

```
usage: vinfra service compute node fence <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra service compute node fence e6255aed-d6e7-41b2-ba90-86164c1cd9a6
Operation successful
```

This command fences the node with the ID `e6255aed-d6e7-41b2-ba90-86164c1cd9a6`.

vinfra service compute node unfence

Unfence a compute node:

```
usage: vinfra service compute node unfence <node>
```

<node>

Node ID or hostname

Example:

```
# vinfra service compute node unfence e6255aed-d6e7-41b2-ba90-86164c1cd9a6
Operation successful
```

This command unfences the node with the ID `e6255aed-d6e7-41b2-ba90-86164c1cd9a6`.

vinfra service compute node release

Release a node from the compute cluster:

```
usage: vinfra service compute node release [--compute] [--controller] <node>
```

`--compute`

Compute node role

`--controller`

Compute controller node role

<node>

Node ID or hostname

Example:

```
# vinfra service compute node release 827a1f4e-56e5-404f-9113-88748c18f0c2
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
```

```
| task_id | 3b39738c-80a6-40a6-a50d-c3c8118ed212 |
+---------+--------------------------------------+
```

This command creates a task to release the node with the ID `827a1f4e-56e5-404f-9113-88748c18f0c2` from the compute cluster.

Task outcome:

```
# vinfra task show 3b39738c-80a6-40a6-a50d-c3c8118ed212
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.presentation.compute.tasks.DeleteComputeNodesTask |
| result  |                                                            |
| state   | success                                                    |
| task_id | 3b39738c-80a6-40a6-a50d-c3c8118ed212                       |
+---------+------------------------------------------------------------+
```

Managing virtual machines

vinfra service compute server create

Create a new virtual machine:

```
usage: vinfra service compute server create [--description <description>]
                                            [--metadata <metadata>]
                                            [--user-data <user-data>]
                                            [--key-name <key-name>]
                                            [--config-drive] [--count <count>]
                                            [--ha-enabled {true,false}]
                                            [--placements <placements>]
                                            [--allow-live-resize]
                                            --network id|<id=id[,key=value,…]>
                                            --volume <source=source
                                            [,key=value,…]>
                                            --flavor <flavor> <server-name>
```

`--description <description>`

Virtual machine description

`--metadata <metadata>`

Virtual machine metadata

`--user-data <user-data>`

User data file

`--key-name <key-name>`

Key pair to inject

`--config-drive`

Use an ephemeral drive

`--count <count>`

If count is specified and greater than 1, the `name` argument is treated as a naming pattern.

`--ha-enabled {true,false}`

Enable or disable HA for the virtual machine.

`--placements <placements>`

Names or IDs of placements to add the virtual machine to.

`--allow-live-resize`

Allow online resize for the virtual machine.

`--network id|<id=id[,key=value,…]>`

Create a virtual machine with a specified network. Specify this option multiple times to create multiple networks.

- `id`: attach network interface to a specified network (ID or name)
- comma-separated `key=value` pairs with keys (optional):
  - `mac`: MAC address for network interface
  - `fixed-ip`: fixed IP address or `None` to automatically allocate an IP address. This option can be used multiple times.
  - `spoofing-protection-enable`: enable spoofing protection for network interface
  - `spoofing-protection-disable`: disable spoofing protection for network interface
  - `security-group`: security group ID or name. This option can be used multiple times.
  - `no-security-group`: do not use a security group

`--volume <source=source[,key=value,…]>`

Create a virtual machine with a specified volume. Specify this option multiple times to create multiple volumes.

- `source`: source type (`volume`, `image`, `snapshot`, or `blank`)
- comma-separated `key=value` pairs with keys (optional):
  - `id`: resource ID or name for the specified source type (required for source types `volume`, `image`, and `snapshot`)
  - `size`: block device size, in gigabytes (required for source types `image` and `blank`)
  - `boot-index`: block device boot index (required for multiple volumes with source type `volume`)
  - `bus`: block device controller type (`scsi`)
  - `type`: block device type (`disk` or `cdrom`)
  - `rm`: remove block device on virtual machine termination (`yes` or `no`)
  - `storage-policy`: block device storage policy

`--flavor <flavor>`

Flavor ID or name

`<server-name>`

A new name for the virtual machine

Example:

```
# vinfra service compute server create myvm \
--network id=private,fixed-ip=192.168.128.100 \
--volume source=image,id=cirros,size=1 --flavor tiny
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allow_live_resize | False                                |
| config_drive      |                                      |
| created           | 2019-05-29T11:24:04Z                 |
| description       |                                      |
| flavor            | disk: 0                              |
|                   | ephemeral: 0                         |
|                   | extra_specs: {}                      |
|                   | original_name: tiny                  |
|                   | ram: 512                             |
|                   | swap: 0                              |
|                   | vcpus: 1                             |
| ha_enabled        | True                                 |
| host              |                                      |
| id                | 8cd29296-8bee-4efb-828d-0e522d816c6e |
| key_name          |                                      |
| metadata          | {}                                   |
| name              | myvm                                 |
| networks          | []                                   |
| power_state       | NOSTATE                              |
| project_id        | b4267de6fd0c442da99542cd20f5932c     |
| status            | BUILD                                |
| task_state        | scheduling                           |
| updated           | 2019-05-29T11:24:21Z                 |
| user_data         |                                      |
| vm_state          | building                             |
| volumes           | []                                   |
+-------------------+--------------------------------------+
```

This command creates a virtual machine `myvm` based on the default Cirros image and the flavor `tiny`, connects it to the network `private` with the fixed IP address 192.168.128.100, and enables HA for it.

vinfra service compute server list

List virtual machines:

```
usage: vinfra service compute server list [--long] [--limit <num>]
                                          [--marker <server>] [--name <name>]
                                          [--id <id>] [--project <project>]
                                          [--status <status>]
                                          [--task-status <task-status>]
```

```
                                        [--host <hostname>]
                                        [--placement <placement>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of virtual machines to list. To list all virtual machines, set the option to -1.

`--marker <server>`

List virtual machines after the marker.

`--name <name>`

List virtual machines with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a server with the specified ID or list virtual machines using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--project <project>`

List virtual machines that belong to the specified project ID. Can only be performed by system administrators.

`--status <status>`

List virtual machines with the specified status.

`--task-status <task-status>`

List virtual machines that have the specified task status.

`--host <hostname>`

List virtual machines located on a node with the specified hostname.

`--placement <placement>`

List virtual machines added to a placement with the specified ID or use a filter. Supported filter operator: `any`. The filter format is `<operator>:<value1>[,<value2>,…]`.

Example:

```
# vinfra service compute server list
+--------------------------------------+------+--------+------------------------+
| id                                   | name | status | host                   |
+--------------------------------------+------+--------+------------------------+
| 8cd29296-8bee-4efb-828d-0e522d816c6e | myvm | ACTIVE | node001.vstoragedomain |
+--------------------------------------+------+--------+------------------------+
```

This command lists all virtual machines in the compute cluster.

vinfra service compute server show

Display virtual machine details:

```
usage: vinfra service compute server show <server>
```

<server>

Virtual machine ID or name

Example:

```
# vinfra service compute server show myvm
+-------------------+------------------------------------------+
| Field             | Value                                    |
+-------------------+------------------------------------------+
| allow_live_resize | False                                    |
| config_drive      |                                          |
| created           | 2019-05-29T11:24:04Z                     |
| description       |                                          |
| flavor            | disk: 0                                  |
|                   | ephemeral: 0                             |
|                   | extra_specs: {}                          |
|                   | original_name: tiny                      |
|                   | ram: 512                                 |
|                   | swap: 0                                  |
|                   | vcpus: 1                                 |
| ha_enabled        | True                                     |
| host              | node001.vstoragedomain                   |
| id                | 8cd29296-8bee-4efb-828d-0e522d816c6e     |
| key_name          |                                          |
| metadata          | {}                                       |
| name              | myvm                                     |
| networks          | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|                   |   ipam_enabled: true                     |
|                   |   ips:                                   |
|                   |   - 192.168.128.100                      |
|                   |   mac_addr: fa:16:3e:d8:42:f6            |
|                   |   name: private                          |
|                   |   spoofing_protection: true              |
| orig_hostname     | node001                                  |
| placements        | []                                       |
| power_state       | RUNNING                                  |
| project_id        | b4267de6fd0c442da99542cd20f5932c         |
| status            | ACTIVE                                   |
| task_state        |                                          |
| updated           | 2019-05-29T11:24:21Z                     |
| user_data         |                                          |
| vm_state          | active                                   |
| volumes           | - delete_on_termination: false           |
```

```
|                   |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+-------------------+--------------------------------------------+
```

This command shows the details of the virtual machine `myvm`.

vinfra service compute server stat

Display virtual machine statistics:

```
usage: vinfra service compute server stat <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server stat myvm
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| datetime | 2019-05-29T11:39:46.429000+00:00 |
| metrics  | block_capacity: 1073741824       |
|          | block_usage: 268435456           |
|          | cpu_usage: 0                     |
|          | mem_usage: 149876736             |
+----------+----------------------------------+
```

This command shows the statistics for the virtual machine `myvm`.

vinfra service compute server set

Modify virtual machine parameters:

```
usage: vinfra service compute server set [--name <name>]
                                         [--description <description>]
                                         [--ha-enabled <ha_enabled>]
                                         [--no-placements | --placement placement]
                                         [--allow-live-resize | --deny-live-resize]
                                         <server>
```

`--name <name>`

A new name for the virtual machine

`--description <description>`

A new description for the virtual machine

`--ha-enabled {true,false}`

Enable or disable HA for the virtual machine.

`--no-placements`

Clean up placements from the virtual machine.

`--placement placement`

Placement name or ID to add the virtual machine to. Specify this option multiple times to add the virtual machine to multiple placements.

`--allow-live-resize`

Allow online resize for the virtual machine.

`--deny-live-resize`

Deny online resize for the virtual machine.

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server set myvm --description "My new VM" --ha-enabled false
+-------------------+------------------------------------------+
| Field             | Value                                    |
+-------------------+------------------------------------------+
| allow_live_resize | False                                    |
| config_drive      |                                          |
| created           | 2019-05-29T11:24:04Z                     |
| description       | My new VM                                |
| flavor            | disk: 0                                  |
|                   | ephemeral: 0                             |
|                   | extra_specs: {}                          |
|                   | original_name: tiny                      |
|                   | ram: 512                                 |
|                   | swap: 0                                  |
|                   | vcpus: 1                                 |
| ha_enabled        | False                                    |
| host              | node001.vstoragedomain                   |
| id                | 8cd29296-8bee-4efb-828d-0e522d816c6e     |
| key_name          |                                          |
| metadata          | {}                                       |
| name              | myvm                                     |
| networks          | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|                   |   ipam_enabled: true                     |
|                   |   ips:                                   |
|                   |   - 192.168.128.100                      |
|                   |   mac_addr: fa:16:3e:d8:42:f6            |
|                   |   name: private                          |
|                   |   spoofing_protection: true              |
| orig_hostname     | node001                                  |
| placements        | []                                       |
| power_state       | RUNNING                                  |
| project_id        | b4267de6fd0c442da99542cd20f5932c         |
| status            | ACTIVE                                   |
| task_state        |                                          |
| updated           | 2019-05-29T11:24:21Z                     |
```

```
| user_data         |                                          |
| vm_state          | active                                   |
| volumes           | - delete_on_termination: false           |
|                   |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+-------------------+------------------------------------------+
```

This command adds a description to the virtual machine `myvm` and disables HA for it.

vinfra service compute server iface attach

Attach a network to a virtual machine:

```
usage: vinfra service compute server iface attach [--fixed-ip <ip-address | ip-address=
                                                  <ip-address>,subnet=<subnet> |
                                                  ip-version=<ip-version>>]
                                                  [--spoofing-protection-enable |
                                                  --spoofing-protection-disable]
                                                  [--security-group
                                                  <security-group> |
                                                  --no-security-groups]
                                                  --server <server>
                                                  --network <network>
                                                  [--mac <mac>]
```

`--fixed-ip <ip-address|ip-address=<ip-address>,subnet=<subnet>|ip-version=<ip-version>>`

Desired IP address and/or subnet. This option can be used multiple times.

`--spoofing-protection-enable`

Enable spoofing protection for the network interface

`--spoofing-protection-disable`

Disable spoofing protection for the network interface

`--security-group <security-group>`

Security group ID or name. This option can be used multiple times.

`--no-security-groups`

Do not set security groups

`--server <server>`

Virtual machine ID or name

`--network <network>`

Network ID or name

`--mac <mac>`

MAC address

Example:


```
# vinfra service compute server iface attach --network myprivnet --fixed-ip
192.168.129.8 --server myvm
+---------------------+----------------------------------------+
| Field               | Value                                  |
+---------------------+----------------------------------------+
| fixed_ip            | 192.168.129.8                          |
| id                  | 690ed3f2-2301-40e2-879a-126db2ecb57b   |
| mac_address         | fa:16:3e:54:59:08                      |
| network_id          | 0710372e-2bdf-4dfe-b413-eb763da37e68   |
| security_groups     | - 1b919f1b-eb9a-492e-9305-ed98d8b1a4e9 |
| spoofing_protection | True                                   |
+---------------------+----------------------------------------+
```

This command attaches the virtual network `myprivnet` to the virtual machine `myvm`.

vinfra service compute server iface list

List virtual machine networks:

```
usage: vinfra service compute server iface list [--long] --server <server>
```

`--long`

Enable access and listing of all fields of objects.

`--server <server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server iface list --server myvm
+---------------+---------------+-------------------+-----------------+
| id            | network_id    | mac_address       | fixed_ips       |
+---------------+---------------+-------------------+-----------------+
| 690ed3f2-<...> | 0710372e-<...> | fa:16:3e:54:59:08 | 192.168.129.8   |
| a5b13bf3-<...> | 1bf2c9da-<...> | fa:16:3e:b9:33:bb | 192.168.128.100 |
+---------------+---------------+-------------------+-----------------+
```

This command lists the virtual networks that the virtual machine `myvm` is attached to. It also shows VM's IP address in each network.

vinfra service compute server iface set

Update a network of a virtual machine:

```
usage: vinfra service compute server iface set [--fixed-ip <ip-address | ip-address=
                                                <ip-address>,subnet=<subnet> |
                                                ip-version=<ip-version>>]
                                                [--spoofing-protection-enable |
                                                --spoofing-protection-disable]
```

```
                                                [--security-group
                                                <security-group> |
                                                --no-security-groups]
                                                --server <server> <interface>
```

`--fixed-ip <ip-address|ip-address=<ip-address>,subnet=<subnet>|ip-version=<ip-version>>`

Desired IP address and/or subnet. This option can be used multiple times.

`--spoofing-protection-enable`

Enable spoofing protection for the network interface

`--spoofing-protection-disable`

Disable spoofing protection for the network interface

`--security-group <security-group>`

Security group ID or name. This option can be used multiple times.

`--no-security-groups`

Do not set security groups

`--server <server>`

Virtual machine ID or name

`<interface>`

Network interface ID

Example:

```
# vinfra service compute server iface set --server myvm --no-security-groups \
611abc06-7557-44c9-bbf8-31fef817e802
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ips           | - 192.168.128.100                    |
|                     | - 192.168.128.200                    |
| id                  | 611abc06-7557-44c9-bbf8-31fef817e802 |
| mac_addr            | fa:16:3e:a6:d4:32                    |
| network_id          | 8774a1a4-f7a0-4729-be9b-d282751434c5 |
| security_groups     | []                                   |
| spoofing_protection | True                                 |
+---------------------+--------------------------------------+
```

This command unassigns security groups from the network interface with the ID 611abc06-7557-44c9-bbf8-31fef817e802 attached to the virtual machine `myvm`.

vinfra service compute server iface detach

Detach a network interface from a virtual machine:

```
usage: vinfra service compute server iface detach --server <server> <interface>
```


`--server <server>`

Virtual machine ID or name

`<interface>`

Network interface ID

Example:

```
# vinfra service compute server iface detach 471e37fd-13ae-4b8f-b70c-90ac02cc4386 \
--server 6c80b07f-da46-4a8a-89a4-eecb8faceb27
Operation successful.
```

This command detaches the network interface with the ID `471e37fd-13ae-4b8f-b70c-90ac02cc4386` from the VM with the ID `6c80b07f-da46-4a8a-89a4-eecb8faceb27`.

vinfra service compute server volume attach

Attach a volume to a virtual machine:

```
usage: vinfra service compute server volume attach --server <server> <volume>
```

`--server <server>`

Virtual machine ID or name

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute server volume attach e4cb5363-1fb2-41f5-b24b-18f98a388cba \
--server 871fef54-519b-4111-b18d-d2039e2410a8
+--------+--------------------------------------+
| Field  | Value                                |
+--------+--------------------------------------+
| device | /dev/vdb                             |
| id     | e4cb5363-1fb2-41f5-b24b-18f98a388cba |
+--------+--------------------------------------+
```

This command attaches the available volume with the ID `e4cb5363-1fb2-41f5-b24b-18f98a388cba` to the VM with the ID `871fef54-519b-4111-b18d-d2039e2410a8`.

vinfra service compute server volume list

List virtual machine volumes:

```
usage: vinfra service compute server volume list [--long] --server <server>
```

`--long`

Enable access and listing of all fields of objects.

`--server <server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server volume list --server myvm
+--------------------------------------+----------+
| id                                   | device   |
+--------------------------------------+----------+
| e4cb5363-1fb2-41f5-b24b-18f98a388cba | /dev/vdb |
| b325cc6e-8de1-4b6c-9807-5a497e3da7e3 | /dev/vda |
+--------------------------------------+----------+
```

This command lists the volumes attached to the virtual machine `myvm`.

vinfra service compute server volume show

Show details of a virtual machine volume:

```
usage: vinfra service compute server volume show --server <server> <volume>
```

`--server <server>`

Virtual machine ID or name

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute server volume show --server myvm \
e4cb5363-1fb2-41f5-b24b-18f98a388cba
+--------+--------------------------------------+
| Field  | Value                                |
+--------+--------------------------------------+
| device | /dev/vdb                             |
| id     | e4cb5363-1fb2-41f5-b24b-18f98a388cba |
+--------+--------------------------------------+
```

This command shows the details for the volume with the ID `e4cb5363-1fb2-41f5-b24b-18f98a388cba` attached to the virtual machine `myvm`.

vinfra service compute server volume detach

Detach a volume from a virtual machine:

```
usage: vinfra service compute server volume detach --server <server> <volume>
```

`--server <server>`

Virtual machine ID or name

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute server volume detach e4cb5363-1fb2-41f5-b24b-18f98a388cba \
--server 871fef54-519b-4111-b18d-d2039e2410a8
Operation successful.
```

This command detaches the volume with the ID `e4cb5363-1fb2-41f5-b24b-18f98a388cba` from the VM with the ID `871fef54-519b-4111-b18d-d2039e2410a8`.

vinfra service compute server log

Display virtual machine log:

```
usage: vinfra service compute server log <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server log myvm > myvm.log
```

This command prints the log of the virtual machine `myvm` to the file `myvm.log`.

vinfra service compute server migrate

Migrate a virtual machine to another host:

```
usage: vinfra service compute server migrate [--cold] [--node <node>] <server>
```

`--cold`

Perform cold migration. If not set, the migration type is determined automatically.

`--node <node>`

Destination node ID or hostname

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server migrate 6c80b07f-da46-4a8a-89a4-eecb8faceb27 \
--node e6255aed-d6e7-41b2-ba90-86164c1cd9a6
Operation successful.
```

This command starts migration of the VM with the ID `6c80b07f-da46-4a8a-89a4-eecb8faceb27` to the compute node with the ID `e6255aed-d6e7-41b2-ba90-86164c1cd9a6`.

vinfra service compute server resize

Resize a virtual machine:

```
usage: vinfra service compute server resize --flavor <flavor> <server>
```

`--flavor <flavor>`

Apply flavor with ID or name

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server resize myvm --flavor small
+---------------+-------------------------------------------+
| Field         | Value                                     |
+---------------+-------------------------------------------+
| config_drive  |                                           |
| created       | 2019-05-29T11:24:04Z                      |
| description   |                                           |
| flavor        | disk: 0                                   |
|               | ephemeral: 0                              |
|               | extra_specs: {}                           |
|               | original_name: tiny                       |
|               | ram: 512                                  |
|               | swap: 0                                   |
|               | vcpus: 1                                  |
| ha_enabled    | False                                     |
| host          | node001.vstoragedomain                    |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e      |
| key_name      |                                           |
| metadata      | {}                                        |
| name          | myvm                                      |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                      |
|               |   ips:                                    |
|               |   - 192.168.128.100                       |
|               |   mac_addr: fa:16:3e:d8:42:f6             |
|               |   name: private                           |
|               |   spoofing_protection: true               |
| orig_hostname | node001                                   |
| placements    | []                                        |
| power_state   | SHUTDOWN                                  |
| project_id    | b4267de6fd0c442da99542cd20f5932c          |
| status        | SHUTOFF                                   |
| task_state    |                                           |
| updated       | 2019-05-29T11:24:21Z                      |
```

```
| user_data     |                                          |
| vm_state      | stopped                                  |
| volumes       | - delete_on_termination: false           |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+------------------------------------------+
```

This command changes the flavor of the virtual machine `myvm` to `small`.

vinfra service compute server start

Start a virtual machine:

```
usage: vinfra service compute server start <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server start myvm
+---------------+------------------------------------------+
| Field         | Value                                    |
+---------------+------------------------------------------+
| config_drive  |                                          |
| created       | 2019-05-29T11:24:04Z                     |
| description   |                                          |
| flavor        | disk: 0                                  |
|               | ephemeral: 0                             |
|               | extra_specs: {}                          |
|               | original_name: tiny                      |
|               | ram: 512                                 |
|               | swap: 0                                  |
|               | vcpus: 1                                 |
| ha_enabled    | False                                    |
| host          | node001.vstoragedomain                   |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e     |
| key_name      |                                          |
| metadata      | {}                                       |
| name          | myvm                                     |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                     |
|               |   ips:                                   |
|               |   - 192.168.128.100                      |
|               |   mac_addr: fa:16:3e:d8:42:f6            |
|               |   name: private                          |
|               |   spoofing_protection: true              |
| orig_hostname | node001                                  |
| placements    | []                                       |
| power_state   | SHUTDOWN                                 |
| project_id    | b4267de6fd0c442da99542cd20f5932c         |
```

```
| status        | SHOTOFF                                     |
| task_state    |                                             |
| updated       | 2019-05-29T11:24:21Z                        |
| user_data     |                                             |
| vm_state      | stopped                                     |
| volumes       | - delete_on_termination: false              |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+---------------------------------------------+
```

This command starts the virtual machine `myvm`.

vinfra service compute server pause

Pause a virtual machine:

```
usage: vinfra service compute server pause <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server pause myvm
+---------------+---------------------------------------------+
| Field         | Value                                       |
+---------------+---------------------------------------------+
| config_drive  |                                             |
| created       | 2019-05-29T11:24:04Z                        |
| description   |                                             |
| flavor        | disk: 0                                     |
|               | ephemeral: 0                                |
|               | extra_specs: {}                             |
|               | original_name: tiny                         |
|               | ram: 512                                    |
|               | swap: 0                                     |
|               | vcpus: 1                                    |
| ha_enabled    | False                                       |
| host          | node001.vstoragedomain                      |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e        |
| key_name      |                                             |
| metadata      | {}                                          |
| name          | myvm                                        |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                        |
|               |   ips:                                      |
|               |   - 192.168.128.100                         |
|               |   mac_addr: fa:16:3e:d8:42:f6               |
|               |   name: private                             |
|               |   spoofing_protection: true                 |
| orig_hostname | node001                                     |
```

```
| placements    | []                                          |
| power_state   | RUNNING                                     |
| project_id    | b4267de6fd0c442da99542cd20f5932c            |
| status        | ACTIVE                                      |
| task_state    |                                             |
| updated       | 2019-05-29T11:24:21Z                        |
| user_data     |                                             |
| vm_state      | active                                      |
| volumes       | - delete_on_termination: false              |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+---------------------------------------------+
```

This command pauses the running virtual machine `myvm`.

vinfra service compute server unpause

Unpause a virtual machine:

```
usage: vinfra service compute server unpause <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server unpause myvm
+---------------+---------------------------------------------+
| Field         | Value                                       |
+---------------+---------------------------------------------+
| config_drive  |                                             |
| created       | 2019-05-29T11:24:04Z                        |
| description   | My new VM                                   |
| flavor        | disk: 0                                     |
|               | ephemeral: 0                                |
|               | extra_specs: {}                             |
|               | original_name: tiny                         |
|               | ram: 512                                    |
|               | swap: 0                                     |
|               | vcpus: 1                                    |
| ha_enabled    | False                                       |
| host          | node001.vstoragedomain                      |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e        |
| key_name      |                                             |
| metadata      | {}                                          |
| name          | myvm                                        |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                        |
|               |   ips:                                      |
|               |   - 192.168.128.100                         |
|               |   mac_addr: fa:16:3e:d8:42:f6               |
```

```
|               |   name: private                              |
|               |   spoofing_protection: true                  |
| orig_hostname | node001                                      |
| placements    | []                                           |
| power_state   | PAUSED                                       |
| project_id    | b4267de6fd0c442da99542cd20f5932c             |
| status        | PAUSED                                       |
| task_state    |                                              |
| updated       | 2019-05-29T11:24:21Z                         |
| vm_state      | paused                                       |
| user_data     |                                              |
| volumes       | - delete_on_termination: false               |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+----------------------------------------------+
```

This command unpauses the paused virtual machine `myvm`.

vinfra service compute server suspend

Suspend a virtual machine:

```
usage: vinfra service compute server suspend <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server suspend myvm
+---------------+----------------------------------------------+
| Field         | Value                                        |
+---------------+----------------------------------------------+
| config_drive  |                                              |
| created       | 2019-05-29T11:24:04Z                         |
| description   |                                              |
| flavor        | disk: 0                                      |
|               | ephemeral: 0                                 |
|               | extra_specs: {}                              |
|               | original_name: tiny                          |
|               | ram: 512                                     |
|               | swap: 0                                      |
|               | vcpus: 1                                     |
| ha_enabled    | False                                        |
| host          | node001.vstoragedomain                       |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e         |
| key_name      |                                              |
| metadata      | {}                                           |
| name          | myvm                                         |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                         |
```

```
|               |   ips:                                      |
|               |   - 192.168.128.100                         |
|               |   mac_addr: fa:16:3e:d8:42:f6               |
|               |   name: private                             |
|               |   spoofing_protection: true                 |
| orig_hostname | node001                                     |
| placements    | []                                          |
| power_state   | RUNNING                                     |
| project_id    | b4267de6fd0c442da99542cd20f5932c            |
| status        | ACTIVE                                      |
| task_state    |                                             |
| updated       | 2019-05-29T11:24:21Z                        |
| user_data     |                                             |
| vm_state      | active                                      |
| volumes       | - delete_on_termination: false              |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+---------------------------------------------+
```

This command suspends the running virtual machine `myvm`.

vinfra service compute server resume

Resume a virtual machine:

```
usage: vinfra service compute server resume <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server resume myvm
+---------------+---------------------------------------------+
| Field         | Value                                       |
+---------------+---------------------------------------------+
| config_drive  |                                             |
| created       | 2019-05-29T11:24:04Z                        |
| description   |                                             |
| flavor        | disk: 0                                     |
|               | ephemeral: 0                                |
|               | extra_specs: {}                             |
|               | original_name: tiny                         |
|               | ram: 512                                    |
|               | swap: 0                                     |
|               | vcpus: 1                                    |
| ha_enabled    | False                                       |
| host          | node001.vstoragedomain                      |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e        |
| key_name      |                                             |
| metadata      | {}                                          |
```

```
| name          | myvm                                         |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                         |
|               |   ips:                                       |
|               |   - 192.168.128.100                          |
|               |   mac_addr: fa:16:3e:d8:42:f6                |
|               |   name: private                              |
|               |   spoofing_protection: true                  |
| orig_hostname | node001                                      |
| placements    | []                                           |
| power_state   | SHUTDOWN                                     |
| project_id    | b4267de6fd0c442da99542cd20f5932c             |
| status        | SUSPENDED                                    |
| task_state    |                                              |
| updated       | 2019-05-29T11:24:21Z                         |
| user_data     |                                              |
| vm_state      | suspended                                    |
| volumes       | - delete_on_termination: false               |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+----------------------------------------------+
```

This command resumes the suspended virtual machine `myvm`.

vinfra service compute server reboot

Reboot a virtual machine:

```
usage: vinfra service compute server reboot [--hard] <server>
```

`--hard`

Perform hard reboot

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server reboot myvm
+---------------+----------------------------------------------+
| Field         | Value                                        |
+---------------+----------------------------------------------+
| config_drive  |                                              |
| created       | 2019-05-29T11:24:04Z                         |
| description   |                                              |
| flavor        | disk: 0                                      |
|               | ephemeral: 0                                 |
|               | extra_specs: {}                              |
|               | original_name: tiny                          |
|               | ram: 512                                     |
|               | swap: 0                                      |
```



```
|               | vcpus: 1                                    |
| ha_enabled    | False                                       |
| host          | node001.vstoragedomain                      |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e        |
| key_name      |                                             |
| metadata      | {}                                          |
| name          | myvm                                        |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                        |
|               |   ips:                                      |
|               |   - 192.168.128.100                         |
|               |   mac_addr: fa:16:3e:d8:42:f6               |
|               |   name: private                             |
|               |   spoofing_protection: true                 |
| orig_hostname | node001                                     |
| placements    | []                                          |
| power_state   | RUNNING                                     |
| project_id    | b4267de6fd0c442da99542cd20f5932c            |
| status        | ACTIVE                                      |
| task_state    |                                             |
| updated       | 2019-05-29T11:24:21Z                        |
| user_data     |                                             |
| vm_state      | active                                      |
| volumes       | - delete_on_termination: false              |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+---------------------------------------------+
```

This command reboots the virtual machine `myvm`.

vinfra service compute server reset-state

Reset virtual machine state:

```
usage: vinfra service compute server reset-state [--state-error] <server>
```

`--state-error`

Reset virtual machine to ‘ERROR’ state

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server reset-state myvm
+---------------+---------------------------------------------+
| Field         | Value                                       |
+---------------+---------------------------------------------+
| config_drive  |                                             |
| created       | 2019-05-29T11:24:04Z                        |
| description   |                                             |
```

```
| flavor        | disk: 0                                      |
|               | ephemeral: 0                                 |
|               | extra_specs: {}                              |
|               | original_name: tiny                          |
|               | ram: 512                                     |
|               | swap: 0                                      |
|               | vcpus: 1                                     |
| ha_enabled    | False                                        |
| host          | node001.vstoragedomain                       |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e         |
| key_name      |                                              |
| metadata      | {}                                           |
| name          | myvm                                         |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7   |
|               |   ipam_enabled: true                         |
|               |   ips:                                       |
|               |   - 192.168.128.100                          |
|               |   mac_addr: fa:16:3e:d8:42:f6                |
|               |   name: private                              |
|               |   spoofing_protection: true                  |
| orig_hostname | node001                                      |
| placements    | []                                           |
| power_state   | SHUTDOWN                                     |
| project_id    | b4267de6fd0c442da99542cd20f5932c             |
| status        | VERIFY_RESIZE                                |
| task_state    |                                              |
| updated       | 2019-05-29T11:24:21Z                         |
| user_data     |                                              |
| vm_state      | resized                                      |
| volumes       | - delete_on_termination: false               |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94   |
+---------------+----------------------------------------------+
```

This command resets the transitional state of the virtual machine `myvm` to the previous one.

vinfra service compute server stop

Shut down a virtual machine:

```
usage: vinfra service compute server stop [--hard | --wait-time <seconds>] <server>
```

`--hard`

Power off a virtual machine

`--wait-time <seconds>`

Shutdown timeout, after which a virtual machine will be powered off. Specify '-1' to set an infinite timeout.

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server stop myvm
+---------------+------------------------------------------+
| Field         | Value                                    |
+---------------+------------------------------------------+
| config_drive  |                                          |
| created       | 2019-05-29T11:24:04Z                     |
| description   |                                          |
| flavor        | disk: 0                                  |
|               | ephemeral: 0                             |
|               | extra_specs: {}                          |
|               | original_name: tiny                      |
|               | ram: 512                                 |
|               | swap: 0                                  |
|               | vcpus: 1                                 |
| ha_enabled    | False                                    |
| host          | node001.vstoragedomain                   |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e     |
| key_name      |                                          |
| metadata      | {}                                       |
| name          | myvm                                     |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                     |
|               |   ips:                                   |
|               |   - 192.168.128.100                      |
|               |   mac_addr: fa:16:3e:d8:42:f6            |
|               |   name: private                          |
|               |   spoofing_protection: true              |
| orig_hostname | node001                                  |
| placements    | []                                       |
| power_state   | RUNNING                                  |
| project_id    | b4267de6fd0c442da99542cd20f5932c         |
| status        | ACTIVE                                   |
| task_state    |                                          |
| updated       | 2019-05-29T11:24:21Z                     |
| user_data     |                                          |
| vm_state      | active                                   |
| volumes       | - delete_on_termination: false           |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+------------------------------------------+
```

This command stops the virtual machine `myvm`.

vinfra service compute server cancel-stop

Cancel shutdown for a virtual machine:

```
usage: vinfra service compute server cancel-stop <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server cancel-stop myvm
Operation successful.
```

This command cancels shutdown for the virtual machine `myvm` if it has the “powering-off” task state.

vinfra service compute server shelve

Shelve a virtual machine:

```
usage: vinfra service compute server shelve <server>
```

`<server>`

Virtual machine ID or name.

Example:

```
# vinfra service compute server shelve myvm
+---------------+-------------------------------------------+
| Field         | Value                                     |
+---------------+-------------------------------------------+
| config_drive  |                                           |
| created       | 2019-05-29T11:24:04Z                      |
| description   |                                           |
| flavor        | disk: 0                                   |
|               | ephemeral: 0                              |
|               | extra_specs: {}                           |
|               | original_name: tiny                       |
|               | ram: 512                                  |
|               | swap: 0                                   |
|               | vcpus: 1                                  |
| ha_enabled    | False                                     |
| host          | node001.vstoragedomain                    |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e      |
| key_name      |                                           |
| metadata      | {}                                        |
| name          | myvm                                      |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                      |
|               |   ips:                                    |
|               |   - 192.168.128.100                       |
|               |   mac_addr: fa:16:3e:d8:42:f6             |
|               |   name: private                           |
|               |   spoofing_protection: true               |
| orig_hostname | node001                                   |
| placements    | []                                        |
| power_state   | SHUTDOWN                                  |
| project_id    | b4267de6fd0c442da99542cd20f5932c          |
| status        | SHOTOFF                                   |
```

```
| task_state    |                                            |
| updated       | 2019-05-29T11:24:21Z                       |
| user_data     |                                            |
| vm_state      | stopped                                    |
| volumes       | - delete_on_termination: false             |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+--------------------------------------------+
```

This command unbinds the virtual machine `myvm` from the node it is hosted on and releases its reserved resources such as CPU and RAM.

vinfra service compute server unshelve

Unshelve a virtual machine:

```
usage: vinfra service compute server unshelve <server>
```

`<server>`

Virtual machine ID or name.

Example:

```
# vinfra service compute server unshelve myvm
+---------------+--------------------------------------------+
| Field         | Value                                      |
+---------------+--------------------------------------------+
| config_drive  |                                            |
| created       | 2019-05-29T11:24:04Z                       |
| description   |                                            |
| flavor        | disk: 0                                    |
|               | ephemeral: 0                               |
|               | extra_specs: {}                            |
|               | original_name: tiny                        |
|               | ram: 512                                   |
|               | swap: 0                                    |
|               | vcpus: 1                                   |
| ha_enabled    | False                                      |
| host          | node001.vstoragedomain                     |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e       |
| key_name      |                                            |
| metadata      | {}                                         |
| name          | myvm                                       |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                       |
|               |   ips:                                     |
|               |   - 192.168.128.100                        |
|               |   mac_addr: fa:16:3e:d8:42:f6              |
|               |   name: private                            |
|               |   spoofing_protection: true                |
| orig_hostname | node001                                    |
```

```
| placements    | []                                          |
| power_state   | SHUTDOWN                                    |
| project_id    | b4267de6fd0c442da99542cd20f5932c            |
| status        | SHELVED_OFFLOADED                           |
| task_state    |                                             |
| updated       | 2019-05-29T11:24:21Z                        |
| user_data     |                                             |
| vm_state      | shelved_offloaded                           |
| volumes       | - delete_on_termination: false              |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+---------------------------------------------+
```

This command spawns the virtual machine `myvm` on a node with enough resources to host it.

vinfra service compute server evacuate

Evacuate a stopped virtual machine from a failed host:

```
usage: vinfra service compute server evacuate <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server evacuate myvm
+---------------+---------------------------------------------+
| Field         | Value                                       |
+---------------+---------------------------------------------+
| config_drive  |                                             |
| created       | 2019-05-29T11:24:04Z                        |
| description   |                                             |
| flavor        | disk: 0                                     |
|               | ephemeral: 0                                |
|               | extra_specs: {}                             |
|               | original_name: tiny                         |
|               | ram: 512                                    |
|               | swap: 0                                     |
|               | vcpus: 1                                    |
| ha_enabled    | False                                       |
| host          | node001.vstoragedomain                      |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e        |
| key_name      |                                             |
| metadata      | {}                                          |
| name          | myvm                                        |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                        |
|               |   ips:                                      |
|               |   - 192.168.128.100                         |
|               |   mac_addr: fa:16:3e:d8:42:f6               |
```

```
|               |   name: private                             |
|               |   spoofing_protection: true                 |
| orig_hostname | node001                                     |
| placements    | []                                          |
| power_state   | SHUTDOWN                                    |
| project_id    | b4267de6fd0c442da99542cd20f5932c            |
| status        | SHUTOFF                                     |
| task_state    |                                             |
| updated       | 2019-05-29T11:24:21Z                        |
| user_data     |                                             |
| vm_state      | stopped                                     |
| volumes       | - delete_on_termination: false              |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+---------------------------------------------+
```

This command evacuates the stopped VM `myvm` from its node to another, healthy compute node.

vinfra service compute server delete

Delete a virtual machine:

```
usage: vinfra service compute server delete <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server delete myvm
Operation accepted.
```

This command deletes the virtual machine `myvm`.

vinfra service compute server rescue

Put a virtual machine to the rescue mode:

```
usage: vinfra service compute server rescue [--image <image>] <server>
```

`<server>`

Virtual machine ID or name

`--image <image>`

Boot from image ID or name

Example:

```
# vinfra service compute server rescue myvm --image cirros
+---------------+------------------------------------------+
| Field         | Value                                    |
+---------------+------------------------------------------+
| config_drive  |                                          |
| created       | 2019-05-29T11:24:04Z                     |
| description   | My new VM                                |
| fault         |                                          |
| flavor        | disk: 0                                  |
|               | ephemeral: 0                             |
|               | extra_specs: {}                          |
|               | original_name: tiny                      |
|               | ram: 512                                 |
|               | swap: 0                                  |
|               | vcpus: 1                                 |
| ha_enabled    | False                                    |
| host          | node001.vstoragedomain                   |
| host_status   | UP                                       |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e     |
| key_name      |                                          |
| metadata      | {}                                       |
| name          | myvm                                     |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                     |
|               |   ips:                                   |
|               |   - 192.168.128.100                      |
|               |   mac_addr: fa:16:3e:d8:42:f6            |
|               |   name: private                          |
|               |   spoofing_protection: true              |
| orig_hostname | node001                                  |
| placements    | []                                       |
| power_state   | RUNNING                                  |
| project_id    | b4267de6fd0c442da99542cd20f5932c         |
| status        | ACTIVE                                   |
| task_state    |                                          |
| traits        | []                                       |
| updated       | 2019-05-29T11:24:21Z                     |
| user_data     |                                          |
| vm_state      | active                                   |
| volumes       | - delete_on_termination: false           |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+------------------------------------------+
```

This command sends the `myvm` virtual machine to the rescue mode with the `cirros` image.

vinfra service compute server unrescue

Exit a virtual machine from the rescue mode:

```
usage: vinfra service compute server unrescue <server>
```

`<server>`

Virtual machine ID or name

Example:

```
# vinfra service compute server unrescue myvm
+---------------+------------------------------------------+
| Field         | Value                                    |
+---------------+------------------------------------------+
| config_drive  |                                          |
| created       | 2019-05-29T11:24:04Z                     |
| description   | My new VM                                |
| fault         |                                          |
| flavor        | disk: 0                                  |
|               | ephemeral: 0                             |
|               | extra_specs: {}                          |
|               | original_name: tiny                      |
|               | ram: 512                                 |
|               | swap: 0                                  |
|               | vcpus: 1                                 |
| ha_enabled    | False                                    |
| host          | node001.vstoragedomain                   |
| host_status   | UP                                       |
| id            | 8cd29296-8bee-4efb-828d-0e522d816c6e     |
| key_name      |                                          |
| metadata      | {}                                       |
| name          | myvm                                     |
| networks      | - id: 79b3da71-c6a2-49e8-97f8-9431a065bed7 |
|               |   ipam_enabled: true                     |
|               |   ips:                                   |
|               |   - 192.168.128.100                      |
|               |   mac_addr: fa:16:3e:d8:42:f6            |
|               |   name: private                          |
|               |   spoofing_protection: true              |
| orig_hostname | node001                                  |
| placements    | []                                       |
| power_state   | RUNNING                                  |
| project_id    | b4267de6fd0c442da99542cd20f5932c         |
| status        | RESCUE                                   |
| task_state    |                                          |
| traits        | []                                       |
| updated       | 2019-05-29T11:24:21Z                     |
| user_data     |                                          |
| vm_state      | rescued                                  |
| volumes       | - delete_on_termination: false           |
|               |   id: edd3df0a-95f5-4892-9053-2793a3976f94 |
+---------------+------------------------------------------+
```

This command stops the rescue mode for the `myvm` virtual machine.

If you face issues stopping the rescue mode for a Windows VM, refer to "Exiting the rescue mode for Windows virtual machines" (p. 354).

Managing images

vinfra service compute image create

Create a new compute image:

```
usage: vinfra service compute image create [--min-disk <size-gb>]
                                           [--min-ram <size-mb>]
                                           [--os-distro <os-distro>]
                                           [--protected | --unprotected]
                                           [--public] [--public]
                                           [--disk-format <disk_format>]
                                           [--container-format <format>]
                                           [--tags <tags>] --file <file>
                                           <image-name>
```

`--min-disk <size-gb>`

Minimum disk size required to boot from image, in gigabytes

`--min-ram <size-mb>`

Minimum RAM size required to boot from image, in megabytes

`--os-distro <os-distro>`

OS distribution. To list available distributions, run `vinfra service compute show`.

`--protected`

Protect image from deletion

`--unprotected`

Allow image to be deleted

`--public`

Make image accessible to all users

`--private`

Make image accessible only to the owners.

`--disk-format <disk_format>`

Disk format: `detect`, `iso`, `qcow2`, `raw` (default: `detect`)

`--container-format <format>`

Container format: `bare`

`--tags <tags>`

A comma-separated list of tags

`--file <file>`

Create image from a local file

`<image-name>`

Image name

Example:

```
# vinfra service compute image create mycirrosimg \
--file /distr/cirros-0.4.0-x86_64-disk.img
Uploading image to server [elapsed time: 0:00:04]... |
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 03874663-d03f-4891-a10b-64837e7faf43 |
+---------+--------------------------------------+
```

This command creates a task to create a Cirros image from the local file and upload it to Virtuozzo Hybrid Infrastructure.

Task outcome:

```
# vinfra task show 03874663-d03f-4891-a10b-64837e7faf43
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.compute.images.tasks.ImportComputeImageTask |
| result  | id: 179f45ef-c5d6-4270-b0c0-085b542544c5                         |
| state   | success                                                          |
| task_id | 03874663-d03f-4891-a10b-64837e7faf43                             |
+---------+------------------------------------------------------------------+
```

vinfra service compute image list

List compute images:

```
usage: vinfra service compute image list [--long] [--limit <num>]
                                         [--marker <image>] [--name <name>]
                                         [--id <id>] [--status <status>]
                                         [--placement <placement>]
                                         [--disk-format <disk-format>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of images to list. To list all images, set the option to -1.

`--marker <image>`

List images after the marker.

`--name <name>`

List images with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show an image with the specified ID or list images using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--status <status>`

List images with the specified status or use a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--placement <placement>`

List images added to a placement with the specified ID or use a filter. Supported filter operator: `any`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--disk-format <disk-format>`

List images with the specified disk format.

Example:

```
# vinfra service compute image list
+--------------------+-------------+----------+--------+-------------+
| id                 | name        |     size | status | disk_format |
+--------------------+-------------+----------+--------+-------------+
| 179f45ef-c5d6-<...> | mycirrosimg | 12716032 | active | qcow2       |
| 4741274f-5cca-<...> | cirros      | 12716032 | active | qcow2       |
+--------------------+-------------+----------+--------+-------------+
```

This command lists images available to the compute cluster.

vinfra service compute image show

Display compute image details:

```
usage: vinfra service compute image show <image>
```

`<image>`

Image ID or name

Example:

```
# vinfra service compute image show 4741274f-5cca-4205-8f66-a2e89fb346cc
+------------------+------------------------------------------------+
| Field            | Value                                          |
+------------------+------------------------------------------------+
| checksum         | 443b7623e27ecf03dc9e01ee93f67afe               |
| container_format | bare                                           |
| created_at       | 2018-09-11T13:29:10Z                           |
| disk_format      | qcow2                                          |
```



```
| file             | /api/v2/compute/images/4741274f-5cca-<...>/file/ |
| id               | 4741274f-5cca-4205-8f66-a2e89fb346cc             |
| min_disk         | 1                                                |
| min_ram          | 0                                                |
| name             | cirros                                           |
| os_distro        | linux                                            |
| os_type          | linux                                            |
| placements       | []                                               |
| project_id       | 72a5db3a033c403a86756021e601ef34                 |
| protected        | False                                            |
| public           | True                                             |
| size             | 12716032                                         |
| status           | active                                           |
| tags             | []                                               |
| updated_at       | 2018-09-11T13:29:13Z                             |
| virtual_size     |                                                  |
+------------------+--------------------------------------------------+
```

This command shows the details of the default Cirros image.

vinfra service compute image set

Modify compute image parameters:

```
usage: vinfra service compute image set [--min-disk <size-gb>]
                                        [--min-ram <size-mb>]
                                        [--os-distro <os-distro>]
                                        [--protected | --unprotected]
                                        [--public] [--private]
                                        [--name <name>] <image>
```

`--min-disk <size-gb>`
: Minimum disk size required to boot from image, in gigabytes

`--min-ram <size-mb>`
: Minimum RAM size required to boot from image, in megabytes

`--os-distro <os-distro>`
: OS distribution. To list available distributions, run `vinfra service compute show`.

`--protected`
: Protect image from deletion

`--unprotected`
: Allow image to be deleted

`--public`
: Make image accessible to all users

`--private`
: Make image accessible only to the owners.

`--name <name>`

Image name

`<image>`

Image ID or name

Example:

```
# vinfra service compute image set 4741274f-5cca-4205-8f66-a2e89fb346cc
--protected --min-ram 1
+------------------+------------------------------------------------+
| Field            | Value                                          |
+------------------+------------------------------------------------+
| checksum         | 443b7623e27ecf03dc9e01ee93f67afe               |
| container_format | bare                                           |
| created_at       | 2018-09-11T13:29:10Z                           |
| disk_format      | qcow2                                          |
| file             | /api/v2/compute/images/4741274f-5cca-<...>/file/ |
| id               | 4741274f-5cca-4205-8f66-a2e89fb346cc           |
| min_disk         | 1                                              |
| min_ram          | 1                                              |
| name             | cirros                                         |
| os_distro        | linux                                          |
| os_type          | linux                                          |
| project_id       | 72a5db3a033c403a86756021e601ef34               |
| protected        | True                                           |
| size             | 12716032                                       |
| status           | active                                         |
| tags             | []                                             |
| updated_at       | 2018-09-12T09:26:29Z                           |
| virtual_size     |                                                |
| visibility       | public                                         |
+------------------+------------------------------------------------+
```

This command protects the default Cirros image and sets the minimum RAM size for it to 1 GB.

vinfra service compute image save

Download a compute image:

```
usage: vinfra service compute image save [--file <filename>] <image>
```

`--file <filename>`

File to save the image to (default: `stdout`)

`<image>`

Image ID or name

Example:

```
# vinfra service compute image save 4741274f-5cca-4205-8f66-a2e89fb346cc --file
cirros.qcow2
Operation successful
```

This command downloads the default Cirros image to the local disk as `cirros.qcow2`.

vinfra service compute image delete

Delete a compute image:

```
usage: vinfra service compute image delete <image>
```

`<image>`

Image ID or name

Example:

```
# vinfra service compute image delete 179f45ef-c5d6-4270-b0c0-085b542544c5
Operation successful
```

This command deletes the image with the ID `179f45ef-c5d6-4270-b0c0-085b542544c5`.

Managing placements

vinfra service compute placement create

Create a new compute placement:

```
usage: vinfra service compute placement create [--isolated | --non-isolated]
                                               [--description <description>]
                                               [--nodes <nodes>]
                                               [--images <images>]
                                               [--flavors <flavors>]
                                               <placement-name>
```

`--isolated`

Create an isolated placement (hard policy, default)

`--non-isolated`

Create a non-isolated placement (soft policy)

`--description <description>`

Placement description

`--nodes <nodes>`

A comma-separated list of compute node IDs or hostnames to assign to a compute placement

`--images <images>`

A comma-separated list of image IDs or names to assign to a compute placement

`--flavors <flavors>`

A comma-separated list of flavor IDs or names to assign to a compute placement

`<placement-name>`

Placement name

Example:

```
# vinfra service compute placement create placement1 \
--nodes node001,node002,node003 --flavors 101
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| description |                                      |
| flavors     | 1                                    |
| id          | e4230b75-a858-404c-be3b-4b3f2dedb057 |
| images      | 0                                    |
| name        | placement1                           |
| nodes       | 3                                    |
| servers     | 0                                    |
+-------------+--------------------------------------+
```

This command creates a placement called `placement1` with the hard policy for the nodes `node001`, `node002`, `node003` and the flavor with the ID 101.

vinfra service compute placement assign

Assign nodes or images to a placement:

```
usage: vinfra service compute placement assign (--images <images> |
                                                --nodes <nodes> |
                                                --flavors <flavors>)
                                                <placement>
```

`--images <images>`

A comma-separated list of image IDs or names to assign to a compute placement

`--nodes <nodes>`

A comma-separated list of compute node IDs or hostnames to assign to a compute placement

`--flavors <flavors>`

A comma-separated list of flavor IDs or names to assign to a compute placement

`<placement>`

Placement ID or name

A virtual machine is assigned a placement when it is created from an image or flavor with the placement assigned. A VM can also inherit a placement from a volume created with an assigned

image. However, a VM does not inherit the placement and changes to it from the node. For example, if you assign a placement to a node with existing VMs, only the node will have the placement. The VMs will not inherit the same placement. Likewise, if you have a node and VMs on it assigned to the same placement, and you delete such a node from a placement, only the node will change the placement. The VMs on it will still keep the original placement.

If you need to edit the VM placement, use the "vinfra service compute server set" (p. 100) command. Make sure that the node and its VMs have the same placement configuration.

Example:

```
# vinfra service compute placement assign \
--images b23e23e8-7338-4a09-a827-3c9c509cf35c placement1
Operation successful.
```

This command assigns the image with the ID `b23e23e8-7338-4a09-a827-3c9c509cf35c` to the placement `placement1`.

vinfra service compute placement delete-assign

Remove images and nodes from a compute placement:

```
usage: vinfra service compute placement delete-assign (--image <images> |
                                                       --node <nodes> |
                                                       --flavor <flavors>)
                                                       <placement>
```

`--image <image>`

An image ID or name to remove from a compute placement

`--node <node>`

A compute node ID or hostname to remove from a compute placement

`--flavor <flavor>`

A flavor ID or name to remove from a compute placement

`<placement>`

Placement ID or name

A virtual machine is assigned a placement when it is created from an image with the placement assigned. A VM can also inherit a placement from a volume created with an assigned image (refer to "Creating, editing, and removing volumes" in the Self-Service Guide). However, a VM does not inherit the placement and changes to it from the node. For example, if you add a node with existing VMs to a placement, only the node will have the placement. The VMs will not inherit the same placement. Likewise, if you have a node and VMs on it assigned to the same placement, and you delete such a node from a placement, only the node will change the placement. The VMs on it will still keep the original placement.

If you need to edit the VM placement, use the "vinfra service compute server set" (p. 100) command. Make sure that the node and its VMs have the same placement configuration.

Example:

```
# vinfra service compute placement delete-assign \
--image b23e23e8-7338-4a09-a827-3c9c509cf35c placement1
Operation successful.
```

This command removes the image with the ID `b23e23e8-7338-4a09-a827-3c9c509cf35c` from the placement `placement1`.

vinfra service compute placement list

List compute placements:

```
usage: vinfra service compute placement list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute placement list -c id -c name -c nodes \
-c images -c flavors -c isolated
+----------------------+------------+-------+--------+---------+----------+
| id                   | name       | nodes | images | flavors | isolated |
+----------------------+------------+-------+--------+---------+----------+
| e4230b75-a858-<...> | placement1 | 3     | 0      | 1       | True     |
+----------------------+------------+-------+--------+---------+----------+
```

This command lists available compute placements.

vinfra service compute placement show

Display compute placement details:

```
usage: vinfra service compute placement show <placement>
```

`<placement>`

Placement ID or name

Example:

```
# vinfra service compute placement show placement1
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
```

```
| description |                                      |
| flavors     | 1                                    |
| id          | e4230b75-a858-404c-be3b-4b3f2dedb057 |
| images      | 0                                    |
| name        | placement1                           |
| nodes       | 3                                    |
| servers     | 0                                    |
+-------------+--------------------------------------+
```

This command shows the details of the placement `placement1`.

vinfra service compute placement update

Update a compute placement:

```
usage: vinfra service compute placement update [--name <placement-name>]
                                               [--description <description>]
                                               [--non-isolated | --isolated]
                                               <placement>
```

`--name <placement-name>`
: A new name for the placement

`--description <description>`
: A new description for the placement

`--non-isolated`
: Make the placement non-isolated (soft policy)

`--isolated`
: Make the placement isolated (hard policy)

`<placement>`
: Placement ID or name

Example:

```
# vinfra service compute placement update --name placement1-UPD placement1
Operation successful
```

This command renames the placement `placement1` to `placement1-UPD`.

vinfra service compute placement delete

Delete a compute placement:

```
usage: vinfra service compute placement delete <placement>
```

`<placement>`

Placement ID or name

Example:

```
# vinfra service compute placement delete placement1-UPD
Operation successful
```

This command deletes the placement `placement1-UPD`.

Managing flavors

vinfra service compute flavor create

Create a new compute flavor:

```
usage: vinfra service compute flavor create [--swap <size-mb>] --vcpus <vcpus>
                                            --ram <size-mb> <flavor-name>
```

`--swap <size-mb>`

Swap space size, in megabytes

`--vcpus <vcpus>`

Number of virtual CPUs

`--ram <size-mb>`

Memory size, in megabytes

`<flavor-name>`

Flavor name

Example:

```
# vinfra service compute flavor create myflavor --vcpus 1 --ram 3072
+-------+--------------------------------------+
| Field | Value                                |
+-------+--------------------------------------+
| id    | 561a48ea-0c1c-4152-8b7d-e4b4af276c2d |
| name  | myflavor                             |
| ram   | 3072                                 |
| swap  | 0                                    |
| vcpus | 1                                    |
+-------+--------------------------------------+
```

This command creates a flavor `myflavor` with 1 vCPU and 3 GB RAM.

vinfra service compute flavor list

List compute flavors:


```
usage: vinfra service compute flavor list [--long] [--placement <placement>]
```

`--long`

Enable access and listing of all fields of objects.

`--placement <placement>`

List flavors added to a placement with the specified ID or use a filter. Supported filter operator: `any`. The filter format is `<operator>:<value1>[,<value2>,…]`.

Example:

```
# vinfra service compute flavor list
+--------------------------------------+----------+-------+------+-------+
| id                                   | name     |   ram | swap | vcpus |
+--------------------------------------+----------+-------+------+-------+
| 100                                  | tiny     |   512 |    0 |     1 |
| 101                                  | small    |  2048 |    0 |     1 |
| 102                                  | medium   |  4096 |    0 |     2 |
| 103                                  | large    |  8192 |    0 |     4 |
| 104                                  | xlarge   | 16384 |    0 |     8 |
| 561a48ea-0c1c-4152-8b7d-e4b4af276c2d | myflavor |  3072 |    0 |     1 |
+--------------------------------------+----------+-------+------+-------+
```

This command lists all flavors.

vinfra service compute flavor show

Display compute flavor details:

```
usage: vinfra service compute flavor show <flavor>
```

`<flavor>`

Flavor ID or name

Example:

```
# vinfra service compute flavor show myflavor
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| id         | 561a48ea-0c1c-4152-8b7d-e4b4af276c2d |
| name       | myflavor                             |
| placements | []                                   |
| ram        | 3072                                 |
| swap       | 0                                    |
| vcpus      | 1                                    |
+------------+--------------------------------------+
```

This command shows the details of the flavor `myflavor`.

vinfra service compute flavor delete

Delete a compute flavor:

```
usage: vinfra service compute flavor delete <flavor>
```

`<flavor>`

Flavor ID or name

Example:

```
# vinfra service compute flavor delete myflavor
Operation successful
```

This command deletes the flavor `myflavor`.

Managing compute SSH keys

vinfra service compute key create

Create a new compute SSH key:

```
usage: vinfra service compute key create --public-key <public-key>
                                         [--description <description>] <ssh-key>
```

`---public-key <public-key>`

Filename for a public key to upload

`--description <description>`

SSH key description

`<ssh-key>`

SSH key name

Example:

```
# vinfra service compute key create publickey --public-key /root/.ssh/id_rsa.pub \
--description 'public key'
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| created_at  | 2019-04-25T13:41:14.241736+00:00 |
| description | public key                       |
| name        | publickey                        |
+-------------+----------------------------------+
```

This command creates a public SSH key `publickey`.

vinfra service compute key list

List compute SSH keys:

```
usage: vinfra service compute key list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute key list
+-----------+-------------+----------------------------------+
| name      | description | created_at                       |
+-----------+-------------+----------------------------------+
| testkey   | test key    | 2019-04-24T13:41:05.209837+00:00 |
| publickey | public key  | 2019-04-25T13:41:14.241736+00:00 |
+-----------+-------------+----------------------------------+
```

This command lists all SSH keys.

vinfra service compute key show

Display compute SSH key details:

```
usage: vinfra service compute key show <ssh-key>
```

`<ssh-key>`

SSH key name

Example:

```
# vinfra service compute key show publickey
+------------------------+--------------------------------------------------+
| Field                  | Value                                            |
+------------------------+--------------------------------------------------+
| created_at             | 2019-04-25T13:41:14.241736+00:00                 |
| description            | public key                                       |
| name                   | publickey                                        |
| public_key_fingerprint | 1a:fb:de:d8:1e:0a:84:30:fc:ff:e4:fd:89:e7:96:a9 |
+------------------------+--------------------------------------------------+
```

This command shows the details of the SSH key `publickey`.

vinfra service compute key delete

Delete a compute SSH key:

```
usage: vinfra service compute key delete <ssh-key>
```

`<ssh-key>`

SSH key name

Example:

```
# vinfra service compute key delete publickey
Operation successful
```

This command deletes the SSH key `publickey`.

Managing compute networks

vinfra service compute network create

Create a compute network:

```
usage: vinfra service compute network create [--dhcp | --no-dhcp]
                                             [--dns-nameserver <dns-nameserver>]
                                             [--allocation-pool <allocation-pool>]
                                             [--gateway <gateway> | --no-gateway]
                                             [--rbac-policies <rbac-policies>]
                                             [--physical-network <physical-network>]
                                             [--vlan-network <vlan-network>]
                                             [--vlan <vlan>] [--cidr <cidr>]
                                             [--ipv6-address-mode <ipv6-address-mode>]
                                             <network-name>
```

`--dhcp`

Enable DHCP.

`--no-dhcp`

Disable DHCP.

`--dns-nameserver <dns-nameserver>`

DNS server IP address. This option can be used multiple times.

`--allocation-pool <allocation-pool>`

Allocation pool to create inside the network in the format: `ip_addr_start-ip_addr_end`. This option can be used multiple times.

`--gateway <gateway>`

Gateway IP address

`--no-gateway`

Do not configure a gateway for this network.

`--rbac-policies <rbac-policies>`

Comma-separated list of RBAC policies in the format: `<target>:<target_id>:<action> | none`. Valid targets: `project`, `domain`. Valid actions: `direct`, `full`, `routed`. '*' is valid `target_id` for all targets. Pass `none` to clear out all existing policies.

Example: `domain:default:routed,project:uuid1:full`

`--physical-network <physical-network>`

An infrastructure network to link to a physical network

`--vlan-network <vlan-network>`

A VLAN network to link

`--vlan <vlan>`

Virtual network VLAN ID

`--cidr <cidr>`

Subnet range in CIDR notation

`--ipv6-address-mode <ipv6-address-mode>`

IPv6 address mode: `dhcpv6-stateful`, `dhcpv6-stateless`, `slaac`

`<network-name>`

Network name

Example 1. Creating a virtual network:

```
# vinfra service compute network create myprivnet
+------------------+--------------------------------------+
| Field            | Value                                |
+------------------+--------------------------------------+
| allocation_pools |                                      |
| cidr             |                                      |
| dns_nameservers  |                                      |
| enable_dhcp      |                                      |
| gateway_ip       |                                      |
| id               | a0019b43-fe64-4b30-8feb-ff772e293769 |
| ip_version       |                                      |
| ipam_enabled     | False                                |
| name             | myprivnet                            |
| physical_network |                                      |
| project_id       | 6b04700556634b60895804e7ef52df3d     |
| rbac_policies    | []                                   |
| router_external  | False                                |
| shared           | False                                |
| tags             | []                                   |
| type             | virtual                              |
| vlan_id          |                                      |
+------------------+--------------------------------------+
```

This command creates a virtual network `myprivnet` with disabled IP management.

Example 2. Creating an untagged physical network and sharing it with a particular domain:


```
# vinfra service compute network create mypubnet --physical-network Public \
--cidr 10.136.16.0/22 --gateway 10.136.16.1 --dns-nameserver 10.35.11.7 \
--allocation-pool 10.136.18.141-10.136.18.148 \
--rbac-policies domain:cd421db9f3e84e3e8cd2c932c1f7a698:full
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 00551a29-c240-4273-ad8f-88535c6113ac |
+---------+--------------------------------------+
```

This command creates a task to create an untagged physical network over the `Public` infrastructure network, with enabled IP management, the specified network parameters, and full network access between all the projects within the specified domain.

Task outcome:

```
# vinfra task show 00551a29-c240-4273-ad8f-88535c6113ac
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.compute.network.tasks.CreateComputeNetwork |
| result  | id: 22674f9d-1c94-4953-b79b-7f6029ee9bd0                         |
|         | ipam_enabled: true                                               |
|         | name: mypubnet                                                   |
|         | physical_network: Public                                         |
|         | project_id: c22613639b3147e0b22ef057b87698fe                      |
|         | rbac_policies:                                                   |
|         | - actions:                                                       |
|         |   - routed                                                       |
|         |   - shared                                                       |
|         |   target_project: f59a0d9a4cd543daa73160575d48611b               |
|         | router_external: false                                           |
|         | shared: false                                                    |
|         | subnet:                                                          |
|         |   allocation_pools:                                              |
|         |   - end: 10.136.18.148                                           |
|         |     start: 10.136.18.141                                         |
|         |   cidr: 10.136.16.0/22                                           |
|         |   dns_nameservers:                                               |
|         |   - 10.35.11.7                                                   |
|         |   enable_dhcp: true                                              |
|         |   gateway_ip: 10.136.16.1                                        |
|         |   ip_version: 4                                                  |
|         | tags: []                                                         |
|         | type: flat                                                       |
|         | vlan_id: null                                                    |
| state   | success                                                          |
| task_id | 00551a29-c240-4273-ad8f-88535c6113ac                             |
+---------+------------------------------------------------------------------+
```

Example 3. Creating a VLAN-based physical network and sharing it between all projects:


```
# vinfra service compute network create mypubnet_vlan --vlan 10 \
--physical-network Public --cidr 10.136.16.0/22 --gateway 10.136.16.1 \
--dns-nameserver 10.35.11.7 --allocation-pool 10.136.18.131-10.136.18.138 \
--rbac-policies project:*:shared
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 3ec1afee-8fe5-4d0c-89da-84c971bf23cd |
+---------+--------------------------------------+
```

This command creates a task to create a VLAN-based physical network over the `Public` infrastructure network, with the VLAN ID 10, enabled IP management, the specified network parameters, and direct (shared) network access between all the projects in the infrastructure.

Task outcome:

```
# vinfra task show 3ec1afee-8fe5-4d0c-89da-84c971bf23cd
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.presentation.compute.network.tasks.CreateComputeNetwork |
| result  | id: 8f0dc747-4c8f-42ad-9a4b-31d7d81c61fd                       |
|         | ipam_enabled: true                                                     |
|         | name: mypubnet_vlan                                                    |
|         | physical_network: Public                                               |
|         | project_id: c22613639b3147e0b22ef057b87698fe                    |
|         | rbac_policies:                                                         |
|         | - actions:                                                             |
|         |   - shared                                                             |
|         |   target_project: '*'                                                  |
|         | router_external: false                                                 |
|         | shared: false                                                          |
|         | subnet:                                                                |
|         |   allocation_pools:                                                    |
|         |   - end: 10.136.18.138                                                 |
|         |     start: 10.136.18.131                                               |
|         |   cidr: 10.136.16.0/22                                                 |
|         |   dns_nameservers:                                                     |
|         |   - 10.35.11.7                                                         |
|         |   enable_dhcp: true                                                    |
|         |   gateway_ip: 10.136.16.1                                              |
|         |   ip_version: 4                                                        |
|         | tags: []                                                               |
|         | type: vlan                                                             |
|         | vlan_id: 10                                                            |
| state   | success                                                                |
| task_id | 3ec1afee-8fe5-4d0c-89da-84c971bf23cd                                   |
+---------+------------------------------------------------------------------------+
```

vinfra service compute network list

List compute networks:

```
usage: vinfra service compute network list [--long] [--limit <num>]
                                           [--marker <network>]
                                           [--name <name>] [--id <id>]
                                           [--project <project>]
                                           [--type <type>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of networks to list. To list all networks, set the option to -1.

`--marker <network>`

List networks after the marker.

`--name <name>`

List networks with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a network with the specified ID or list networks using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--project <project>`

List networks that belong to the specified project ID. Can only be performed by system administrators.

`--type <type>`

List networks with the specified type.

Example:

```
# vinfra service compute network list -c id -c name -c cidr -c allocation_pools
+---------------+--------------+---------------+-----------------------------+
| id            | name         | cidr          | allocation_pools            |
+---------------+--------------+---------------+-----------------------------+
| 22674f9d-<...> | mypubnet     | 10.136.16.0/22 | 10.136.18.141-10.136.18.148 |
| 8f0dc747-<...> | mypubnet_vlan | 10.136.16.0/22 | 10.136.18.131-10.136.18.138 |
| a0019b43-<...> | myprivnet    |               |                             |
+---------------+--------------+---------------+-----------------------------+
```

This command lists networks used in the compute cluster.

vinfra service compute network show

Display compute network details:

```
usage: vinfra service compute network show <network>
```

`<network>`

Network ID or name

Example:

```
# vinfra service compute network show mypubnet
+------------------+-------------------------------------------------------+
| Field            | Value                                                 |
+------------------+-------------------------------------------------------+
| allocation_pools | 10.136.18.141-10.136.18.148                           |
| cidr             | 10.136.16.0/22                                        |
| dns_nameservers  | 10.35.11.7                                            |
| enable_dhcp      | True                                                  |
| gateway_ip       | 10.136.16.1                                           |
| id               | 22674f9d-1c94-4953-b79b-7f6029ee9bd0                  |
| ip_version       | 4                                                     |
| ipam_enabled     | True                                                  |
| name             | mypubnet                                              |
| physical_network | Public                                                |
| project_id       | c22613639b3147e0b22ef057b87698fe                      |
| rbac_policies    | - actions:                                            |
|                  |   - routed                                            |
|                  |   - shared                                            |
|                  |   target_domain: cd421db9f3e84e3e8cd2c932c1f7a698     |
|                  |   target_project: f59a0d9a4cd543daa73160575d48611b    |
| router_external  | True                                                  |
| shared           | False                                                 |
| tags             | []                                                    |
| type             | physical                                              |
| vlan_id          |                                                       |
+------------------+-------------------------------------------------------+
```

This command shows the details of the compute network `mypubnet`.

vinfra service compute network set

Modify compute network parameters:

```
usage: vinfra service compute network set [--dhcp | --no-dhcp]
                                          [--dns-nameserver <dns-nameserver>]
                                          [--allocation-pool <allocation-pool>]
                                          [--gateway <gateway> | --no-gateway]
```

```
                                        [--rbac-policies <rbac-policies>]
                                        [--name <name>] <network>
```

`--dhcp`

Enable DHCP.

`--no-dhcp`

Disable DHCP.

`--dns-nameserver <dns-nameserver>`

DNS server IP address. This option can be used multiple times.

`--allocation-pool <allocation-pool>`

Allocation pool to create inside the network in the format: `ip_addr_start-ip_addr_end`. This option can be used multiple times.

`--gateway <gateway>`

Gateway IP address

`--no-gateway`

Do not configure a gateway for this network.

`--rbac-policies <rbac-policies>`

Comma-separated list of RBAC policies in the format: `<target>:<target_id>:<action> | none`. Valid targets: `project`, `domain`. Valid actions: `direct`, `full`, `routed`. '*' is valid `target_id` for all targets. Pass `none` to clear out all existing policies.

Example: `domain:default:routed,project:uuid1:full`

`--name <name>`

A new name for the network

`<network>`

Network ID or name

Example:

```
# vinfra service compute network set mypubnet --rbac-policies none
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| allocation_pools | 10.136.18.141-10.136.18.148          |
| cidr            | 10.136.16.0/22                       |
| dns_nameservers | 10.35.11.7                           |
| enable_dhcp     | True                                 |
| gateway_ip      | 10.136.16.1                          |
| id              | 22674f9d-1c94-4953-b79b-7f6029ee9bd0 |
| ip_version      | 4                                    |
| ipam_enabled    | True                                 |
| name            | mypubnet                             |
```



```
| physical_network | Public                                 |
| project_id       | c22613639b3147e0b22ef057b87698fe       |
| rbac_policies    | []                                     |
| router_external  | False                                  |
| shared           | False                                  |
| tags             | []                                     |
| type             | physical                               |
| vlan_id          |                                        |
+------------------+----------------------------------------+
```

This command disables network access for the compute network `mypubnet`.

vinfra service compute network delete

Delete a compute network:

```
usage: vinfra service compute network delete <network>
```

`<network>`

Network ID or name

Example:

```
# vinfra service compute network delete myprivnet
Operation accepted.
```

This command deletes the compute network `myprivnet`.

Managing security groups

vinfra service compute security-group create

Create a security group:

```
usage: vinfra service compute security-group create [--description <description>]
                                                    <name>
```

`--description <description>`

Security group description

`<name>`

Security group name

Example:

```
# vinfra service compute security-group create mygroup
+---------------------+-------------------------------------------------------+
```

```
| Field                | Value                                                |
+----------------------+------------------------------------------------------+
| description          |                                                      |
| id                   | 12e6b260-0b61-4551-8168-3e59602a2433                 |
| name                 | mygroup                                              |
| project_id           | e215189c0472482f93e71d10e1245253                     |
| security_group_rules | - description: null                                  |
|                      |   direction: egress                                  |
|                      |   ethertype: IPv4                                    |
|                      |   id: ce854e2b-537f-4618-bea9-e9ec3d8616ac           |
|                      |   port_range_max: null                               |
|                      |   port_range_min: null                               |
|                      |   project_id: e215189c0472482f93e71d10e1245253       |
|                      |   protocol: null                                     |
|                      |   remote_group_id: null                              |
|                      |   remote_ip_prefix: null                             |
|                      |   security_group_id: 12e6b260-0b61-4551-8168<...>    |
|                      | - description: null                                  |
|                      |   direction: egress                                  |
|                      |   ethertype: IPv6                                    |
|                      |   id: a7c65861-df3d-47f2-bec3-089747141936           |
|                      |   port_range_max: null                               |
|                      |   port_range_min: null                               |
|                      |   project_id: e215189c0472482f93e71d10e1245253       |
|                      |   protocol: null                                     |
|                      |   remote_group_id: null                              |
|                      |   remote_ip_prefix: null                             |
|                      |   security_group_id: 12e6b260-0b61-4551-8168<...>    |
| tags                 | []                                                   |
+----------------------+------------------------------------------------------+
```

This command creates a security group `mygroup`.

vinfra service compute security-group list

List security groups:

```
usage: vinfra service compute security-group list [--long] [--limit <num>]
                                                  [--marker <marker>]
                                                  [--name <name>] [--id <id>]
                                                  [--project <project>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of security groups to list. To list all security groups, set the option to -1.

`--marker <router>`

List security groups after the marker.

`--name <name>`

List security groups with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a security group with the specified ID or list security groups using a filter. Supported filter `operator`: in. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--project <project>`

List security groups that belong to the specified project ID. Can only be performed by system administrators.

Example:

```
# vinfra service compute security-group list -c id -c name
+--------------------------------------+---------+
| id                                   | name    |
+--------------------------------------+---------+
| 062f75cf-abc0-419d-bb1a-92989ad9383f | default |
| 12e6b260-0b61-4551-8168-3e59602a2433 | mygroup |
+--------------------------------------+---------+
```

This command lists security groups in the compute cluster.

vinfra service compute security-group show

Display information about a security group:

```
usage: vinfra service compute security-group show <security-group>
```

`<security-group>`

Security group name or ID

Example:

```
# vinfra service compute security-group show mygroup
+----------------------+-------------------------------------------------------+
| Field                | Value                                                 |
+----------------------+-------------------------------------------------------+
| description          |                                                       |
| id                   | 12e6b260-0b61-4551-8168-3e59602a2433                  |
| name                 | mygroup                                               |
| project_id           | e215189c0472482f93e71d10e1245253                      |
| security_group_rules | - description: null                                   |
|                      |   direction: egress                                   |
|                      |   ethertype: IPv4                                     |
|                      |   id: ce854e2b-537f-4618-bea9-e9ec3d8616ac            |
|                      |   port_range_max: null                                |
|                      |   port_range_min: null                                |
```

```
|                       |   project_id: e215189c0472482f93e71d10e1245253    |
|                       |   protocol: null                                  |
|                       |   remote_group_id: null                           |
|                       |   remote_ip_prefix: null                          |
|                       |   security_group_id: 12e6b260-0b61-4551-8168<...> |
|                       | - description: null                               |
|                       |   direction: egress                               |
|                       |   ethertype: IPv6                                 |
|                       |   id: a7c65861-df3d-47f2-bec3-089747141936        |
|                       |   port_range_max: null                            |
|                       |   port_range_min: null                            |
|                       |   project_id: e215189c0472482f93e71d10e1245253    |
|                       |   protocol: null                                  |
|                       |   remote_group_id: null                           |
|                       |   remote_ip_prefix: null                          |
|                       |   security_group_id: 12e6b260-0b61-4551-8168<...> |
| tags                  | []                                                |
+-----------------------+---------------------------------------------------+
```

This command shows the details of the security group `mygroup`.

vinfra service compute security-group set

Modify a security group:

```
usage: vinfra service compute security-group set [--name <name>]
                                                 [--description <description>]
                                                 <security-group>
```

`--name <name>`

Security group name

`--description <description>`

Security group description

`<security-group>`

Security group name or ID

Example:

```
# vinfra service compute security-group set mygroup \
--description "A new security group"
+----------------------+-----------------------------------------------------+
| Field                | Value                                               |
+----------------------+-----------------------------------------------------+
| description          | A new security group                                |
| id                   | 12e6b260-0b61-4551-8168-3e59602a2433                |
| name                 | mygroup                                             |
| project_id           | e215189c0472482f93e71d10e1245253                    |
| security_group_rules | - description: null                                 |
```

```
|                      |   direction: egress                                |
|                      |   ethertype: IPv4                                  |
|                      |   id: ce854e2b-537f-4618-bea9-e9ec3d8616ac         |
|                      |   port_range_max: null                             |
|                      |   port_range_min: null                             |
|                      |   project_id: e215189c0472482f93e71d10e1245253     |
|                      |   protocol: null                                   |
|                      |   remote_group_id: null                            |
|                      |   remote_ip_prefix: null                           |
|                      |   security_group_id: 12e6b260-0b61-4551-8168<...>  |
|                      | - description: null                                |
|                      |   direction: egress                                |
|                      |   ethertype: IPv6                                  |
|                      |   id: a7c65861-df3d-47f2-bec3-089747141936         |
|                      |   port_range_max: null                             |
|                      |   port_range_min: null                             |
|                      |   project_id: e215189c0472482f93e71d10e1245253     |
|                      |   protocol: null                                   |
|                      |   remote_group_id: null                            |
|                      |   remote_ip_prefix: null                           |
|                      |   security_group_id: 12e6b260-0b61-4551-8168<...>  |
| tags                 | []                                                 |
+----------------------+----------------------------------------------------+
```

This command adds a description to the security group `mygroup`.

vinfra service compute security-group rule create

Create a security group rule:

```
usage: vinfra service compute security-group rule create [--remote-group
                                                          <remote-group>]
                                                          [--remote-ip
                                                          <ip-address>]
                                                          [--ethertype
                                                          <ethertype>]
                                                          [--protocol <protocol>]
                                                          [--port-range-max
                                                          <port-range-max>]
                                                          [--port-range-min
                                                          <port-range-min>]
                                                          (--ingress | --egress)
                                                          <security-group>
```

`--remote-group <remote-group>`

Remote security group name or ID

`--remote-ip <ip-address>`

Remote IP address block in CIDR notation

`--ethertype <ethertype>`

Ethertype of network traffic: `IPv4` or `IPv6`

`--protocol <protocol>`

IP protocol: `tcp`, `udp`, `icmp`, `vrrp` and others

`--port-range-max <port-range-max>`

The maximum port number in the port range that satisfies the security group rule

`--port-range-min <port-range-min>`

The minimum port number in the port range that satisfies the security group rule

`--ingress`

Rule for incoming network traffic

`--egress`

Rule for outgoing network traffic

`<security-group>`

Security group name or ID to create the rule in

Example:

```
# vinfra service compute security-group rule create mygroup \
--ethertype IPv4 --protocol tcp --port-range-max 22 \
--port-range-min 22 --ingress
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 0f395e2f-a8ab-47f4-b670-64399461393c |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| project_id        | e215189c0472482f93e71d10e1245253     |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 12e6b260-0b61-4551-8168-3e59602a2433 |
+-------------------+--------------------------------------+
```

This command creates a rule in the security group `mygroup` to allow incoming IPv4 network traffic on TCP port 22.

vinfra service compute security-group rule list

List security group rules:


```
usage: vinfra service compute security-group rule list [--long] [--limit <num>]
                                                       [--marker <marker>]
                                                       [--id <id>] [<group>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of security group rules to list. To list all security group rules, set the option to -1.

`--marker <router>`

List security group rules after the marker.

`--id <id>`

Show a security group rule with the specified ID or list security group rules using a filter. Supported filter `operator`: in. The filter format is `<operator>:<value1>[,<value2>,…]`.

`<group>`

List security group rules in a particular security group specified by name or ID.

Example:

```
# vinfra service compute security-group rule list mygroup \
-c id -c direction -c protocol
+--------------------------------------+-----------+----------+
| id                                   | direction | protocol |
+--------------------------------------+-----------+----------+
| 0f395e2f-a8ab-47f4-b670-64399461393c | ingress   | tcp      |
| a7c65861-df3d-47f2-bec3-089747141936 | egress    |          |
| ce854e2b-537f-4618-bea9-e9ec3d8616ac | egress    |          |
+--------------------------------------+-----------+----------+
```

This command lists rules in the security group `mygroup`.

vinfra service compute security-group rule show

Display information about a security group rule:

```
usage: vinfra service compute security-group rule show <security-group-rule>
```

`<security-group-rule>`

Security group rule ID

Example:

```
# vinfra service compute security-group rule show \
0f395e2f-a8ab-47f4-b670-64399461393c
+-------------------+--------------------------------------+
```

```
| Field             | Value                                |
+-------------------+--------------------------------------+
| description       |                                      |
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 0f395e2f-a8ab-47f4-b670-64399461393c |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| project_id        | e215189c0472482f93e71d10e1245253     |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | 12e6b260-0b61-4551-8168-3e59602a2433 |
+-------------------+--------------------------------------+
```

This command shows the details of the security group rule with the ID `0f395e2f-a8ab-47f4-b670-64399461393c`.

vinfra service compute security-group rule delete

Delete a security group rule:

```
usage: vinfra service compute security-group rule delete <security-group-rule>
```

`<security-group-rule>`

Security group rule ID

Example:

```
# vinfra service compute security-group rule delete \
0f395e2f-a8ab-47f4-b670-64399461393c
Operation successful.
```

This command deletes the security group rule with the ID `0f395e2f-a8ab-47f4-b670-64399461393c`.

vinfra service compute security-group delete

Delete a security group:

```
usage: vinfra service compute security-group delete <security-group>
```

`<security-group>`

Security group name or ID

Example:

```
# vinfra service compute security-group delete mygroup
Operation successful.
```

This command deletes the security group `mygroup`.

Managing virtual routers

vinfra service compute router create

Create a virtual router:

```
usage: vinfra service compute router create [--external-gateway <network>]
                                            [--enable-snat | --disable-snat]
                                            [--fixed-ip <fixid-ip>]
                                            [--internal-interface
                                            <network=network,ip-addr=ip-addr> |
                                            <network>] <router-name>
```

`--external-gateway <network>`

Specify a physical network to be used as the router's external gateway (name or ID)

`--enable-snat`

Enable source NAT on the external gateway

`--disable-snat`

Disable source NAT on the external gateway

`--fixed-ip <fixid-ip>`

Desired IP on the external gateway

`--internal-interface <network=network,ip-addr=ip-addr>|<network>`

Specify an internal interface. This option can be used multiple times.

- `network`: name of a virtual network.
- `ip-addr`: an unused IP address from the selected virtual network to assign to the interface; specify if the default gateway of the selected virtual network is in use.

`<router-name>`

Virtual router name

Example:

```
# vinfra service compute router create myrouter --external-gateway public \
--internal-interface private --enable-snat
+----------------------+-------------------------------------------------+
| Field                | Value                                           |
+----------------------+-------------------------------------------------+
| external_gateway_info | enable_snat: true                              |
|                      | ip_addresses:                                   |
|                      | - 10.94.129.76                                  |
|                      | network_id: 720e45bc-4225-49de-9346-26513d8d1262 |
| id                   | b9d8b000-5d06-4768-9f65-2715250cda53            |
```

```
| name                  | myrouter                                          |
| project_id            | 894696133031439f8aaa7e4868dcbd4d                  |
| routes                | []                                                |
| status                | ACTIVE                                            |
+-----------------------+---------------------------------------------------+
```

This command creates a router `myrouter` between the physical network `public` and the virtual network `private` with enabled SNAT on the external gateway.

vinfra service compute router list

List virtual routers:

```
usage: vinfra service compute router list [--long] [--limit <num>]
                                          [--marker <router>]
                                          [--name <name>]
                                          [--id <id>] [--project <project>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of routers to list. To list all routers, set the option to -1.

`--marker <router>`

List routers after the marker.

`--name <name>`

List routers with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a router with the specified ID or list routers using a filter. Supported filter `operator`: in. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--project <project>`

List routers that belong to the specified project ID. Can only be performed by system administrators.

Example:

```
# vinfra service compute router list -c id -c external_gateway_info -c name
-c status
+---------------------+---------------------------------+----------+--------+
| id                  | external_gateway_info           | name     | status |
+---------------------+---------------------------------+----------+--------+
| b9d8b000-5d06-<...> | enable_snat: true               | myrouter | ACTIVE |
|                     | ip_addresses:                   |          |        |
|                     | - 10.94.129.76                  |          |        |
```



```
|                     | network_id: 720e45bc-4225-<...> |          |        |
+---------------------+---------------------------------+----------+--------+
```

This command lists virtual routers used in the compute cluster.

vinfra service compute router show

Display information about a virtual router:

```
usage: vinfra service compute router show <router>
```

`<router>`

Virtual router name

Example:

```
# vinfra service compute router show myrouter
+----------------------+-----------------------------------------------------+
| Field                | Value                                               |
+----------------------+-----------------------------------------------------+
| external_gateway_info | enable_snat: true                                  |
|                      | ip_addresses:                                       |
|                      | - 10.94.129.76                                      |
|                      | network_id: 720e45bc-4225-49de-9346-26513d8d1262 |
| id                   | b9d8b000-5d06-4768-9f65-2715250cda53              |
| name                 | myrouter                                            |
| project_id           | 894696133031439f8aaa7e4868dcbd4d                  |
| routes               | []                                                  |
| status               | ACTIVE                                              |
+----------------------+-----------------------------------------------------+
```

This command shows the details of the virtual router `myrouter`.

vinfra service compute router set

Modify virtual router parameters:

```
usage: vinfra service compute router set [--name <name>]
                                         [--external-gateway <network> |
                                         --no-external-gateway]
                                         [--fixed-ip <fixed-ip>]
                                         [--enable-snat | --disable-snat]
                                         [--route <destination=destination,
                                         nexthop=nexthop> | --no-route]
                                         <router>
```

`--name <name>`

Virtual router name

`--external-gateway <network>`

Specify a physical network to be used as the router's external gateway (name or ID)

`--no-external-gateway`

Remove the external gateway from the router

`--enable-snat`

Enable source NAT on the external gateway

`--disable-snat`

Disable source NAT on the external gateway

`--fixed-ip <fixed-ip>`

Desired IP on the external gateway

`--route <destination=destination,nexthop=nexthop>`

A static route for the router. This option can be used multiple times.

- `destination`: destination subnet range in CIDR notation.
- `nexthop`: next hop IP address from one of the networks that the router is connected to.

`--no-route`

Clear routes associated with the router

`<router>`

Virtual router name or ID

Example:

```
# vinfra service compute router set myrouter --disable-snat --external-gateway public
+----------------------+-----------------------------------------------------+
| Field                | Value                                               |
+----------------------+-----------------------------------------------------+
| external_gateway_info | enable_snat: false                                 |
|                      | ip_addresses:                                       |
|                      | - 10.94.129.76                                      |
|                      | network_id: 720e45bc-4225-49de-9346-26513d8d1262 |
| id                   | b9d8b000-5d06-4768-9f65-2715250cda53              |
| name                 | myrouter                                            |
| project_id           | 894696133031439f8aaa7e4868dcbd4d                  |
| routes               | []                                                  |
| status               | ACTIVE                                              |
+----------------------+-----------------------------------------------------+
```

This command disables SNAT on the external gateway of the virtual router `myrouter`.

vinfra service compute router iface add

Add an interface to a virtual router:


```
usage: vinfra service compute router iface add [--ip-address <ip-address>]
                                               --interface <network> router
```

`--ip-address <ip-address>`

IP address

`--interface <network>`

Network name or ID

`router`

Virtual router name or ID

Example:

```
# vinfra service compute router iface add myrouter --interface private2 \
--ip-address 192.168.30.3
+--------------------------------------+-------------+-----------------+--------+
| network_id                           | is_external | ip_addresses    | status |
+--------------------------------------+-------------+-----------------+--------+
| 720e45bc-4225-49de-9346-26513d8d1262 | True        | - 10.94.129.76  | ACTIVE |
| e6f146ce-a6d0-48b2-9e4f-64a128ce97ae | False       | - 192.168.128.1 | ACTIVE |
| 86803e07-a6d7-4809-9566-1cbe4a89adfd | False       | - 192.168.30.3  | DOWN   |
+--------------------------------------+-------------+-----------------+--------+
```

This command adds an interface from the virtual network `private2` to the virtual router `myrouter` with the IP address `192.168.30.3`.

vinfra service compute router iface list

List router interfaces:

```
usage: vinfra service compute router iface list [--long] router
```

`--long`

Enable access and listing of all fields of objects.

`router`

Virtual router name or ID

Example:

```
# vinfra service compute router iface list myrouter
+--------------------------------+-------------+-----------------+--------+
| network_id                     | is_external | ip_addresses    | status |
+--------------------------------+-------------+-----------------+--------+
| 720e45bc-4225-<...> (public)   | True        | - 10.94.129.76  | ACTIVE |
| e6f146ce-a6d0-<...> (private)  | False       | - 192.168.128.1 | ACTIVE |
| 86803e07-a6d7-<...> (private2) | False       | - 192.168.30.3  | ACTIVE |
+--------------------------------+-------------+-----------------+--------+
```

This command lists interfaces of the virtual router `myrouter`.

vinfra service compute router iface remove

Remove an interface from a virtual router:

```
usage: vinfra service compute router iface remove --interface <network> router
```

`--interface <network>`

Network name or ID

`router`

Virtual router name or ID

Example:

```
# vinfra service compute router iface remove myrouter --interface private2
+--------------------------------------+-------------+-----------------+--------+
| network_id                           | is_external | ip_addresses    | status |
+--------------------------------------+-------------+-----------------+--------+
| 720e45bc-4225-49de-9346-26513d8d1262 | True        | - 10.94.129.76  | ACTIVE |
| e6f146ce-a6d0-48b2-9e4f-64a128ce97ae | False       | - 192.168.128.1 | ACTIVE |
+--------------------------------------+-------------+-----------------+--------+
```

This command removes the interface from the virtual network `private2` from the virtual router `myrouter`.

vinfra service compute router delete

Delete a virtual router:

```
usage: vinfra service compute router delete <router>
```

`<router>`

Virtual router ID or name

Example:

```
# vinfra service compute router delete myrouter
Operation successful
```

This command deletes the virtual router `myrouter`.

Managing floating IP addresses

vinfra service compute floatingip create

Create a floating IP address:


```
usage: vinfra service compute floatingip create [--floating-ip <floating-ip>]
                                                [--port-id <port-id>]
                                                [--fixed-ip <fixed-ip>]
                                                [--description <description>]
                                                --network <network>
```

`--floating-ip <floating-ip>`

Floating IP address

`--port-id <port-id>`

ID of the port to be associated with the floating IP address. To learn the port ID of the selected virtual machine, use the command "vinfra service compute server iface list" (p. 103).

`--fixed-ip <fixed-ip>`

Port IP address (required only if the port has multiple IP addresses)

`--description <description>`

Description of the floating IP address

`--network <network>`

ID or name of the network from which to allocate the floating IP

Example:

```
# vinfra service compute floatingip create 720e45bc-4225-49de-9346-26513d8d1262 \
--port-id 418c8c9e-aaa5-42f2-8da7-24bfead6f28b --fixed-ip-address 192.168.128.5
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attached_to         | a172cb6a-1c7b-4157-9e86-035f3077646f |
| description         |                                      |
| fixed_ip_address    | 192.168.128.5                        |
| floating_ip_address | 10.94.129.72                         |
| floating_network_id | 720e45bc-4225-49de-9346-26513d8d1262 |
| id                  | a709f884-c43f-4a9a-a243-a340d7682ef8 |
| port_id             | 418c8c9e-aaa5-42f2-8da7-24bfead6f28b |
| project_id          | 894696133031439f8aaa7e4868dcbd4d     |
| router_id           | f7f86029-a553-4d61-b7ec-6f581d9c5f5f |
| status              | DOWN                                 |
+---------------------+--------------------------------------+
```

This command creates a floating IP address from the physical network with the ID `720e45bc-4225-49de-9346-26513d8d1262` and assigns it to a virtual machine on port with the ID `418c8c9e-aaa5-42f2-8da7-24bfead6f28b` and the virtual IP address `192.168.128.5`.

vinfra service compute floatingip list

List floating IP addresses:

```
usage: vinfra service compute floatingip list [--long] [--limit <num>]
                                              [--marker <floating-ip>]
                                              [--ip-address <ip-address>]
                                              [--id <id>]
                                              [--network <network>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of floating IPs to list. To list all floating IPs, set the option to -1.

`--marker <floating-ip>`

List floating IPs after the marker.

`--ip-address <ip-address>`

List floating IPs with the specified IP address or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a floating IP with the specified ID or list floating IPs using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--network <network>`

List floating IPs that have the specified network name or ID.

Example:

```
# vinfra service compute floatingip list -c id -c fixed_ip_address -c port_id \
-c floating_ip_address
+---------------+-----------------+---------------+---------------------+
| id            | fixed_ip_address | port_id       | floating_ip_address |
+---------------+-----------------+---------------+---------------------+
| a709f884-<...> | 192.168.128.5    | 418c8c9e-<...> | 10.94.129.72        |
+---------------+-----------------+---------------+---------------------+
```

This command lists floating IP addresses used in the compute cluster.

vinfra service compute floatingip show

Display information about a floating IP address:

```
usage: vinfra service compute floatingip show <floating-ip>
```

`<floating-ip>`

ID of the floating IP address

Example:


```
# vinfra service compute floatingip show a709f884-c43f-4a9a-a243-a340d7682ef8
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attached_to         | a172cb6a-1c7b-4157-9e86-035f3077646f |
| description         |                                      |
| fixed_ip_address    | 192.168.128.5                        |
| floating_ip_address | 10.94.129.72                         |
| floating_network_id | 720e45bc-4225-49de-9346-26513d8d1262 |
| id                  | a709f884-c43f-4a9a-a243-a340d7682ef8 |
| port_id             | 418c8c9e-aaa5-42f2-8da7-24bfead6f28b |
| project_id          | 894696133031439f8aaa7e4868dcbd4d     |
| router_id           | f7f86029-a553-4d61-b7ec-6f581d9c5f5f |
| status              | ACTIVE                               |
+---------------------+--------------------------------------+
```

This command shows the details of the floating IP address with the ID `a709f884-c43f-4a9a-a243-a340d7682ef8`.

vinfra service compute floatingip set

Modify parameters of a floating IP address:

```
usage: vinfra service compute floatingip set [--port-id <port-id>]
                                             [--fixed-ip <fixed-ip>]
                                             [--description <description>]
                                             <floating-ip>
```

`--port-id <port-id>`
: ID of the port to be associated with the floating IP address

`--fixed-ip <fixed-ip>`
: Port IP address (required only if the port has multiple IP addresses)

`--description <description>`
: Description of the floating IP address

`<floating-ip>`
: ID of the floating IP address

Example:

```
# vinfra service compute floatingip set a709f884-c43f-4a9a-a243-a340d7682ef8 \
--description "Floating IP for myvm"
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attached_to         | a172cb6a-1c7b-4157-9e86-035f3077646f |
| description         | Floating IP for myvm                 |
| fixed_ip_address    | 192.168.128.5                        |
```

```
| floating_ip_address | 10.94.129.72                         |
| floating_network_id | 720e45bc-4225-49de-9346-26513d8d1262 |
| id                  | a709f884-c43f-4a9a-a243-a340d7682ef8 |
| port_id             | 418c8c9e-aaa5-42f2-8da7-24bfead6f28b |
| project_id          | 894696133031439f8aaa7e4868dcbd4d     |
| router_id           | f7f86029-a553-4d61-b7ec-6f581d9c5f5f |
| status              | ACTIVE                               |
+---------------------+--------------------------------------+
```

This command adds a description for the floating IP address with the ID `a709f884-c43f-4a9a-a243-a340d7682ef8`.

vinfra service compute floatingip delete

Delete a floating IP address:

```
usage: vinfra service compute floatingip delete <floating-ip>
```

`<floating-ip>`

ID of the floating IP address

Example:

```
# vinfra service compute floatingip delete a709f884-c43f-4a9a-a243-a340d7682ef8
Operation successful
```

This command deletes the floating IP address with the ID `a709f884-c43f-4a9a-a243-a340d7682ef8`.

Managing load balancers

vinfra service compute load-balancer create

Create a load balancer:

```
usage: vinfra service compute load-balancer create [--description <description>]
                                                   [--enable | --disable]
                                                   [--address <address>]
                                                   [--floating-ip <floating-ip>]
                                                   [--pools-config <pools>]
                                                   <name> <network>
```

`--description <description>`

Load balancer description

`--enable`

Enable the load balancer.

`--disable`

Disable the load balancer.

`--address <address>`

The IP address the load balancer will try to allocate in the network.

`--floating-ip <floating-ip>`

The floating IP address that will be used to connect to the load balancer from public networks.

`--pools-config <pools>`

Pool configuration file
Below is an example of a pool configuration file in the YAML format:

```
- backend_protocol: HTTPS
  backend_protocol_port: 443
  healthmonitor: {delay: 5, max_retries: 3, max_retries_down: 3, timeout: 5,
    type: PING, url_path: /}
  lb_algorithm: ROUND_ROBIN
  members:
  - address: 192.168.30.49
  - address: 192.168.30.15
  name: pool1
  protocol: HTTPS
  protocol_port: 443
  sticky_session: False
```

`<name>`

Load balancer name

`<network>`

The ID or name of network the load balancer will operate in.

Example:

```
# vinfra service compute load-balancer create mylbaas private1 \
--floating-ip 10.94.129.70
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| address       | 192.168.30.230                       |
| amphorae      |                                      |
| created_at    | 2019-11-18T12:59:08.243413           |
| description   |                                      |
| enabled       | True                                 |
| floating_ip   | 10.94.129.70                         |
| ha_enabled    |                                      |
| id            | 941bf637-2d55-40f0-92c0-e65d6567b468 |
| members_count | 0                                    |
| name          | mylbaas                              |
| network_id    | 2b821d00-e428-4a76-b1ae-d181c9f5ae7f |
| pools         | []                                   |
| port_id       | 2d8ab88a-847c-4396-857e-11eaa80e1b24 |
```

```
| project_id    | e4e059c67dee4736851df14d4519a5a5     |
| status        | CREATING                             |
| updated_at    |                                      |
+---------------+--------------------------------------+
```

This command creates a load balancer `mylbaas` without balancing pools that will operate in the network `private` with the floating IP address `10.94.129.70`.

vinfra service compute load-balancer list

List load balancers:

```
usage: vinfra service compute load-balancer list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute load-balancer list -c id -c name -c status \
-c floating_ip
+--------------------------------------+----------+--------+--------------+
| id                                   | name     | status | floating_ip  |
+--------------------------------------+----------+--------+--------------+
| 941bf637-2d55-40f0-92c0-e65d6567b468 | mylbaas  | ACTIVE | 10.94.129.70 |
+--------------------------------------+----------+--------+--------------+
```

This command lists load balancers in the compute cluster.

vinfra service compute load-balancer show

Display load balancer details:

```
usage: vinfra service compute load-balancer show <load-balancer>
```

`<load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer show mylbaas
+---------------+-------------------------------------------------------+
| Field         | Value                                                 |
+---------------+-------------------------------------------------------+
| address       | 192.168.30.230                                        |
| amphorae      | - active: true                                        |
|               |   compute_id: b0c4793f-e1b1-4251-91c2-94e34787f537 |
|               |   created_at: '2019-11-18T12:59:12.742446'         |
```



```
|               |   id: b7b23106-a87b-412d-9ce6-7c69b5594342         |
|               |   image_id: 6d1ba6f9-cf86-4ea4-a32d-f138868a9742   |
|               |   role: STANDALONE                                 |
|               |   status: ALLOCATED                                |
|               |   updated_at: '2019-11-18T13:01:07.601184'         |
| created_at    | 2019-11-18T12:59:08.243413                         |
| description   |                                                    |
| enabled       | True                                               |
| floating_ip   | 10.94.129.70                                       |
| ha_enabled    | False                                              |
| id            | 941bf637-2d55-40f0-92c0-e65d6567b468               |
| members_count | 0                                                  |
| name          | mylbaas                                            |
| network_id    | 2b821d00-e428-4a76-b1ae-d181c9f5ae7f               |
| pools         | []                                                 |
| port_id       | 2d8ab88a-847c-4396-857e-11eaa80e1b24               |
| project_id    | e4e059c67dee4736851df14d4519a5a5                   |
| status        | ACTIVE                                             |
| updated_at    | 2019-11-18T13:01:10.983144                         |
+---------------+----------------------------------------------------+
```

This command shows the details of the load balancer `mylbaas`.

vinfra service compute load-balancer stats

Show statistics for a load balancer:

```
usage: vinfra service compute load-balancer stats <load-balancer>
```

`<load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer stats mylbaas
+-------+------------------------------------------------------+
| Field | Value                                                |
+-------+------------------------------------------------------+
| stats | active_connections: 0                                |
|       | bytes_in: 0                                          |
|       | bytes_out: 0                                         |
|       | listeners: null                                      |
|       | loadbalancer_id: 17cfa86f-c374-4ca3-8cd6-f638a5234fe7 |
|       | request_errors: 0                                    |
|       | total_connections: 0                                 |
+-------+------------------------------------------------------+
```

This command shows statistics for the load balancer `mylbaas`.

vinfra service compute load-balancer set

Modify a load balancer:

```
usage: vinfra service compute load-balancer set [--description <description>]
                                                [--enable | --disable]
                                                [--name <name>] <load-balancer>
```

`--description <description>`

Load balancer description

`--enable`

Enable the load balancer.

`--disable`

Disable the load balancer.

`--name <name>`

Load balancer name

`<load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer set mylbaas --disable \
--description "Disabled load balancer"
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| address       | 192.168.30.230                       |
| amphorae      |                                      |
| created_at    | 2019-11-18T12:59:08.243413           |
| description   | Disabled load balancer               |
| enabled       | False                                |
| floating_ip   |                                      |
| ha_enabled    |                                      |
| id            | 941bf637-2d55-40f0-92c0-e65d6567b468 |
| members_count | 0                                    |
| name          | mylbaas                              |
| network_id    | 2b821d00-e428-4a76-b1ae-d181c9f5ae7f |
| pools         | []                                   |
| port_id       | 2d8ab88a-847c-4396-857e-11eaa80e1b24 |
| project_id    | e4e059c67dee4736851df14d4519a5a5     |
| status        | DISABLED                             |
| updated_at    | 2019-11-18T13:09:09.151442           |
+---------------+--------------------------------------+
```

This command disables the load balancer `mylbaas` and adds a description to it.

vinfra service compute load-balancer pool create

Create a load balancer pool:

```
usage: vinfra service compute load-balancer pool create --protocol
                                                        {HTTP,HTTPS,TCP,UDP}
                                                        --port <port>
                                                        --algorithm <algorithm>
                                                        --backend-protocol
                                                        {HTTP,HTTPS,TCP,UDP}
                                                        --backend-port <backend-port>
                                                        [--certificate-file <cert-file>]
                                                        [--connection-limit <limit>]
                                                        [--description <description>]
                                                        [--healthmonitor type=<type>,
                                                        url_path=<url>[,key=value,…]]
                                                        [--member address=<ip>
                                                        [,enabled=<bool>, weight=<int>]]
                                                        [--privatekey-file <key>]
                                                        [--enable-sticky-session |
                                                        --disable-sticky-session]
                                                        [--enable | --disable]
                                                        [--name <name>] <load-balancer>
```

`--protocol {HTTP,HTTPS,TCP,UDP}`

The protocol for incoming connections

`--port <port>`

The port for incoming connections

`--algorithm <algorithm>`

Load balancing algorithm (`LEAST_CONNECTIONS`, `ROUND_ROBIN`, or `SOURCE_IP`)

`--backend-protocol {HTTP,HTTPS,TCP,UDP}`

The protocol for destination connections

`--backend-port <backend-port>`

The port for destination connections

`--certificate-file <cert-file>`

An x.509 certificate file in the PEM format. Required for TLS-terminated HTTPS->HTTP load balancers.

`--connection-limit <limit>`

The maximum number of connections permitted for this pool. The default value is -1 (infinite connections).

`--description <description>`

Pool description

`--healthmonitor type=<type>,url_path=<url>[,key=value,…]`

Health monitor parameters:

- `type`: the health monitor type (`HTTP`, `HTTPS`, `PING`, `TCP`, or `UDP`)
- `url_path`: the URL path to the health monitor
- comma-separated `key=value` pairs with keys (optional):
  - `delay`: the time, in seconds, between sending probes to members.
  - `enabled`: declares whether the health monitor is enabled or not (`true` or `false`).
  - `max_retries`: the number of successful checks required to change member status to 'HEALTHY'. Ranges from 1 to 10.
  - `max_retries_down`: the number of unsuccessful checks required to change member status to 'UNHEALTHY'. Ranges from 1 to 10.
  - `timeout`: the maximum time, in seconds, that a monitor waits to connect before it times out. This value must be less than the `delay` value.

`--member address=<ip>[,enabled=<bool>,weight=<int>]`

Member parameters:

- `address=<ip>`: an IPv4 address of the virtual machine
- `enabled=<bool>`: declares whether the member is enabled or not. Can be 'true' or 'false'.
- `weight=<int>`: determines the share of connections that the member services compared to the other pool members. For example, a weight of 10 means that the member handles five times as many connections than a member with a weight of 2. A weight of 0 means that the member does not receive new connections but continues to service existing ones. Ranges from 0 to 256. The default value is 1. This option can be used multiple times.

`--privatekey-file <key>`

A private TLS key file in the PEM format. Required for TLS-terminated HTTPS->HTTP load balancers.

`--enable-sticky-session`

Enable session persistence.

`--disable-sticky-session`

Disable session persistence.

`--enable`

Enable the pool.

`--disable`

Disable the pool.

`--name <name>`

Pool name

`<load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer pool create mylbaas --protocol HTTP \
--port 80 --backend-protocol HTTP --backend-port 80 --name mypool \
--algorithm LEAST_CONNECTIONS --member address=192.168.31.153 \
--member address=192.168.31.22 --enable-sticky-session
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| backend_protocol     | HTTP                                 |
| backend_protocol_port | 80                                   |
| certificate          |                                      |
| connection_limit     | -1                                   |
| created_at           | 2019-11-18T13:11:27.982129           |
| description          |                                      |
| enabled              | True                                 |
| healthmonitor        |                                      |
| id                   | fa40e282-b29a-465a-afaa-2c702d2bde17 |
| lb_algorithm         | LEAST_CONNECTIONS                    |
| listener_id          | 66cc714e-af7f-40eb-9db8-67b8b6b6d23c |
| loadbalancer_id      | 941bf637-2d55-40f0-92c0-e65d6567b468 |
| members              | []                                   |
| name                 | mypool                               |
| private_key          |                                      |
| project_id           | e4e059c67dee4736851df14d4519a5a5     |
| protocol             | HTTP                                 |
| protocol_port        | 80                                   |
| status               | CREATING                             |
| sticky_session       | True                                 |
| updated_at           |                                      |
+----------------------+--------------------------------------+
```

This command adds a balancing pool `mypool` to the load balancer `mylbaas` with the following parameters:

- “HTTP on port 80 -> HTTP on port 80” forwarding rule
- the `LEAST_CONNECTIONS` balancing algorithm
- two members in the pool
- enabled sticky session

vinfra service compute load-balancer pool list

List load balancer pools:

```
usage: vinfra service compute load-balancer pool list [--long]
                                                      [--load-balancer
                                                      <load-balancer>]
```

`--long`

Enable access and listing of all fields of objects.

`--load-balancer <load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer pool list --load-balancer mylbaas -c id \
-c protocol -c protocol_port -c backend_protocol -c backend_protocol_port -c status
+-----------+----------+---------------+--------------+--------------+--------+
| id        | protocol | protocol_port | backend<...> | backend<...> | status |
+-----------+----------+---------------+--------------+--------------+--------+
| fa40<...> | HTTP     |            80 | HTTP         |           80 | ACTIVE |
+-----------+----------+---------------+--------------+--------------+--------+
```

This command lists load balancer pools of the load balancer `mylbaas`.

vinfra service compute load-balancer pool show

Display load balancer pool details:

```
usage: vinfra service compute load-balancer pool show <pool>
```

`<pool>`

Load balancer pool ID or name

Example:

```
# vinfra service compute load-balancer pool show mypool
+----------------------+------------------------------------------+
| Field                | Value                                    |
+----------------------+------------------------------------------+
| backend_protocol     | HTTP                                     |
| backend_protocol_port | 80                                      |
| certificate          |                                          |
| connection_limit     | -1                                       |
| created_at           | 2019-11-18T13:11:27.982129               |
| description          |                                          |
| enabled              | True                                     |
| healthmonitor        |                                          |
| id                   | fa40e282-b29a-465a-afaa-2c702d2bde17     |
| lb_algorithm         | LEAST_CONNECTIONS                        |
| listener_id          | 66cc714e-af7f-40eb-9db8-67b8b6b6d23c     |
| loadbalancer_id      | 941bf637-2d55-40f0-92c0-e65d6567b468     |
| members              | - address: 192.168.31.153                |
|                      |   compute_server_id: d51c10a7-6187-<...> |
|                      |   created_at: '2019-11-18T13:11:59.681101' |
|                      |   enabled: true                          |
|                      |   id: 3fd5dcc5-6e2c-4e22-8d0a-8e94e20a122f |
|                      |   name: ''                               |
|                      |   pool_id: null                          |
|                      |   status: HEALTHY                        |
```

```
|                       |   updated_at: '2019-11-18T13:12:01.467306' |
|                       |   weight: 1                                |
|                       | - address: 192.168.31.22                   |
|                       |   compute_server_id: 54603109-8963-<...>   |
|                       |   created_at: '2019-11-18T13:12:10.176853' |
|                       |   enabled: true                            |
|                       |   id: ccb645b3-63c7-44f8-b861-b197c85506d4 |
|                       |   name: ''                                 |
|                       |   pool_id: null                            |
|                       |   status: HEALTHY                          |
|                       |   updated_at: '2019-11-18T13:12:12.281578' |
|                       |   weight: 1                                |
| name                  | mypool                                     |
| private_key           |                                            |
| project_id            | e4e059c67dee4736851df14d4519a5a5           |
| protocol              | HTTP                                       |
| protocol_port         | 80                                         |
| status                | ACTIVE                                     |
| sticky_session        | True                                       |
| updated_at            | 2019-11-18T13:12:12.305509                 |
+-----------------------+--------------------------------------------+
```

This command shows the details of the load balancer pool `mypool`.

vinfra service compute load-balancer pool set

Modify a load balancer pool:

```
usage: vinfra service compute load-balancer pool set [--name <name>
                                                      --protocol {HTTP,HTTPS,TCP,UDP}]
                                                      [--port <port>
                                                      --algorithm <algorithm>]
                                                      [--backend-protocol
                                                      {HTTP,HTTPS,TCP,UDP}]
                                                      [--backend-port <backend-port>]
                                                      [--certificate-file <cert-file>]
                                                      [--connection-limit <limit>]
                                                      [--description <description>]
                                                      [--healthmonitor type=<type>,
                                                      url_path=<url>[,key=value,…]]
                                                      [--member address=<ip>
                                                      [,enabled=<bool>,weight=<int>]]
                                                      [--privatekey-file <key>]
                                                      [--enable-sticky-session |
                                                      --disable-sticky-session]
                                                      [--enable | --disable]  <pool>
```

`--name <name>`

Pool name

`--protocol {HTTP,HTTPS,TCP,UDP}`

The protocol for incoming connections

`--port <port>`

The port for incoming connections

`--algorithm <algorithm>`

Load balancing algorithm (`LEAST_CONNECTIONS`, `ROUND_ROBIN`, or `SOURCE_IP`)

`--backend-protocol {HTTP,HTTPS,TCP,UDP}`

The protocol for destination connections

`--backend-port <backend-port>`

The port for destination connections

`--certificate-file <cert-file>`

An x.509 certificate file in the PEM format. Required for TLS-terminated HTTPS->HTTP load balancers.

`--connection-limit <limit>`

The maximum number of connections permitted for this pool. The default value is -1 (infinite connections).

`--description <description>`

Pool description

`--healthmonitor type=<type>,url_path=<url>[,key=value,…]`

Health monitor parameters:

- `type`: the health monitor type (`HTTP`, `HTTPS`, `PING`, `TCP`, or `UDP`)
- `url_path`: the URL path to the health monitor
- comma-separated `key=value` pairs with keys (optional):
  - `delay`: the time, in seconds, between sending probes to members.
  - `enabled`: declares whether the health monitor is enabled or not (`true` or `false`).
  - `max_retries`: the number of successful checks required to change member status to 'HEALTHY'. Ranges from 1 to 10.
  - `max_retries_down`: the number of unsuccessful checks required to change member status to 'UNHEALTHY'. Ranges from 1 to 10.
  - `timeout`: the maximum time, in seconds, that a monitor waits to connect before it times out. This value must be less than the `delay` value.

`--member address=<ip>[,enabled=<bool>,weight=<int>]`

Member parameters:

- `address=<ip>`: an IPv4 address of the virtual machine
- `enabled=<bool>`: declares whether the member is enabled or not. Can be 'true' or 'false'.
- `weight=<int>`: determines the share of connections that the member services compared to the other pool members. For example, a weight of 10 means that the member handles five times as many connections than a member with a weight of 2. A weight of 0 means that the

member does not receive new connections but continues to service existing ones. Ranges from 0 to 256. The default value is 1. This option can be used multiple times.

`--privatekey-file <key>`

A private TLS key file in the PEM format. Required for TLS-terminated HTTPS->HTTP load balancers.

`--enable-sticky-session`

Enable session persistence.

`--disable-sticky-session`

Disable session persistence.

`--enable`

Enable the pool.

`--disable`

Disable the pool.

`<pool>`

Load balancer pool ID or name

Example:

```
# vinfra service compute load-balancer pool set mypool --algorithm ROUND_ROBIN \
--member address=192.168.31.153 --member address=192.168.31.22 \
--member address=192.168.31.51 --disable-sticky-session
Operation accepted.
```

This command changes the parameters for the balancing pool `mypool` as follows:

- sets the balancing algorithm to `ROUND_ROBIN`
- adds the third member to the pool
- disables sticky session

vinfra service compute load-balancer pool delete

Delete a load balancer pool:

```
usage: vinfra service compute load-balancer pool delete <pool>
```

`<pool>`

Load balancer pool ID or name

Example:

```
# vinfra service compute load-balancer pool delete mypool
Operation successful.
```

This command removes the load balancer pool `mypool`.

vinfra service compute load-balancer failover

Perform a failover of a load balancer:

```
usage: vinfra service compute load-balancer failover <load-balancer>
```

`<load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer failover mylbaas
Operation accepted.
```

This command performs a failover of the load balancer `mylbaas`.

vinfra service compute load-balancer delete

Delete a load balancer:

```
usage: vinfra service compute load-balancer delete <load-balancer>
```

`<load-balancer>`

Load balancer ID or name

Example:

```
# vinfra service compute load-balancer delete mylbaas
Operation accepted.
```

This command deletes the load balancer `mylbaas`.

Managing volumes

vinfra service compute volume create

Create a new compute volume:

```
usage: vinfra service compute volume create [--description <description>]
                                            [--network-install <network_install>]
                                            [--image <image>]
                                            [--snapshot <snapshot>]
                                            --storage-policy <storage_policy>
                                            --size <size-gb> <volume-name>
```

`--description <description>`

Volume description

`--network-install <network_install>`

Perform network installation (`true` or `false`).

`--image <image>`

Source compute image ID or name

`--snapshot <snapshot>`

Source compute volume snapshot ID or name

`--storage-policy <storage_policy>`

Storage policy ID or name

`--size <size-gb>`

Volume size, in gigabytes

`<volume-name>`

Volume name

Example:

```
# vinfra service compute volume create myvolume --storage-policy default --size 8
+--------------------------------+--------------------------------------+
| Field                          | Value                                |
+--------------------------------+--------------------------------------+
| attachments                    | []                                   |
| availability_zone              | nova                                 |
| bootable                       | False                                |
| consistencygroup_id            |                                      |
| created_at                     | 2018-09-12T12:30:12.665916           |
| description                    |                                      |
| encrypted                      | False                                |
| id                             | c9c0e9e7-ce7a-4566-99d5-d7e40f2987ab |
| imageRef                       |                                      |
| migration_status               |                                      |
| multiattach                    | False                                |
| name                           | myvolume                             |
| network_install                | False                                |
| os-vol-host-attr:host          |                                      |
| os-vol-mig-status-attr:migstat |                                      |
| os-vol-mig-status-attr:name_id |                                      |
| project_id                     | 72a5db3a033c403a86756021e601ef34     |
| replication_status             |                                      |
| size                           | 8                                    |
| snapshot_id                    |                                      |
| source_volid                   |                                      |
| status                         | creating                             |
| storage_policy_name            | default                              |
| updated_at                     |                                      |
| user_id                        | 98bf389983c24c07af9677b931783143     |
| volume_image_metadata          |                                      |
+--------------------------------+--------------------------------------+
```

This command creates a volume `myvolume` sized 8 GB and chooses the default storage policy for it.

vinfra service compute volume list

List compute volumes:

```
usage: vinfra service compute volume list [--long] [--limit <num>]
                                          [--marker <volume>] [--name <name>]
                                          [--id <id>] [--project <project>]
                                          [--status <status>] [--size <size>]
                                          [--storage-policy <host>]
```

`--long`

Enable access and listing of all fields of objects.

`--limit <num>`

The maximum number of volumes to list. To list all volumes, set the option to -1.

`--marker <volume>`

List volumes after the marker.

`--name <name>`

List volumes with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a volume with the specified ID or list volumes using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--project <project>`

List volumes that belong to the specified project ID. Can only be performed by system administrators.

`--status <status>`

List volumes with the specified status.

`--size <size>`

List volumes with the specified size.

`--storage-policy <host>`

List volumes with the specified storage policy name or ID.

Example:

```
# vinfra service compute volume list -c id -c name -c size -c status
+--------------------------------------+----------+------+-----------+
| id                                   | name     | size | status    |
+--------------------------------------+----------+------+-----------+
```

```
| c9c0e9e7-ce7a-4566-99d5-d7e40f2987ab | myvolume |    8 | available |
+--------------------------------------+----------+------+-----------+
```

This command lists volumes available to the compute cluster.

vinfra service compute volume show

Display compute volume details:

```
usage: vinfra service compute volume show <volume>
```

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute volume show myvolume
+-------------------------------+------------------------------------------+
| Field                         | Value                                    |
+-------------------------------+------------------------------------------+
| attachments                   | []                                       |
| availability_zone             | nova                                     |
| bootable                      | False                                    |
| consistencygroup_id           |                                          |
| created_at                    | 2018-09-12T12:30:12.665916               |
| description                   |                                          |
| encrypted                     | False                                    |
| id                            | c9c0e9e7-ce7a-4566-99d5-d7e40f2987ab     |
| imageRef                      |                                          |
| migration_status              |                                          |
| multiattach                   | False                                    |
| name                          | myvolume                                 |
| network_install               | False                                    |
| os-vol-host-attr:host         | node001.vstoragedomain@vstorage#vstorage |
| os-vol-mig-status-attr:migstat |                                          |
| os-vol-mig-status-attr:name_id |                                          |
| project_id                    | 72a5db3a033c403a86756021e601ef34         |
| replication_status            |                                          |
| size                          | 8                                        |
| snapshot_id                   |                                          |
| source_volid                  |                                          |
| status                        | available                                |
| storage_policy_name           | default                                  |
| updated_at                    | 2018-09-12T12:30:33.167654               |
| user_id                       | 98bf389983c24c07af9677b931783143         |
| volume_image_metadata         |                                          |
+-------------------------------+------------------------------------------+
```

This command shows the details for the volume `myvolume`.

vinfra service compute volume set

Modify volume parameters:

```
usage: vinfra service compute volume set [--description <description>]
                                         [--network-install <network_install>]
                                         [--storage-policy <storage_policy>]
                                         [--bootable <bootable>]
                                         [--name <name>] [--no-placements]
                                         <volume>
```

`--description <description>`

Volume description

`--network-install <network_install>`

Perform network install (`true` or `false`)

`--storage-policy <storage_policy>`

Storage policy ID or name

`--bootable <bootable>`

Make bootable (`true` or `false`)

`--name <name>`

A new name for the volume

`--no-placements`

Clean up placements

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute volume set myvolume --storage-policy mystorpolicy
+-------------------------------+------------------------------------------+
| Field                         | Value                                    |
+-------------------------------+------------------------------------------+
| attachments                   | []                                       |
| availability_zone             | nova                                     |
| bootable                      | False                                    |
| consistencygroup_id           |                                          |
| created_at                    | 2018-09-12T12:30:12.665916               |
| description                   |                                          |
| encrypted                     | False                                    |
| id                            | c9c0e9e7-ce7a-4566-99d5-d7e40f2987ab     |
| imageRef                      |                                          |
| migration_status              |                                          |
| multiattach                   | False                                    |
| name                          | myvolume                                 |
```

```
| network_install                | False                                      |
| os-vol-host-attr:host          | node001.vstoragedomain@vstorage#vstorage |
| os-vol-mig-status-attr:migstat |                                            |
| os-vol-mig-status-attr:name_id |                                            |
| project_id                     | 72a5db3a033c403a86756021e601ef34         |
| replication_status             |                                            |
| size                           | 8                                          |
| snapshot_id                    |                                            |
| source_volid                   |                                            |
| status                         | available                                  |
| storage_policy_name            | mystorpolicy                               |
| updated_at                     | 2018-09-12T12:55:29.298717                 |
| user_id                        | 98bf389983c24c07af9677b931783143         |
| volume_image_metadata          |                                            |
+--------------------------------+--------------------------------------------+
```

This command changes the storage policy of the volume `myvolume` to `mystorpolicy`.

vinfra service compute volume extend

Extend a compute volume:

```
usage: vinfra service compute volume extend --size <size_gb> <volume>
```

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute volume extend myvolume --size 16
Operation successful
```

This command extends the volume `myvolume` to 16 GB.

vinfra service compute volume delete

Delete a compute volume:

```
usage: vinfra service compute volume delete <volume>
```

`<volume>`

Volume ID or name

Example:

```
# vinfra service compute volume delete myvolume2
Operation successful
```

This command deletes the volume `myvolume2`.

Managing volume snapshots

vinfra service compute volume snapshot create

Create a snapshot of a volume:

```
usage: vinfra service compute volume snapshot create [--description
                                                      <description>]
                                                      --volume <volume>
                                                      <volume-snapshot-name>
```

`--description <description>`

Volume snapshot description

`--volume <volume>`

Volume ID or name

`<volume-snapshot-name>`

Volume snapshot name

Example:

```
# vinfra service compute volume snapshot create mysnapshot --volume myvolume
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2019-04-30T13:12:54.297629+00:00     |
| description |                                      |
| id          | 3fdfe5d6-8bd2-4bf5-8599-a9cef50e5b71 |
| metadata    | {}                                   |
| name        | mysnapshot                           |
| project_id  | fd0ae61496d04ef6bb637bc3167b7eaf     |
| size        | 8                                    |
| status      | creating                             |
| volume_id   | 92dc3bd7-713d-42bf-83cd-4de40c24fed9 |
+-------------+--------------------------------------+
```

This command initiates creation of a snapshot `mysnapshot` of the volume `myvolume`.

vinfra service compute volume snapshot list

List volume snapshots:

```
usage: vinfra service compute volume snapshot list [--long] [--volume <volume>]
```

`--long`

Enable access and listing of all fields of objects.

`--volume <volume>`

Volume ID or name

Example:

```
# vinfra service compute volume snapshot list -c id -c name -c size -c status
+--------------------------------------+------------+-----------+
| id                                   | name       | status    |
+--------------------------------------+------------+-----------+
| 3fdfe5d6-8bd2-4bf5-8599-a9cef50e5b71 | mysnapshot | available |
+--------------------------------------+------------+-----------+
```

This command lists volume snapshots available to the compute cluster.

vinfra service compute volume snapshot show

Display details of a volume snapshot:

```
usage: vinfra service compute volume snapshot show <volume-snapshot>
```

`<volume-snapshot>`

Volume snapshot ID or name

Example:

```
# vinfra service compute volume snapshot show mysnapshot
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2019-04-30T13:12:54.297629+00:00     |
| description |                                      |
| id          | 3fdfe5d6-8bd2-4bf5-8599-a9cef50e5b71 |
| metadata    | {}                                   |
| name        | mysnapshot                           |
| project_id  | fd0ae61496d04ef6bb637bc3167b7eaf     |
| size        | 8                                    |
| status      | available                            |
| volume_id   | 92dc3bd7-713d-42bf-83cd-4de40c24fed9 |
+-------------+--------------------------------------+
```

This command shows the details for the volume snapshot `mysnapshot`.

vinfra service compute volume snapshot set

Modify volume snapshot parameters:

```
usage: vinfra service compute volume snapshot set [--description <description>]
                                                  [--name <name>]
                                                  <volume-snapshot>
```

`--description <description>`

Volume snapshot description

`--name <name>`

A new name for the volume snapshot

`<volume-snapshot>`

Volume snapshot ID or name

Example:

```
# vinfra service compute volume snapshot set mysnapshot --name mynewsnapshot
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2019-04-30T13:12:54.297629+00:00     |
| description |                                      |
| id          | 3fdfe5d6-8bd2-4bf5-8599-a9cef50e5b71 |
| metadata    | {}                                   |
| name        | mynewsnapshot                        |
| project_id  | fd0ae61496d04ef6bb637bc3167b7eaf     |
| size        | 8                                    |
| status      | available                            |
| volume_id   | 92dc3bd7-713d-42bf-83cd-4de40c24fed9 |
+-------------+--------------------------------------+
```

This command changes the name of the volume snapshot `mysnapshot` to `mynewsnapshot`.

vinfra service compute volume snapshot upload-to-image

Create a compute image from a compute volume snapshot:

```
usage: vinfra service compute volume snapshot upload-to-image [--name <name>]
                                                              <volume-snapshot>
```

`--name <name>`

Image name

`<volume-snapshot>`

Volume snapshot ID or name

Example:

```
# vinfra service compute volume snapshot upload-to-image --name myvm-image \
mynewsnapshot
+------------------+--------------------------------------+
| Field            | Value                                |
+------------------+--------------------------------------+
| checksum         |                                      |
| container_format | bare                                 |
```



```
| created_at       |                                      |
| disk_format      | qcow2                                |
| id               | 6a7a78c1-7168-4387-9b55-23fd477fdaa0 |
| min_disk         |                                      |
| min_ram          |                                      |
| name             | myvm-image                           |
| os_distro        | linux                                |
| os_type          | linux                                |
| project_id       |                                      |
| protected        | False                                |
| public           | False                                |
| size             | 1                                    |
| status           | uploading                            |
| tags             |                                      |
| updated_at       | 2019-06-07T12:30:43.462707           |
| virtual_size     |                                      |
+------------------+--------------------------------------+
```

This command creates the compute image `myvm-image` from the volume snapshot `mynewsnapshot`.

vinfra service compute volume snapshot revert

Revert a volume to the specified snapshot:

```
usage: vinfra service compute volume snapshot revert <volume-snapshot>
```

`<volume-snapshot>`

Volume snapshot ID or name

Example:

```
# vinfra service compute volume snapshot revert mynewsnapshot
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2019-04-30T13:12:54.297629+00:00     |
| description |                                      |
| id          | 3fdfe5d6-8bd2-4bf5-8599-a9cef50e5b71 |
| metadata    | {}                                   |
| name        | mynewsnapshot                        |
| project_id  | fd0ae61496d04ef6bb637bc3167b7eaf     |
| size        | 8                                    |
| status      | available                            |
| volume_id   | 92dc3bd7-713d-42bf-83cd-4de40c24fed9 |
+-------------+--------------------------------------+
```

This command reverts the volume to its snapshot `mynewsnapshot`.

vinfra service compute volume snapshot reset-state

Reset a volume snapshot stuck in the “Error” state or one of transitional states to the “Available” state:

```
usage: vinfra service compute volume snapshot reset-state <volume-snapshot>
```

`<volume-snapshot>`

Volume snapshot ID or name

Example:

```
# vinfra service compute volume snapshot reset-state mynewsnapshot
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2019-04-30T13:12:54.297629+00:00     |
| description |                                      |
| id          | 3fdfe5d6-8bd2-4bf5-8599-a9cef50e5b71 |
| metadata    | {}                                   |
| name        | mynewsnapshot                        |
| project_id  | fd0ae61496d04ef6bb637bc3167b7eaf     |
| size        | 8                                    |
| status      | available                            |
| volume_id   | 92dc3bd7-713d-42bf-83cd-4de40c24fed9 |
+-------------+--------------------------------------+
```

This command resets the state of the volume snapshot `mynewsnapshot`.

vinfra service compute volume snapshot delete

Delete a volume snapshot:

```
usage: vinfra service compute volume snapshot delete <volume-snapshot>
```

`<volume-snapshot>`

Volume snapshot ID or name

Example:

```
# vinfra service compute volume snapshot delete mynewsnapshot
Operation successful
```

This command deletes the volume snapshot `mynewsnapshot`.

Managing storage policies

You can manage storage policies only after creating the compute cluster.

vinfra service compute storage-policy create

Create a new storage policy:

```
usage: vinfra service compute storage-policy create --tier {0,1,2,3}
                                                    (--replicas <norm>[:<min>] |
                                                    --encoding <M>+<N>)
                                                    --failure-domain
                                                    {disk,host,rack,row,room}
                                                    [--total-bytes-sec <bytes>]
                                                    [--total-iops-sec <iops>]
                                                    <name>
```

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

`--replicas <norm>[:<min>]`

Storage replication mapping in the format:

- `norm`: the number of replicas to maintain
- `min`: the minimum required number of replicas (optional)

`--total-bytes-sec <bytes>`

Total bytes per second

`--total-iops-sec <iops>`

Total iops

`<name>`

Storage policy name

Example:

```
# vinfra service compute storage-policy create mystorpolicy --tier 3 \
--encoding 3+2 --failure-domain host --total-bytes-sec \
104857600 --total-iops-sec 100
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| available      | False                                |
```

```
| failure_domain | host                                 |
| id             | 2199e71e-ce8a-4ba9-81cd-75502f0344ca |
| name           | mystorpolicy                         |
| qos            | total_bytes_sec: 104857600           |
|                | total_iops_sec: 100                  |
| redundancy     | encoding=3+2                         |
| tier           | 3                                    |
+----------------+--------------------------------------+
```

This command creates a storage policy `mystorpolicy` with the tier set to 3, redundancy scheme to erasure coding 3+2, and failure domain set to host. It also sets the limits of 100 iops and 104857600 bytes per second.

vinfra service compute storage-policy list

List existing storage policies:

```
usage: vinfra service compute storage-policy list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute storage-policy list
+-----------+----------+------+-------------+-----------+---------------------+
| id        | name     | tier | redundancy   | fail<...> | qos                 |
+-----------+----------+------+-------------+-----------+---------------------+
| 2199<...> | mypolicy |    3 | encoding=3+2 | host      | total_bytes_sec:    |
|           |          |      |              |           | 104857600           |
|           |          |      |              |           | total_iops_sec: 100 |
| 4274<...> | default  |    0 | replicas=3   | host      | total_bytes_sec: -1 |
|           |          |      |              |           | total_iops_sec: -1  |
+-----------+----------+------+-------------+-----------+---------------------+
```

This command lists storage policies available to the compute cluster.

vinfra service compute storage-policy show

Show details of a storage policy:

```
usage: vinfra service compute storage-policy show <storage-policy>
```

`<storage-policy>`

Storage policy ID or name

Example:


```
# vinfra service compute storage-policy show mystorpolicy
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| available      | False                                |
| failure_domain | host                                 |
| id             | 2199e71e-ce8a-4ba9-81cd-75502f0344ca |
| name           | mystorpolicy                         |
| qos            | total_bytes_sec: 104857600           |
|                | total_iops_sec: 100                  |
| redundancy     | encoding=3+2                         |
| tier           | 3                                    |
+----------------+--------------------------------------+
```

This command shows the details of the storage policy `mystorpolicy`.

vinfra service compute storage-policy set

Modify storage policy parameters:

```
usage: vinfra service compute storage-policy set [--name <name>]
                                                 [--tier {0,1,2,3}]
                                                 [--replicas <norm>[:<min>] |
                                                 --encoding <M>+<N>]
                                                 [--failure-domain
                                                 {disk,host,rack,row,room}]
                                                 [--total-bytes-sec <bytes>]
                                                 [--total-iops-sec <iops>]
                                                 <storage-policy>
```

`--name <name>`

A new name for the storage policy

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

`--replicas <norm>[:<min>]`

Storage replication mapping in the format:

- `norm`: the number of replicas to maintain
- `min`: the minimum required number of replicas (optional)

`--total-bytes-sec <bytes>`

Total bytes per second

`--total-iops-sec <iops>`

Total iops

`<storage-policy>`

Storage policy ID or name

Example:

```
# vinfra service compute storage-policy set mystorpolicy --encoding 5+2
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| available      | False                                |
| failure_domain | host                                 |
| id             | 2199e71e-ce8a-4ba9-81cd-75502f0344ca |
| name           | mystorpolicy                         |
| qos            | total_bytes_sec: 104857600           |
|                | total_iops_sec: 100                  |
| redundancy     | encoding=5+2                         |
| tier           | 3                                    |
+----------------+--------------------------------------+
```

This command changes the redundancy type for the storage policy `mystorpolicy` from erasure coding 3+2 to 5+2.

vinfra service compute storage-policy delete

The default policy cannot be deleted.

Remove an existing storage policy:

```
usage: vinfra service compute storage-policy delete <storage-policy>
```

`<storage-policy>`

Storage policy ID or name

Example:

```
# vinfra service compute storage-policy delete mystorpolicy
Operation successful
```

This command deletes the storage policy `mystorpolicy`.

Managing Kubernetes clusters

vinfra service compute k8saas create

Create a new Kubernetes cluster:

```
usage: vinfra service compute k8saas create [--master-node-count <count>]
                                            [--node-count <count>]
                                            [--volume-storage-policy <policy>]
                                            [--kubernetes-version <version>]
                                            --master-flavor <flavor> --flavor
                                            <flavor> [--volume-size <size>]
                                            --external-network <network>
                                            [--network <network>]
                                            --key-name <key-name>
                                            [--use-floating-ip <use-floating-ip>]
                                            [--enable-public-access]
                                            [--containers-network-cidr <cidr>]
                                            [--containers-network-node-subnet-
                                            prefix-length <prefix_length>]
                                            [--service-network-cidr <cidr>]
                                            [--dns-service-ip <ip>] <name>
```

`<name>`

Kubernetes cluster name

`--master-node-count <count>`

The amount of master nodes in the Kubernetes cluster

`--node-count <count>`

The amount of worker nodes in the Kubernetes cluster

`--volume-storage-policy <policy>`

The name of the storage policy for the volume where containers will reside.

`--kubernetes-version <version>`

Kubernetes version (`v1.21.3`, `v1.20.7`, and `v1.19.9`)

`--master-flavor <flavor>`

The flavor to use for Kubernetes master nodes.

`--flavor <flavor>`

The flavor to use for Kubernetes worker nodes.

`--volume-size <size>`

The size of the storage volume on each Kubernetes node

`--external-network <network>`

The ID or name of a physical network that will provide Internet access to Kubernetes nodes.

`--network <netwlork>`

The ID or name of a virtual network that will provide networking between Kubernetes nodes.

`--key-name <key-name>`

The key pair to use for accessing the Kubernetes nodes.

`--use-floating-ip <use-floating-ip>`

Use floating IP addresses for all Kubernetes nodes (`true` or `false`).

`--enable-public-access`

Use a floating IP address for the Kubernetes API (`true` or `false`).

`--containers-network-cidr <cidr>`

Container network range in CIDR notation

`--containers-network-node-subnet-prefix-length <prefix_length>`

The prefix length of a container subnet allocated to each Kubernetes node

`--service-network-cidr <cidr>`

Kubernetes service network range in CIDR notation

`--dns-service-ip <ip>`

DNS service IP address

The prerequisites for creating a Kubernetes cluster are:

- The Kubernetes-as-a-service component. It can be deployed along with the compute cluster or later (refer to "Creating the compute cluster" or "Provisioning Kubernetes clusters" in the Administrator Guide).
- A virtual network that will interconnect Kubernetes nodes. It needs to have a gateway and a DNS server specified.
- An SSH key that will be installed on both the master and worker nodes.
- Enough resources for all of the Kubernetes nodes, taking their flavors into account.

Example:

```
# vinfra service compute k8saas create k8s1 --kubernetes-version v1.20.7 \
--master-node-count 1 --node-count 3 --volume-storage-policy default \
--master-flavor medium --volume-size 10 --external-network public \
--network private1 --flavor small --key-name key1 --use-floating-ip true \
--vinfra-username user1 --vinfra-password password --vinfra-domain domain1 \
--vinfra-project project1
+---------------------------------+------------------------------------------+
| Field                           | Value                                    |
+---------------------------------+------------------------------------------+
| action_status                   | CREATE_IN_PROGRESS                       |
| boot_volume_size                | 10                                       |
| boot_volume_storage_policy      | default                                  |
| containers_volume_size          | 10                                       |
| containers_volume_storage_policy | default                                  |
```



```
| create_timeout                  | 60                                          |
| external_network_id             | 10cc4d59-adac-4ec1-8e0a-df5015b82c64        |
| id                              | 749737ae-2452-4a98-a057-b59b1c579a85        |
| key_name                        | key1                                        |
| master_flavor                   | medium                                      |
| master_node_count               | 1                                           |
| name                            | k8s1                                        |
| network_id                      | d037623b-0db7-40c2-b38a-9ac34fbd1cc5        |
| nodegroups                      | - action_status: CREATE_IN_PROGRESS         |
|                                 |   flavor: medium                            |
|                                 |   id: c3b4ec41-b8c1-4dae-9e1c-aa586b99a62c  |
|                                 |   is_default: true                          |
|                                 |   name: default-master                      |
|                                 |   node_count: 1                             |
|                                 |   role: master                              |
|                                 |   status: CREATING                          |
|                                 |   version: v1.20.7                          |
|                                 | - action_status: CREATE_IN_PROGRESS         |
|                                 |   flavor: small                             |
|                                 |   id: 65b80f19-0920-48b7-84e0-d0c63c46e99f  |
|                                 |   is_default: true                          |
|                                 |   name: default-worker                      |
|                                 |   node_count: 3                             |
|                                 |   role: worker                              |
|                                 |   status: CREATING                          |
|                                 |   version: v1.20.7                          |
| project_id                      | d8a72d59539c431381989af6cb48b05d            |
| status                          | CREATING                                    |
| user_id                         | 5846f988280f42199ed030a22970d48e            |
| worker_pools                    | - flavor: small                             |
|                                 |   node_count: 3                             |
+---------------------------------+---------------------------------------------+
```

This command, run as `user1` from `domain1` > `project1`, starts creation of the Kubernetes cluster `k8s1` with these parameters:

- Kubernetes version 1.20.7
- 1 master node based on the `medium` flavor and 3 worker nodes based on the `small` flavor
- 10 GB storage volumes covered by the default storage policy
- Virtual network `private1` that will connect to the Internet via the physical network `public`
- Floating IP addresses for each node picked from the specified physical network
- Public SSH key `key1`

vinfra service compute k8saas list

List Kubernetes clusters:

```
usage: vinfra service compute k8saas list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute k8saas list
+--------------------------------------+------+--------+
| id                                   | name | status |
+--------------------------------------+------+--------+
| 749737ae-2452-4a98-a057-b59b1c579a85 | k8s1 | ACTIVE |
+--------------------------------------+------+--------+
```

This command displays the list of Kubernetes clusters.

vinfra service compute k8saas config

Print Kubernetes cluster configuration (must be run as the user who created the Kubernetes cluster):

```
usage: vinfra service compute k8saas config <cluster>
```

`<cluster>`

Cluster ID or name

Example:

```
# vinfra service compute k8saas config k8s1 --vinfra-domain domain1 \
--vinfra-project project1 --vinfra-username user1 --vinfra-password password \
> kubeconfig
```

This command prints the configuration of the Kubernetes cluster `k8s1` to the file `kubeconfig`.

vinfra service compute k8saas show

Display Kubernetes cluster details:

```
usage: vinfra service compute k8saas show <cluster>
```

`<cluster>`

Cluster ID or name

Example:

```
# vinfra service compute k8saas show k8s1
+---------------------------------+------------------------------------------+
| Field                           | Value                                    |
+---------------------------------+------------------------------------------+
| action_status                   | CREATE_COMPLETE                          |
```

```
| boot_volume_size                | 10                                          |
| boot_volume_storage_policy      | default                                     |
| containers_volume_size          | 10                                          |
| containers_volume_storage_policy | default                                    |
| create_timeout                  | 60                                          |
| external_network_id             | 10cc4d59-adac-4ec1-8e0a-df5015b82c64        |
| id                              | 749737ae-2452-4a98-a057-b59b1c579a85        |
| key_name                        | key1                                        |
| master_flavor                   | medium                                      |
| master_node_count               | 1                                           |
| name                            | k8s1                                        |
| network_id                      | d037623b-0db7-40c2-b38a-9ac34fbd1cc5        |
| nodegroups                      | - action_status: CREATE_COMPLETE            |
|                                 |   flavor: medium                            |
|                                 |   id: c3b4ec41-b8c1-4dae-9e1c-aa586b99a62c  |
|                                 |   is_default: true                          |
|                                 |   name: default-master                      |
|                                 |   node_count: 1                             |
|                                 |   role: master                              |
|                                 |   status: ACTIVE                            |
|                                 |   version: v1.20.7                          |
|                                 | - action_status: CREATE_COMPLETE            |
|                                 |   flavor: small                             |
|                                 |   id: 65b80f19-0920-48b7-84e0-d0c63c46e99f  |
|                                 |   is_default: true                          |
|                                 |   name: default-worker                      |
|                                 |   node_count: 3                             |
|                                 |   role: worker                              |
|                                 |   status: ACTIVE                            |
|                                 |   version: v1.20.7                          |
| project_id                      | d8a72d59539c431381989af6cb48b05d            |
| status                          | ACTIVE                                      |
| user_id                         | 5846f988280f42199ed030a22970d48e            |
| worker_pools                    | - flavor: small                             |
|                                 |   node_count: 3                             |
+---------------------------------+---------------------------------------------+
```

This command displays the details of the Kubernetes cluster `k8s1`.

vinfra service compute k8saas set

Modify Kubernetes cluster parameters (must be run as the user who created the Kubernetes cluster):

```
usage: vinfra service compute k8saas set [--node-count <count>]
                                         <cluster>
```

`<cluster>`

Cluster ID or name

`--node-count <count>`

The amount of worker nodes in the Kubernetes cluster

Example:

```
# vinfra service compute k8saas set --node-count 5 k8s1 \
--vinfra-domain domain1 --vinfra-project project1 \
--vinfra-username user1 --vinfra-password password
+---------------------------------+------------------------------------------+
| Field                           | Value                                    |
+---------------------------------+------------------------------------------+
| action_status                   | UPDATE_COMPLETE                          |
| boot_volume_size                | 10                                       |
| boot_volume_storage_policy      | default                                  |
| containers_volume_size          | 10                                       |
| containers_volume_storage_policy | default                                  |
| create_timeout                  | 60                                       |
| external_network_id             | 10cc4d59-adac-4ec1-8e0a-df5015b82c64     |
| id                              | 749737ae-2452-4a98-a057-b59b1c579a85     |
| key_name                        | key1                                     |
| master_flavor                   | medium                                   |
| master_node_count               | 1                                        |
| name                            | k8s1                                     |
| network_id                      | d037623b-0db7-40c2-b38a-9ac34fbd1cc5     |
| nodegroups                      | - action_status: UPDATE_COMPLETE         |
|                                 |   flavor: medium                         |
|                                 |   id: c3b4ec41-b8c1-4dae-9e1c-aa586b99a62c |
|                                 |   is_default: true                       |
|                                 |   name: default-master                   |
|                                 |   node_count: 1                          |
|                                 |   role: master                           |
|                                 |   status: ACTIVE                         |
|                                 |   version: v1.20.7                       |
|                                 | - action_status: UPDATE_COMPLETE         |
|                                 |   flavor: small                          |
|                                 |   id: 65b80f19-0920-48b7-84e0-d0c63c46e99f |
|                                 |   is_default: true                       |
|                                 |   name: default-worker                   |
|                                 |   node_count: 5                          |
|                                 |   role: worker                           |
|                                 |   status: ACTIVE                         |
|                                 |   version: v1.20.7                       |
| project_id                      | d8a72d59539c431381989af6cb48b05d         |
| status                          | ACTIVE                                   |
| user_id                         | 5846f988280f42199ed030a22970d48e         |
| worker_pools                    | - flavor: small                          |
|                                 |   node_count: 5                          |
+---------------------------------+------------------------------------------+
```

This command starts configuring the Kubernetes cluster `k8s1` to bring the number of worker nodes to 5.

vinfra service compute k8saas workergroup create

Create a new Kubernetes worker group:

```
usage: vinfra service compute k8saas workergroup create --flavor <flavor>
                                                        [--node-count <count>]
                                                        <cluster> <name>
```

`<cluster>`

Cluster ID or name

`<name>`

Kubernetes worker group name

`--flavor <flavor>`

The flavor to be used for Kubernetes worker group

`--node-count <count>`

The amount of worker nodes in the Kubernetes worker group

Example:

```
# vinfra service compute k8saas workergroup create k8s1 mygroup \
--flavor small --node-count 3 --vinfra-domain domain1 \
--vinfra-project project1 --vinfra-username user1 \
--vinfra-password password
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| flavor     | small                                |
| id         | 70d071eb-7a81-471f-ae50-99758ae27678 |
| is_default | False                                |
| name       | mygroup                              |
| node_count | 3                                    |
| role       | worker                               |
| status     | CREATING                             |
+------------+--------------------------------------+
```

This command starts creating the worker group `mygroup` with 3 nodes for the Kubernetes cluster `k8s1`.

vinfra service compute k8saas workergroup list

List Kubernetes worker groups:

```
usage: vinfra service compute k8saas workergroup list [--long] <cluster>
```

`cluster`

Cluster ID or name

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service compute k8saas workergroup list k8s1\
--vinfra-domain domain1 --vinfra-project project1 \
--vinfra-username user1 --vinfra-password password
+--------------------------------------+----------------+----------+
| id                                   | name           | status   |
+--------------------------------------+----------------+----------+
| efa32b7d-55d7-4a16-a765-68950404ec45 | default-master | ACTIVE   |
| 54242833-3416-474e-9651-a266d62e0962 | default-worker | ACTIVE   |
| 70d071eb-7a81-471f-ae50-99758ae27678 | mygroup        | CREATING |
+--------------------------------------+----------------+----------+
```

This command displays all of the node groups for the Kubernetes cluster `k8s1`.

vinfra service compute k8saas workergroup show

Display Kubernetes worker group details:

```
usage: vinfra service compute k8saas workergroup show <cluster> <worker-group>
```

`<cluster>`

Cluster ID or name

`<worker-group>`

Worker group ID or name

Example:

```
# vinfra service compute k8saas workergroup show k8s1 mygroup \
--vinfra-domain domain1 --vinfra-project project1 \
--vinfra-username user1 --vinfra-password password
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| flavor          | small                                |
| id              | 70d071eb-7a81-471f-ae50-99758ae27678 |
| is_default      | False                                |
| name            | mygroup                              |
| node_count      | 3                                    |
| role            | worker                               |
| server_group_id | 50f4ae08-4e44-4132-a40b-7043a2c3e739 |
| status          | ACTIVE                               |
+-----------------+--------------------------------------+
```

This command displays the details of the worker group `mygroup` for the Kubernetes cluster `k8s1`.

vinfra service compute k8saas workergroup set

Modify Kubernetes worker group parameters:

```
usage: vinfra service compute k8saas workergroup set [--node-count <count>]
                                                     <cluster> <worker-group>
```

`<cluster>`

Cluster ID or name

`<worker-group>`

Worker group ID or name

`--node-count <count>`

The amount of worker nodes in the Kubernetes worker group

Example:

```
# vinfra service compute k8saas workergroup set k8s1 mygroup \
--node-count 5 --vinfra-domain domain1 --vinfra-project project1 \
--vinfra-username user1 --vinfra-password password
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| flavor          | small                                |
| id              | 70d071eb-7a81-471f-ae50-99758ae27678 |
| is_default      | False                                |
| name            | mygroup                              |
| node_count      | 3                                    |
| role            | worker                               |
| server_group_id | 50f4ae08-4e44-4132-a40b-7043a2c3e739 |
| status          | ACTIVE                               |
+-----------------+--------------------------------------+
```

This command starts configuring the worker group `mygroup` for the Kubernetes cluster `k8s1` to bring the number of worker nodes to 5.

vinfra service compute k8saas workergroup delete

Delete a Kubernetes worker group:

```
usage: vinfra service compute k8saas workergroup delete <cluster> <worker-group>
```

`<cluster>`

Cluster ID or name

`<worker-group>`

Worker group ID or name

Example:

```
# vinfra service compute k8saas workergroup delete k8s1 mygroup \
--vinfra-domain domain1 --vinfra-project project1 \
--vinfra-username user1 --vinfra-password password
Operation accepted.
```

This command deletes the worker group `mygroup` for the Kubernetes cluster `k8s1`.

vinfra service compute k8saas upgrade

Display Kubernetes cluster details:

```
usage: vinfra service compute k8saas upgrade <cluster> <version>
```

`<cluster>`

Cluster ID or name

`<version>`

Kubernetes version (`v1.21.3`, `v1.20.7`, and `v1.19.9`)

Example:

```
# vinfra service compute k8saas upgrade k8s1 v1.19.9
+----------------------------------+--------------------------------------+
| Field                            | Value                                |
+----------------------------------+--------------------------------------+
| action_status                    | CREATE_COMPLETE                      |
| boot_volume_size                 | 10                                   |
| boot_volume_storage_policy       | default                              |
| containers_volume_size           | 10                                   |
| containers_volume_storage_policy | default                              |
| create_timeout                   | 60                                   |
| external_network_id              | c51e1250-6ecb-4224-b623-ee11351a8758 |
| floating_ip_enabled              | False                                |
| id                               | c481f3af-f076-4a42-b7aa-283e74f35919 |
| key_name                         | key1                                 |
| master_flavor                    | medium                               |
| master_node_count                | 1                                    |
| name                             | k8s1                                 |
| network_id                       | 422d511c-2fa8-4642-98f1-55135502c203 |
| nodegroups                       | - action_status: CREATE_COMPLETE     |
|                                  |   flavor: small                      |
|                                  |   id: eccc37e3-a6f2-41f8-9b5c<…>     |
|                                  |   is_default: true                   |
|                                  |   name: default-worker               |
|                                  |   node_count: 1                      |
|                                  |   role: worker                       |
|                                  |   server_group_id: 1d6a8e92-cb0c<…>  |
|                                  |   status: ACTIVE                     |
```

```
|                                  |   version: v1.18.6                    |
|                                  | - action_status: CREATE_COMPLETE      |
|                                  |   flavor: medium                      |
|                                  |   id: 8826ca85-c5ec-4d55-abf6<…>      |
|                                  |   is_default: true                    |
|                                  |   name: default-master                |
|                                  |   node_count: 1                       |
|                                  |   role: master                        |
|                                  |   server_group_id: 39d4a051-ba1b<…>   |
|                                  |   status: ACTIVE                      |
|                                  |   version: v1.18.6                    |
| project_id                       | 75521ab61d1f4e9090aac5836c219492      |
| public_access_enabled            | True                                  |
| stack_id                         | 2826d603-4d17-472b-ab24-611ee909a941  |
| status                           | ACTIVE                                |
| updated_at                       | 2021-06-24T11:26:19.041602+00:00      |
| user_id                          | 9021894a6eb44a2898f69eb518f43bd3      |
| version                          | v1.18.6                               |
| worker_pools                     | - flavor: small                       |
|                                  |   node_count: 1                       |
+----------------------------------+---------------------------------------+
```

This command upgrades the Kubernetes cluster `k8s1` to version 1.19.9.

vinfra service compute k8saas rotate-ca

Rotate Kubernetes cluster CA certificates:

```
usage: vinfra service compute k8saas rotate-ca <cluster>
```

`<cluster>`

Cluster ID or name

Example:

```
# vinfra service compute k8saas rotate-ca k8s1
+----------------------------------+------------------------------------------+
| Field                            | Value                                    |
+----------------------------------+------------------------------------------+
| action_status                    | CREATE_COMPLETE                          |
| boot_volume_size                 | 10                                       |
| boot_volume_storage_policy       | default                                  |
| containers_volume_size           | 10                                       |
| containers_volume_storage_policy | default                                  |
| create_timeout                   | 60                                       |
| external_network_id              | 10cc4d59-adac-4ec1-8e0a-df5015b82c64     |
| id                               | 749737ae-2452-4a98-a057-b59b1c579a85     |
| key_name                         | key1                                     |
| master_flavor                    | medium                                   |
| master_node_count                | 1                                        |
```

```
| name                              | k8s1                                          |
| network_id                        | d037623b-0db7-40c2-b38a-9ac34fbd1cc5          |
| nodegroups                        | - action_status: CREATE_COMPLETE              |
|                                   |   flavor: medium                              |
|                                   |   id: c3b4ec41-b8c1-4dae-9e1c-aa586b99a62c    |
|                                   |   is_default: true                            |
|                                   |   name: default-master                        |
|                                   |   node_count: 1                               |
|                                   |   role: master                                |
|                                   |   status: ACTIVE                              |
|                                   |   version: v1.20.7                            |
|                                   | - action_status: CREATE_COMPLETE              |
|                                   |   flavor: small                               |
|                                   |   id: 65b80f19-0920-48b7-84e0-d0c63c46e99f    |
|                                   |   is_default: true                            |
|                                   |   name: default-worker                        |
|                                   |   node_count: 5                               |
|                                   |   role: worker                                |
|                                   |   status: ACTIVE                              |
|                                   |   version: v1.20.7                            |
| project_id                        | d8a72d59539c431381989af6cb48b05d              |
| status                            | ACTIVE                                        |
| user_id                           | 5846f988280f42199ed030a22970d48e              |
| worker_pools                      | - flavor: small                               |
|                                   |   node_count: 5                               |
+-----------------------------------+-----------------------------------------------+
```

This command rotates CA certificates for the Kubernetes cluster `k8s1`.

vinfra service compute k8saas delete

Delete a Kubernetes cluster:

```
usage: vinfra service compute k8saas delete <cluster>
```

`<cluster>`

Cluster ID or name

Example:

```
# vinfra service compute k8saas delete k8s1
Operation accepted.
```

This command deletes the Kubernetes cluster `k8s1`.

Managing compute quotas

vinfra service compute quotas show

List compute quotas:

```
usage: vinfra service compute quotas show [--usage] <project_id>
```

`--usage`

Include quota usage.

`<project_id>`

Project ID

Example:

```
# vinfra service compute quotas show 6ef6f48f01b640ccb8ff53117b830fa3 --usage
+---------------------------------------+---------+
| Field                                 | Value   |
+---------------------------------------+---------+
| compute.cores.limit                   | 20      |
| compute.cores.used                    | 2       |
| compute.ram.limit                     | 10.0GiB |
| compute.ram.used                      | 1.0GiB  |
| compute.ram_quota.limit               | 10.0GiB |
| compute.ram_quota.used                | 1.0GiB  |
| k8saas.cluster.limit                  | 10      |
| k8saas.cluster.used                   | 0       |
| lbaas.loadbalancer.limit              | 10      |
| lbaas.loadbalancer.used               | 0       |
| network.floatingip.limit              | 10      |
| network.floatingip.used               | 0       |
| storage.gigabytes.default.limit       | 1.0TiB  |
| storage.gigabytes.default.used        | 2.0GiB  |
| storage.storage_policies.default.limit | 1.0TiB  |
| storage.storage_policies.default.used  | 2.0GiB  |
+---------------------------------------+---------+
```

This command shows compute quotas with their usage for the project with the ID `6ef6f48f01b640ccb8ff53117b830fa3`.

vinfra service compute quotas update

Update compute quotas:

```
usage: vinfra service compute quotas update [--cores <cores>] [--ram-size <ram>]
                                            [--floatingip <floating-ip>]
                                            [--storage-policy
```

```
                                        <storage_policy>:<size>]
                                        [--k8saas-cluster <cluster>]
                                        [--lbaas-loadbalancer <load-balancer>]
                                        [--placement <placement>] <project-id>
```

`--cores <cores>`

Number of cores

`--ram-size <ram>`

Number of RAM. Use the following units: `M` or `MiB` for mebibytes, `G` or `GiB` for gibibytes, `T` or `TiB` for tebibytes, `P` or `PiB` for pebibytes, and `E` or `EiB` for exbibytes.

`--floatingip <floating-ip>`

Number of floating IP addresses

`--storage-policy <storage_policy>:<size>`

Comma-separated list of `<storage_policy>:<size>`. To specify the size, use the following units: `M` or `MiB` for mebibytes, `G` or `GiB` for gibibytes, `T` or `TiB` for tebibytes, `P` or `PiB` for pebibytes, and `E` or `EiB` for exbibytes.

`--k8saas-cluster <cluster>`

Number of Kubernetes clusters

`--lbaas-loadbalancer <load-balancer>`

The new value for the load balancer quota limit. The value `-1` means unlimited.

`--placement <placement>`

Comma-separated list of `<placement-id>:<size>`

`<project-id>`

Project ID

Example:

```
# vinfra service compute quotas update 6ef6f48f01b640ccb8ff53117b830fa3 \
--cores 10 --ram-size 10G --storage-policy default:512G
Operation successful.
```

This command updates compute quotas to 10 vCPUs, 20 GiB of RAM, and 512 GiB of disk space for the `default` storage policy.

Managing the backup cluster

Creating, showing, and deleting the backup cluster

vinfra service backup cluster create

Create the backup cluster:

```
usage: vinfra service backup cluster create --nodes <nodes> --domain <domain>
                                            --reg-account <reg-account>
                                            --reg-server <reg-server>
                                            --tier {0,1,2,3}
                                            --encoding <M>+<N>
                                            --failure-domain {0,1,2,3,4}
                                            --storage-type {local,nfs,s3,
                                            swift,azure,google} [--stdin]
                                            [--nfs-host <host>]
                                            [--nfs-export <export>]
                                            [--nfs-version <version>]
                                            [--s3-flavor <flavor>]
                                            [--s3-region <region>]
                                            [--s3-bucket <bucket>]
                                            [--s3-endpoint <endpoint>]
                                            [--s3-access-key-id <access-key-id>]
                                            [--s3-secret-key-id <secret-key-id>]
                                            [--s3-cert-verify <cert-verify>]
                                            [--swift-auth-url <auth-url>]
                                            [--swift-auth-version <auth-version>]
                                            [--swift-user-name <user-name>]
                                            [--swift-api-key <api-key>]
                                            [--swift-domain <domain>]
                                            [--swift-domain-id <domain-id>]
                                            [--swift-tenant <tenant>]
                                            [--swift-tenant-id <tenant-id>]
                                            [--swift-tenant-domain
                                            <tenant-domain>]
                                            [--swift-tenant-domain-id
                                            <tenant-domain-id>]
                                            [--swift-trust-id <trust-id>]
                                            [--swift-region <region>]
                                            [--swift-internal <internal>]
                                            [--swift-container <container>]
                                            [--swift-cert-verify <cert-verify>]
                                            [--azure-endpoint <endpoint>]
                                            [--azure-container <container>]
                                            [--azure-account-name <account-name>]
                                            [--azure-account-key <account-key>]
                                            [--google-bucket <bucket>]
                                            [--google-credentials <credentials>]
```

`--nodes <nodes>`

A comma-separated list of node hostnames or IDs

`--domain <domain>`

Domain name for the backup cluster

`--reg-account <reg-account>`

Partner account in the cloud or of an organization administrator on the local management server

`--reg-server <reg-server>`

URL of the cloud management portal or the hostname/IP address and port of the local management server

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

`--storage-type {local,nfs,s3,swift,azure,google}`

Storage type

`--stdin`

Use for setting registration password from `stdin`.

Storage parameters for the `nfs` storage type:

`--nfs-host <host>`

NFS hostname or IP address

`--nfs-export <export>`

Full path to the NFS export

`--nfs-version <version>`

NFS version (`3` or `4`)

Storage parameters for the `s3` storage type:

`--s3-flavor <flavor>` (optional)

Flavor name

`--s3-region <region>` (optional)

Set region for Amazon S3.

`--s3-bucket <bucket>`

Bucket name

`--s3-endpoint <endpoint>`

Endpoint URL

`--s3-access-key-id <access-key-id>`

Access key ID

`--s3-secret-key-id <secret-key-id>`

Secret key ID

`--s3-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the S3 endpoint

Storage parameters for the `swift` storage type:

`--swift-auth-url <auth-url>`

Authentication (keystone) URL

`--swift-auth-version <auth-version>` (optional)

Authentication protocol version

`--swift-user-name <user-name>`

User name

`--swift-api-key <api-key>`

API key (password)

`--swift-domain <domain>` (optional)

Domain name

`--swift-domain-id <domain-id>` (optional)

Domain ID

`--swift-tenant <tenant>` (optional)

Tenant name

`--swift-tenant-id <tenant-id>` (optional)

Tenant ID

`--swift-tenant-domain <tenant-domain>` (optional)

Tenant domain name

`--swift-tenant-domain-id <tenant-domain-id>` (optional)

Tenant domain ID

`--swift-trust-id <trust-id>` (optional)

Trust ID

`--swift-region <region>` (optional)

Region name

`--swift-container <container>` (optional)

Container name

`--swift-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the Swift endpoint (`true` or `false`)

Storage parameters for the `azure` storage type:

`--azure-endpoint <endpoint>`

Endpoint URL

`--azure-container <container>`

Container name

`--azure-account-name <account-name>`

Account name

`--azure-account-key <account-key>`

Account key

Storage parameters for the `google` storage type:

`--google-bucket <bucket>`

Google bucket name

`--google-credentials <credentials>`

Path to the file with Google credentials

Example 1. Creating the backup cluster on the local storage:

```
# vinfra service backup cluster create --nodes 2f3f6091-0d44-45aa-94e3-\
ebc2b65c0eeb,74cbd22b-fb1b-4441-ae52-532078c54f9a,eeb06dce-4cfd-4c89-bc7f-\
4689ea5c7058 --storage-type local --domain dns.example.com --tier 0 \
--encoding 1+2 --failure-domain 1 --reg-account account@example.com \
--reg-server https://cloud.acronis.com/ --stdin
Password:
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | ee7e60c5-5447-4177-8581-26657ac380c0 |
+---------+--------------------------------------+
```

This command creates a task to create the backup cluster from three nodes specified by ID on the local storage. It also specifies the domain name, tier, failure domain, registration account and server.

Task outcome:

```
# vinfra task show ee7e60c5-5447-4177-8581-26657ac380c0
+---------+------------------------------------------------------+
| Field   | Value                                                |
```



```
+---------+-------------------------------------------------+
| details |                                                 |
| name    | backend.presentation.abgw.tasks.RegisterAbgwTask |
| result  |                                                 |
| state   | success                                         |
| task_id | ee7e60c5-5447-4177-8581-26657ac380c0            |
+---------+-------------------------------------------------+
```

Example 2. Creating the backup cluster on the S3 storage:

```
# vinfra service backup cluster create --nodes 2f3f6091-0d44-45aa-94e3-\
ebc2b65c0eeb,74cbd22b-fb1b-4441-ae52-532078c54f9a,eeb06dce-4cfd-4c89-bc7f-\
4689ea5c7058 --storage-type s3 --domain dns.example.com --tier 0 \
--encoding 1+2 --failure-domain host --s3-bucket mybucket \
--s3-endpoint s3.amazonaws.com --s3-access-key-id e302a06df8adbe9fAIF1 \
--s3-secret-key-id x1gXquRHQXuyiUJQoQMoAohA2TkYHer20o8tfPX7 \
--s3-cert-verify true --reg-account account@example.com --reg-server \
https://cloud.acronis.com/ --stdin
Password:
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 0fb53a6f-2bc4-410a-aa1c-b3cda6ca8570 |
+---------+--------------------------------------+
```

This command creates a task to create the backup cluster from three nodes specified by ID on the S3 storage. It also specifies the domain name, tier, failure domain, registration account and server, as well as the required S3 parameters.

Task outcome:

```
# vinfra task show 0fb53a6f-2bc4-410a-aa1c-b3cda6ca8570
+---------+-------------------------------------------------+
| Field   | Value                                           |
+---------+-------------------------------------------------+
| details |                                                 |
| name    | backend.presentation.abgw.tasks.RegisterAbgwTask |
| result  |                                                 |
| state   | success                                         |
| task_id | 0fb53a6f-2bc4-410a-aa1c-b3cda6ca8570            |
+---------+-------------------------------------------------+
```

Example 3. Creating the backup cluster on the NFS storage:

```
# vinfra service backup cluster create --nodes eeb06dce-4cfd-4c89-bc7f-\
4689ea5c7058 --storage-type nfs --domain dns.example.com --tier 0 \
--encoding 1+2 --failure-domain host --nfs-host nfs.example.com --nfs-version 4 \
--nfs-export /myshare/myexport --reg-account account@example.com \
--reg-server https://cloud.acronis.com/ --stdin
Password:
```

```
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | d76ceb22-48e7-4eac-b04f-03d3aa3377b7 |
+---------+--------------------------------------+
```

This command creates a task to create the backup cluster from one node with the ID `eeb06dce-4cfd-4c89-bc7f-4689ea5c7058` on the NFS storage. It also specifies the domain name, tier, failure domain, registration account and server, as well as the required NFS parameters.

Task outcome:

```
# vinfra task show d76ceb22-48e7-4eac-b04f-03d3aa3377b7
+---------+-----------------------------------------------------+
| Field   | Value                                               |
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.presentation.abgw.tasks.RegisterAbgwTask |
| result  |                                                     |
| state   | success                                             |
| task_id | d76ceb22-48e7-4eac-b04f-03d3aa3377b7             |
+---------+-----------------------------------------------------+
```

vinfra service backup cluster show

Display backup cluster details:

```
usage: vinfra service backup cluster show
```

Example:

```
# vinfra service backup cluster show
+----------------+-------------------------------------------------------------+
| Field          | Value                                                       |
+----------------+-------------------------------------------------------------+
| abgw_address   | dns.example.com                                             |
| account_server | https://cloud.acronis.com                                   |
| dc_uid         | 44893a40296ecd9ae64567297a5b2b07-1577203369                 |
| migration      | dns: null                                                   |
|                | ips: []                                                     |
|                | running: false                                              |
|                | time_left: 0.0                                              |
| reg_type       | abc                                                         |
| storage_params | access_key_id: e302a06df8adbe9fAIF1                         |
|                | bucket: mybucket                                            |
|                | cert_verify: true                                           |
|                | endpoint: s3.amazonaws.com                                  |
|                | flavour: null                                               |
|                | region: null                                                |
```

```
|                | secret_key_id: x1gXquRHQXuyiUJQoQMoAohA2TkYHer20o8tfPX7 |
| storage_type   | s3                                                      |
+----------------+---------------------------------------------------------+
```

This command shows the domain name, registration details, and storage parameters of the backup cluster.

vinfra service backup cluster release

Delete the backup cluster and all its data:

```
usage: vinfra service backup cluster release [--reg-account <reg-account>]
                                             [--force] [--stdin]
```

`--reg-account <reg-account>`

Partner account in the cloud or of an organization administrator on the local management server

`--force`

Release the backup cluster but does not unregister it from your backup software.

Note
Choose this option only if you are sure that the cluster has already been unregistered from your backup software.

`--stdin`

Use for setting registration password from `stdin`.

Example:

```
# vinfra service backup cluster release --reg-account account@example.com --stdin
Password:
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | cf270233-06d5-4a4a-8dea-443d6fb59b10 |
+---------+--------------------------------------+
```

This command creates a task to delete the backup cluster with all its data and unregister it from your backup software.

Task outcome:

```
# vinfra task show cf270233-06d5-4a4a-8dea-443d6fb59b10
+---------+------------------------------------------------+
| Field   | Value                                          |
+---------+------------------------------------------------+
| details |                                                |
```

```
| name    | backend.presentation.abgw.tasks.ReleaseNodesTask |
| result  |                                                  |
| state   | success                                          |
| task_id | cf270233-06d5-4a4a-8dea-443d6fb59b10             |
+---------+--------------------------------------------------+
```

Managing backup nodes

vinfra service backup node add

Add a list of nodes to the backup cluster:

```
usage: vinfra service backup node add --nodes <nodes>
```

`--nodes <nodes>`

A comma-separated list of node hostnames or IDs

Example:

```
# vinfra service backup node add --nodes 2f3f6091-0d44-45aa-94e3-ebc2b65c0eeb
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | affe92f4-0c01-4a06-b91b-4ee0355d9a87 |
+---------+--------------------------------------+
```

This command creates a task to add the node with the ID `2f3f6091-0d44-45aa-94e3-ebc2b65c0eeb` to the backup cluster.

Task outcome:

```
# vinfra task show affe92f4-0c01-4a06-b91b-4ee0355d9a87
+---------+--------------------------------------------------+
| Field   | Value                                            |
+---------+--------------------------------------------------+
| details |                                                  |
| name    | backend.presentation.abgw.tasks.AssignNodesTask  |
| result  |                                                  |
| state   | success                                          |
| task_id | affe92f4-0c01-4a06-b91b-4ee0355d9a87             |
+---------+--------------------------------------------------+
```

vinfra service backup node list

List nodes in the backup cluster:

```
usage: vinfra service backup node list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra service backup node list
+--------------------------------------+------------------------+-----------+
| id                                   | host                   | is_online |
+--------------------------------------+------------------------+-----------+
| 2f3f6091-0d44-45aa-94e3-ebc2b65c0eeb | node003.vstoragedomain | True      |
| 74cbd22b-fb1b-4441-ae52-532078c54f9a | node001.vstoragedomain | True      |
| eeb06dce-4cfd-4c89-bc7f-4689ea5c7058 | node002.vstoragedomain | True      |
+--------------------------------------+------------------------+-----------+
```

This command lists nodes in the backup cluster.

vinfra service backup node release

Release a list of nodes from the backup cluster:

```
usage: vinfra service backup node release --nodes <nodes>
```

`--nodes <nodes>`

A comma-separated list of node hostnames or IDs

Example:

```
# vinfra service backup node release --nodes 2f3f6091-0d44-45aa-94e3-ebc2b65c0eeb
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | ea09642c-291c-4df8-87a5-a8958d6308c1 |
+---------+--------------------------------------+
```

This command creates a task to release the node with the ID `2f3f6091-0d44-45aa-94e3-ebc2b65c0eeb` from the backup cluster.

Task outcome:

```
# vinfra task show ea09642c-291c-4df8-87a5-a8958d6308c1
+---------+-----------------------------------------------------+
| Field   | Value                                               |
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.presentation.abgw.tasks.ReleaseNodesTask |
| result  |                                                     |
| state   | success                                             |
| task_id | ea09642c-291c-4df8-87a5-a8958d6308c1             |
+---------+-----------------------------------------------------+
```

Updating backup cluster certificates

Update certificates for the backup cluster:

```
usage: vinfra service backup cluster renew-certificates --reg-account <reg-account>
                                                        --reg-server <reg-server>
                                                        [--stdin]
```

`--stdin`

Use for setting registration password from `stdin`.

`--reg-account <reg-account>`

Partner account in the cloud or of an organization administrator on the local management server

`--reg-server <reg-server>`

URL of the cloud management portal or the hostname/IP address and port of the local management server

Example:

```
# vinfra service backup cluster renew-certificates --reg-account account@example.com \
--reg-server https://cloud.acronis.com/ --stdin
Password:
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 7f1873a7-cd9b-49f3-ae17-fb14ff08ddf5 |
+---------+--------------------------------------+
```

This command creates a task to update certificates for the backup cluster.

Task outcome:

```
# vinfra task show 7f1873a7-cd9b-49f3-ae17-fb14ff08ddf5
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.presentation.abgw.tasks.RenewRegistrationAbgwTask |
| result  |                                                            |
| state   | success                                                    |
| task_id | 7f1873a7-cd9b-49f3-ae17-fb14ff08ddf5                       |
+---------+------------------------------------------------------------+
```

Re-registering the backup cluster

Re-register the backup cluster:

```
usage: vinfra service backup cluster re-register --domain <domain>
                                                 --reg-account <reg-account>
                                                 --reg-server <reg-server>
                                                 [--stdin]
```

`--domain <domain>`

Domain name for the backup cluster

`--reg-account <reg-account>`

Partner account in the cloud or of an organization administrator on the local management server

`--reg-server <reg-server>`

URL of the cloud management portal or the hostname/IP address and port of the local management server

`--stdin`

Use for setting registration password from `stdin`.

Example:

```
# vinfra service backup cluster re-register --domain newdns.example.com \
--reg-account account@example.com --reg-server https://cloud.acronis.com/ --stdin
Password:
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | de9e743d-7335-8246-6215-83455ff824e5 |
+---------+--------------------------------------+
```

This command creates a task to re-register the backup cluster in a new Acronis Cyber Protect instance.

Task outcome:

```
# vinfra task show ee7e60c5-5447-4177-8581-26657ac380c0
+---------+-----------------------------------------------------+
| Field   | Value                                               |
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.presentation.abgw.tasks.ReRegisterAbgwTask |
| result  |                                                     |
| state   | success                                             |
| task_id | de9e743d-7335-8246-6215-83455ff824e5                |
+---------+-----------------------------------------------------+
```

Changing storage parameters

vinfra service backup storage-params show

Display storage parameters:

```
usage: vinfra service backup storage-params show
```

Example:

```
# vinfra service backup storage-params show
+----------------+-------------------------------------------+
| Field          | Value                                     |
+----------------+-------------------------------------------+
| abgw_address   | dns.example.com                           |
| account_server | https://cloud.acronis.com                 |
| dc_uid         | 44893a40296ecd9ae64567297a5b2b07-1577264050 |
| migration      | dns: null                                 |
|                | ips: []                                   |
|                | running: false                            |
|                | time_left: 0.0                            |
| reg_type       | abc                                       |
| storage_params | export: /myshare/myexport                 |
|                | host: 10.94.129.70                        |
|                | version: 4                                |
| storage_type   | nfs                                       |
+----------------+-------------------------------------------+
```

This command shows the storage parameters of the backup cluster.

vinfra service backup storage-params change

Important

Change storage parameters with caution and only within the existing configuration. You can change the IP address of an external storage or its access credentials.

Modify storage parameters:

```
usage: vinfra service backup storage-params change --storage-type {local,nfs,
                                        s3,swift,azure,google}
                                        [--nfs-host <host>]
                                        [--nfs-export <export>]
                                        [--nfs-version <version>]
                                        [--s3-flavor <flavor>]
                                        [--s3-region <region>]
                                        [--s3-bucket <bucket>]
                                        [--s3-endpoint <endpoint>]
```

```
                                        [--s3-access-key-id <access-key-id>]
                                        [--s3-secret-key-id <secret-key-id>]
                                        [--s3-cert-verify <cert-verify>]
                                        [--swift-auth-url <auth-url>]
                                        [--swift-auth-version <auth-version>]
                                        [--swift-user-name <user-name>]
                                        [--swift-api-key <api-key>]
                                        [--swift-domain <domain>]
                                        [--swift-domain-id <domain-id>]
                                        [--swift-tenant <tenant>]
                                        [--swift-tenant-id <tenant-id>]
                                        [--swift-tenant-domain
                                        <tenant-domain>]
                                        [--swift-tenant-domain-id
                                        <tenant-domain-id>]
                                        [--swift-trust-id <trust-id>]
                                        [--swift-region <region>]
                                        [--swift-internal <internal>]
                                        [--swift-container <container>]
                                        [--swift-cert-verify <cert-verify>]
                                        [--azure-endpoint <endpoint>]
                                        [--azure-container <container>]
                                        [--azure-account-name <account-name>]
                                        [--azure-account-key <account-key>]
                                        [--google-bucket <bucket>]
                                        [--google-credentials <credentials>]
```

`--storage-type {local,nfs,s3,swift,azure,google}`

Storage type

Storage parameters for the `nfs` storage type:

`--nfs-host <host>`

NFS hostname or IP address

`--nfs-export <export>`

Full path to the NFS export

`--nfs-version <version>`

NFS version (`3` or `4`)

Storage parameters for the `s3` storage type:

`--s3-flavor <flavor>` (optional)

Flavor name

`--s3-region <region>` (optional)

Set region for Amazon S3.

`--s3-bucket <bucket>`

Bucket name

`--s3-endpoint <endpoint>`

Endpoint URL

`--s3-access-key-id <access-key-id>`

Access key ID

`--s3-secret-key-id <secret-key-id>`

Secret key ID

`--s3-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the S3 endpoint

Storage parameters for the `swift` storage type:

`--swift-auth-url <auth-url>`

Authentication (keystone) URL

`--swift-auth-version <auth-version>` (optional)

Authentication protocol version

`--swift-user-name <user-name>`

User name

`--swift-api-key <api-key>`

API key (password)

`--swift-domain <domain>` (optional)

Domain name

`--swift-domain-id <domain-id>` (optional)

Domain ID

`--swift-tenant <tenant>` (optional)

Tenant name

`--swift-tenant-id <tenant-id>` (optional)

Tenant ID

`--swift-tenant-domain <tenant-domain>` (optional)

Tenant domain name

`--swift-tenant-domain-id <tenant-domain-id>` (optional)

Tenant domain ID

`--swift-trust-id <trust-id>` (optional)

Trust ID

`--swift-region <region>` (optional)

Region name

`--swift-container <container>` (optional)

Container name

`--swift-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the Swift endpoint (`true` or `false`)

Storage parameters for the `azure` storage type:

`--azure-endpoint <endpoint>`

Endpoint URL

`--azure-container <container>`

Container name

`--azure-account-name <account-name>`

Account name

`--azure-account-key <account-key>`

Account key

Storage parameters for the `google` storage type:

`--google-bucket <bucket>`

Google bucket name

`--google-credentials <credentials>`

Path to the file with Google credentials

Example:

```
# vinfra service backup storage-params change --storage-type nfs --nfs-host \
10.94.129.71 --nfs-export /myshare/myexport --nfs-version 4
Operation successful.
```

This command changes the NFS storage parameters for the backup cluster.

Changing volume parameters

vinfra service backup volume-params show

Display volume parameters:

```
usage: vinfra service backup volume-params show
```

Example:

```
# vinfra service backup volume-params show
+----------------+-------------+
| Field          | Value       |
+----------------+-------------+
| failure_domain | host        |
| redundancy     | m: 1        |
```



```
|                | n: 2        |
|                | type: raid6 |
| tier           | 0           |
+----------------+-------------+
```

This command shows the volume parameters of the backup cluster: failure domain, redundancy scheme, and tier.

vinfra service backup volume-params change

Modify volume parameters:

```
usage: vinfra service backup volume-params change [--tier {0,1,2,3}]
                                                  [--encoding <M>+<N>]
                                                  [--failure-domain
                                                  {disk,host,rack,row,room}]
```

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

Example:

```
# vinfra service backup volume-params change --tier 1 --encoding 1+0 \
--failure-domain host
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 28ae19dc-51c9-49bf-bd93-51a763fa181b |
+---------+--------------------------------------+
```

This command creates a task to change volume parameters of the backup cluster as follows:

- The storage tier to 1
- The erasure coding scheme to 1+0
- The failure domain to host

Task outcome:

```
# vinfra task show 28ae19dc-51c9-49bf-bd93-51a763fa181b
+---------+------------------------------------------------------------+
```

```
| Field   | Value                                                    |
+---------+----------------------------------------------------------+
| details |                                                          |
| name    | backend.presentation.abgw.tasks.ChangeVolumeParamsTask |
| result  |                                                          |
| state   | success                                                  |
| task_id | 28ae19dc-51c9-49bf-bd93-51a763fa181b                     |
+---------+----------------------------------------------------------+
```

Managing backup cluster geo-replication

Important
To enable geo-replication, the backup clusters must be able to reach each other via domain names on TCP port 44445.

To enable geo-replication between two backup clusters, execute the following commands (refer to examples in the sections below):

1. On the cluster that will be configured as secondary, run `vinfra service backup geo-replication show` to learn its address and UID.
2. On the cluster that will be configured as primary, run `vinfra service backup geo-replication master setup`, using the address and UID of the secondary cluster.
3. On the primary cluster, run `vinfra service backup geo-replication master download-configs` to generate the configuration file of the primary cluster.
4. Move the configuration file of the primary cluster to the secondary cluster using the standard Linux command-line tool, for example, `scp`.
5. On the secondary cluster, run `vinfra service backup geo-replication slave setup` to upload the configuration file of the primary cluster.
6. On the primary cluster, run `vinfra service backup geo-replication master establish` to establish a connection between the primary and secondary clusters.
7. On one of the clusters, run `vinfra service backup geo-replication show` to check that geo-replication has been successfully enabled.

vinfra service backup geo-replication show

Display the geo-replication configuration:

```
usage: vinfra service backup geo-replication show
```

Example:

```
# vinfra service backup geo-replication show
+-------+-------------------------------------------------------------+
| Field | Value                                                       |
```

```
+-------+-------------------------------------------------------------+
| self  | address: slave.example.com                                  |
|       | datacenter_uid: e63a67388deb3c99d044eecbd7b79ad3-1577275849 |
+-------+-------------------------------------------------------------+
```

This command shows the geo-replication configuration of the secondary cluster.

vinfra service backup geo-replication master setup

Configure geo-replication for the primary cluster:

```
usage: vinfra service backup geo-replication master setup --slave-cluster-address
                                                          <slave-cluster-address>
                                                          --slave-cluster-uid
                                                          <slave-cluster-uid>
```

`--slave-cluster-address <slave-cluster-address>`

Secondary cluster DNS name

`--slave-cluster-uid <slave-cluster-uid>`

Secondary cluster UID

Example:

```
# vinfra service backup geo-replication master setup --slave-cluster-address \
slave.example.com --slave-cluster-uid e63a67388deb3c99d044eecbd7b79ad3-1577275849
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 07df4a57-704e-47de-b681-615ee0c26a21 |
+---------+--------------------------------------+
```

This command creates a task to configure geo-replication for the primary backup cluster.

Task outcome:

```
# vinfra task show 07df4a57-704e-47de-b681-615ee0c26a21
+---------+-----------------------------------------------------+
| Field   | Value                                               |
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                     |
| state   | success                                             |
| task_id | 07df4a57-704e-47de-b681-615ee0c26a21                |
+---------+-----------------------------------------------------+
```

vinfra service backup geo-replication master download-configs

Download the geo-replication configuration file of the primary cluster:


```
usage: vinfra service backup geo-replication master download-configs
                                                                    [--conf-file-path
                                                                    <conf-file-path>]
```

`--conf-file-path <conf-file-path>`

Path where the configuration file will be downloaded

Example:

```
# vinfra service backup geo-replication master download-configs \
--conf-file-path master_dc.conf
```

This command downloads the geo-replication configuration file of the primary cluster to the specified file.

vinfra service backup geo-replication slave setup

Configure geo-replication for the secondary cluster:

```
usage: vinfra service backup geo-replication slave setup --dc-config-file
                                                         <dc-config-file>
```

`--dc-config-file <dc-config-file>`

Path to the configuration file of the primary cluster on the local server

Example:

```
# vinfra service backup geo-replication slave setup \
--dc-config-file master_dc.conf
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | d34b3a4f-6e16-4e60-b20a-844052945d3e |
+---------+--------------------------------------+
```

This command creates a task to configure geo-replication for the secondary backup cluster.

Task outcome:

```
# vinfra task show d34b3a4f-6e16-4e60-b20a-844052945d3e
+---------+------------------------------------------------------+
| Field   | Value                                                |
+---------+------------------------------------------------------+
| details |                                                      |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                      |
| state   | success                                              |
| task_id | d34b3a4f-6e16-4e60-b20a-844052945d3e                 |
+---------+------------------------------------------------------+
```

vinfra service backup geo-replication master establish

Establish a connection between the primary and secondary clusters to enable geo-replication:

```
usage: vinfra service backup geo-replication master establish
```

Example:

```
# vinfra service backup geo-replication master establish
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 014903e4-c2e6-4e03-b1af-06c28b672f6e |
+---------+--------------------------------------+
```

This command creates a task to connect the primary and secondary clusters to enable geo-replication.

Task outcome:

```
# vinfra task show 014903e4-c2e6-4e03-b1af-06c28b672f6e
+---------+-------------------------------------------------------+
| Field   | Value                                                 |
+---------+-------------------------------------------------------+
| details |                                                       |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                       |
| state   | success                                               |
| task_id | 014903e4-c2e6-4e03-b1af-06c28b672f6e                  |
+---------+-------------------------------------------------------+
```

vinfra service backup geo-replication slave update-certificates

Update the primary cluster's configuration on the secondary cluster:

```
usage: vinfra service backup geo-replication slave update-certificates
                                                                  --dc-config-file
                                                                  <dc-config-file>
```

`--dc-config-file <dc-config-file>`

Path to the configuration file of the primary cluster

Example:

```
# vinfra service backup geo-replication slave update-certificates \
--dc-config-file primary_dc_updated.conf
+---------+--------------------------------------+
| Field   | Value                                |
```

```
+---------+--------------------------------------+
| task_id | 0ab89de1-b02d-426b-a03c-b1922e610594 |
+---------+--------------------------------------+
```

This command creates a task to update the configuration of the primary backup cluster.

Task outcome:

```
# vinfra task show 0ab89de1-b02d-426b-a03c-b1922e610594
+---------+------------------------------------------------------+
| Field   | Value                                                |
+---------+------------------------------------------------------+
| details |                                                      |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                      |
| state   | success                                              |
| task_id | 0ab89de1-b02d-426b-a03c-b1922e610594               |
+---------+------------------------------------------------------+
```

vinfra service backup geo-replication master disable

Disable geo-replication on the primary cluster:

```
usage: vinfra service backup geo-replication master disable
```

Example:

```
# vinfra service backup geo-replication master disable
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | dc2bb8ae-8e32-4d37-8d97-4c4c46189d27 |
+---------+--------------------------------------+
```

This command creates a task to disable geo-replication on the primary cluster.

Task outcome:

```
# vinfra task show dc2bb8ae-8e32-4d37-8d97-4c4c46189d27
+---------+------------------------------------------------------+
| Field   | Value                                                |
+---------+------------------------------------------------------+
| details |                                                      |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                      |
| state   | success                                              |
| task_id | dc2bb8ae-8e32-4d37-8d97-4c4c46189d27               |
+---------+------------------------------------------------------+
```

vinfra service backup geo-replication slave promote-to-master

Promote the secondary cluster to primary in the geo-replication configuration:

```
usage: vinfra service backup geo-replication slave promote-to-master
```

Example:

```
# vinfra service backup geo-replication slave promote-to-master
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 083a7d6e-3be8-490f-b468-a3f84abb3487 |
+---------+--------------------------------------+
```

This command creates a task to promote the secondary cluster to primary in the geo-replication configuration.

Task outcome:

```
# vinfra task show 083a7d6e-3be8-490f-b468-a3f84abb3487
+---------+-------------------------------------------------------+
| Field   | Value                                                 |
+---------+-------------------------------------------------------+
| details |                                                       |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                       |
| state   | success                                               |
| task_id | 083a7d6e-3be8-490f-b468-a3f84abb3487                  |
+---------+-------------------------------------------------------+
```

vinfra service backup geo-replication slave cancel

Cancel geo-replication for the secondary cluster:

```
usage: vinfra service backup geo-replication slave cancel
```

Example:

```
# vinfra service backup geo-replication slave cancel
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | ad977d03-995c-4677-9308-5e73ec8a2821 |
+---------+--------------------------------------+
```

This command creates a task to cancel geo-replication for the secondary backup cluster.

Task outcome:


```
# vinfra task show ad977d03-995c-4677-9308-5e73ec8a2821
+---------+-------------------------------------------------------+
| Field   | Value                                                 |
+---------+-------------------------------------------------------+
| details |                                                       |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                       |
| state   | success                                               |
| task_id | ad977d03-995c-4677-9308-5e73ec8a2821                  |
+---------+-------------------------------------------------------+
```

vinfra service backup geo-replication master cancel

Cancel geo-replication for the primary cluster:

```
usage: vinfra service backup geo-replication master cancel
```

Example:

```
# vinfra service backup geo-replication master cancel
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | e1931274-24a5-491e-a5f8-d24fdf4385f7 |
+---------+--------------------------------------+
```

This command creates a task to cancel geo-replication for the primary backup cluster.

Task outcome:

```
# vinfra task show e1931274-24a5-491e-a5f8-d24fdf4385f7
+---------+-------------------------------------------------------+
| Field   | Value                                                 |
+---------+-------------------------------------------------------+
| details |                                                       |
| name    | backend.tasks.message_dispatcher.CommandDispatcher |
| result  |                                                       |
| state   | success                                               |
| task_id | e1931274-24a5-491e-a5f8-d24fdf4385f7                  |
+---------+-------------------------------------------------------+
```

Configuring a backup storage proxy

By using backup storage in the reverse proxy mode, you can proxy backup data to other multiple backup clusters, which are called upstream. The minimum proxy configuration consists of one reverse proxy backup storage and one upstream backup storage. If you have more backup storage clusters, you can add them as upstream to the reverse proxy backup storage. The scenario below

describes a proxy configuration of three backup storage clusters: one reverse proxy backup storage and two upstream ones.

To configure a backup storage proxy, follow these steps (refer to examples in the sections below):

1. On the first storage cluster, deploy the standalone backup storage by running `vinfra service backup cluster deploy-standalone`.
2. On the first storage cluster, turn the standalone backup storage to an upstream backup storage with the command `vinfra service backup cluster turn-to-upstream`.
3. Change the DNS server configuration as follows:
   a. Delete the DNS name of the standalone storage from the DNS records.
   b. Add the DNS name of the upstream backup storage that resolves to the public IP addresses of its nodes.
4. On the second storage cluster, download the configuration file of the upstream backup storage by using `vinfra service backup cluster download-upstream-info`.
5. On the second storage cluster, deploy the reverse proxy backup storage and register the upstream backup storage with it. To do this, run `vinfra service backup cluster deploy-reverse-proxy`.
6. In the DNS records, add the first backup storage's DNS name that resolves to the public IP addresses of the reverse proxy nodes.
7. On the third storage cluster, deploy the upstream backup storage with the command `vinfra service backup cluster deploy-upstream`.
8. In the DNS records, add the DNS name of the new upstream backup storage that resolves to the public IP addresses of its nodes.
9. On the second storage cluster, download the configuration file of the new upstream backup storage by using `vinfra service backup cluster download-upstream-info`.
10. On the second storage cluster, register the new upstream backup storage with the reverse proxy by running `vinfra service backup cluster add-upstream`.

You can view each backup storage process and retry a suspended process with `vinfra service backup cluster process`.

Finally, your backup storage proxy configuration is complete.

vinfra service backup cluster deploy-standalone

Create the backup cluster:

```
usage: vinfra service backup cluster deploy-standalone --nodes <nodes>
                                            --domain <domain>
                                            --reg-account <reg-account>
                                            --reg-server <reg-server>
                                            --tier {0,1,2,3}
                                            --encoding <M>+<N>
                                            --failure-domain {0,1,2,3,4}
                                            --storage-type {local,nfs,s3,
```



```
                                        swift,azure,google} [--stdin]
                                        [--nfs-host <host>]
                                        [--nfs-export <export>]
                                        [--nfs-version <version>]
                                        [--s3-flavor <flavor>]
                                        [--s3-region <region>]
                                        [--s3-bucket <bucket>]
                                        [--s3-endpoint <endpoint>]
                                        [--s3-access-key-id <access-key-id>]
                                        [--s3-secret-key-id <secret-key-id>]
                                        [--s3-cert-verify <cert-verify>]
                                        [--swift-auth-url <auth-url>]
                                        [--swift-auth-version <auth-version>]
                                        [--swift-user-name <user-name>]
                                        [--swift-api-key <api-key>]
                                        [--swift-domain <domain>]
                                        [--swift-domain-id <domain-id>]
                                        [--swift-tenant <tenant>]
                                        [--swift-tenant-id <tenant-id>]
                                        [--swift-tenant-domain
                                        <tenant-domain>]
                                        [--swift-tenant-domain-id
                                        <tenant-domain-id>]
                                        [--swift-trust-id <trust-id>]
                                        [--swift-region <region>]
                                        [--swift-internal <internal>]
                                        [--swift-container <container>]
                                        [--swift-cert-verify <cert-verify>]
                                        [--azure-endpoint <endpoint>]
                                        [--azure-container <container>]
                                        [--azure-account-name <account-name>]
                                        [--azure-account-key <account-key>]
                                        [--google-bucket <bucket>]
                                        [--google-credentials <credentials>]
```

`--nodes <nodes>`

A comma-separated list of node hostnames or IDs

`--domain <domain>`

Domain name for the backup cluster

`--reg-account <reg-account>`

Partner account in the cloud or of an organization administrator on the local management server

`--reg-server <reg-server>`

URL of the cloud management portal or the hostname/IP address and port of the local management server

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

`--storage-type {local,nfs,s3,swift,azure,google}`

Storage type

`--stdin`

Use for setting registration password from `stdin`.

Storage parameters for the `nfs` storage type:

`--nfs-host <host>`

NFS hostname or IP address

`--nfs-export <export>`

Full path to the NFS export

`--nfs-version <version>`

NFS version (`3` or `4`)

Storage parameters for the `s3` storage type:

`--s3-flavor <flavor>` (optional)

Flavor name

`--s3-region <region>` (optional)

Set region for Amazon S3.

`--s3-bucket <bucket>`

Bucket name

`--s3-endpoint <endpoint>`

Endpoint URL

`--s3-access-key-id <access-key-id>`

Access key ID

`--s3-secret-key-id <secret-key-id>`

Secret key ID

`--s3-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the S3 endpoint

Storage parameters for the `swift` storage type:

`--swift-auth-url <auth-url>`

Authentication (keystone) URL

`--swift-auth-version <auth-version>` (optional)

Authentication protocol version

`--swift-user-name <user-name>`

User name

`--swift-api-key <api-key>`

API key (password)

`--swift-domain <domain>` (optional)

Domain name

`--swift-domain-id <domain-id>` (optional)

Domain ID

`--swift-tenant <tenant>` (optional)

Tenant name

`--swift-tenant-id <tenant-id>` (optional)

Tenant ID

`--swift-tenant-domain <tenant-domain>` (optional)

Tenant domain name

`--swift-tenant-domain-id <tenant-domain-id>` (optional)

Tenant domain ID

`--swift-trust-id <trust-id>` (optional)

Trust ID

`--swift-region <region>` (optional)

Region name

`--swift-container <container>` (optional)

Container name

`--swift-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the Swift endpoint (`true` or `false`)

Storage parameters for the `azure` storage type:

`--azure-endpoint <endpoint>`

Endpoint URL

`--azure-container <container>`

Container name

`--azure-account-name <account-name>`

Account name

`--azure-account-key <account-key>`

Account key

Storage parameters for the `google` storage type:

`--google-bucket <bucket>`

Google bucket name

`--google-credentials <credentials>`

Path to the file with Google credentials

Example:

```
# vinfra service backup cluster deploy-standalone \
--nodes 2f3f6091-0d44-45aa-94e3-ebc2b65c0eeb --storage-type local \
--domain backup1.example.com --tier 0 --encoding 1+0 \
--failure-domain 0 --reg-account account@example.com \
--reg-server https://cloud.acronis.com/ --stdin
Password:
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | f71687e5-243c-4ff9-81f5-f4f3e3560be8 |
+---------+--------------------------------------+
```

This command creates a task to create the standalone backup storage on the local storage.

Task outcome:

```
# vinfra task show f71687e5-243c-4ff9-81f5-f4f3e3560be8
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.business.models.abgw.standalone_deployment.<…> |
| result  |                                                            |
| state   | success                                                    |
| task_id | f71687e5-243c-4ff9-81f5-f4f3e3560be8                       |
+---------+------------------------------------------------------------+
```

vinfra service backup cluster turn-to-upstream

Turn the existing standalone backup storage to upstream:

```
usage: vinfra service backup cluster turn-to-upstream --address <address>
```

`--address <address>`

Address of the upstream backup storage

Example:

```
# vinfra service backup cluster turn-to-upstream --address upstream1.example.com
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| failed  | False                                |
| id      | 1d0d0e1a-8ccc-47e1-82a8-2210b19b1006 |
| message |                                      |
| state   | new                                  |
+---------+--------------------------------------+
```

This command turns the standalone backup storage to an upstream one.

vinfra service backup cluster download-upstream-info

Download information about the upstream backup storage:

```
usage: vinfra service backup cluster download-upstream-info [--output-file
                                                             <output-filepath>]
```

`--output-file <output-filepath>`

Path where the configuration file will be downloaded

Example:

```
# vinfra service backup cluster download-upstream-info \
--output-file /root/upstream1.info \
--vinfra-portal https://upstream1.example.com:8888 \
--vinfra-username admin --vinfra-password 1q2w3e
```

This command downloads the configuration file `/root/upstream1.info` from the upstream backup storage.

vinfra service backup cluster deploy-reverse-proxy

Create the reverse proxy backup storage:

```
usage: vinfra service backup cluster deploy-reverse-proxy --nodes <nodes>
                                          --tier {0,1,2,3}
                                          --encoding <M>+<N>
                                          --failure-domain {0,1,2,3,4}
                                          --storage-type {local,nfs,s3,
                                          swift,azure,google} [--stdin]
                                          --upstream-info-file
                                          <upstream-info-file>
                                          [--nfs-host <host>]
                                          [--nfs-export <export>]
                                          [--nfs-version <version>]
                                          [--s3-flavor <flavor>]
                                          [--s3-region <region>]
```

```
                                        [--s3-bucket <bucket>]
                                        [--s3-endpoint <endpoint>]
                                        [--s3-access-key-id <access-key-id>]
                                        [--s3-secret-key-id <secret-key-id>]
                                        [--s3-cert-verify <cert-verify>]
                                        [--swift-auth-url <auth-url>]
                                        [--swift-auth-version <auth-version>]
                                        [--swift-user-name <user-name>]
                                        [--swift-api-key <api-key>]
                                        [--swift-domain <domain>]
                                        [--swift-domain-id <domain-id>]
                                        [--swift-tenant <tenant>]
                                        [--swift-tenant-id <tenant-id>]
                                        [--swift-tenant-domain
                                        <tenant-domain>]
                                        [--swift-tenant-domain-id
                                        <tenant-domain-id>]
                                        [--swift-trust-id <trust-id>]
                                        [--swift-region <region>]
                                        [--swift-internal <internal>]
                                        [--swift-container <container>]
                                        [--swift-cert-verify <cert-verify>]
                                        [--azure-endpoint <endpoint>]
                                        [--azure-container <container>]
                                        [--azure-account-name <account-name>]
                                        [--azure-account-key <account-key>]
                                        [--google-bucket <bucket>]
                                        [--google-credentials <credentials>]
```

`--nodes <nodes>`

A comma-separated list of node hostnames or IDs

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

`--storage-type {local,nfs,s3,swift,azure,google}`

Storage type

`--stdin`

Use for setting registration password from `stdin`.

`--upstream-info-file <upstream-info-file>`

Path to the upstream information file

Storage parameters for the `nfs` storage type:

`--nfs-host <host>`

NFS hostname or IP address

`--nfs-export <export>`

Full path to the NFS export

`--nfs-version <version>`

NFS version (`3` or `4`)

Storage parameters for the `s3` storage type:

`--s3-flavor <flavor>` (optional)

Flavor name

`--s3-region <region>` (optional)

Set region for Amazon S3.

`--s3-bucket <bucket>`

Bucket name

`--s3-endpoint <endpoint>`

Endpoint URL

`--s3-access-key-id <access-key-id>`

Access key ID

`--s3-secret-key-id <secret-key-id>`

Secret key ID

`--s3-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the S3 endpoint

Storage parameters for the `swift` storage type:

`--swift-auth-url <auth-url>`

Authentication (keystone) URL

`--swift-auth-version <auth-version>` (optional)

Authentication protocol version

`--swift-user-name <user-name>`

User name

`--swift-api-key <api-key>`

API key (password)

`--swift-domain <domain>` (optional)

Domain name

`--swift-domain-id <domain-id>` (optional)

Domain ID

`--swift-tenant <tenant>` (optional)

Tenant name

`--swift-tenant-id <tenant-id>` (optional)

Tenant ID

`--swift-tenant-domain <tenant-domain>` (optional)

Tenant domain name

`--swift-tenant-domain-id <tenant-domain-id>` (optional)

Tenant domain ID

`--swift-trust-id <trust-id>` (optional)

Trust ID

`--swift-region <region>` (optional)

Region name

`--swift-container <container>` (optional)

Container name

`--swift-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the Swift endpoint (`true` or `false`)

Storage parameters for the `azure` storage type:

`--azure-endpoint <endpoint>`

Endpoint URL

`--azure-container <container>`

Container name

`--azure-account-name <account-name>`

Account name

`--azure-account-key <account-key>`

Account key

Storage parameters for the `google` storage type:

`--google-bucket <bucket>`

Google bucket name

`--google-credentials <credentials>`

Path to the file with Google credentials

Example:

```
# vinfra service backup cluster deploy-reverse-proxy \
--nodes 74cbd22b-fb1b-4441-ae52-532078c54f9a --storage-type local \
--tier 0 --encoding 1+0 --failure-domain 0 \
--upstream-info-file /root/upstream1.info
Operation accepted.
```

This command creates the reverse proxy backup storage on the local storage and registers the upstream backup storage by using the upstream configuration file.

vinfra service backup cluster deploy-upstream

Create the upstream backup cluster for the reverse proxy:

```
usage: vinfra service backup cluster deploy-upstream --nodes <nodes>
                                                     --tier {0,1,2,3}
                                                     --encoding <M>+<N>
                                                     --failure-domain {0,1,2,3,4}
                                                     --storage-type {local,nfs,s3,
                                                     swift,azure,google} [--stdin]
                                                     --address <address>
                                                     [--nfs-host <host>]
                                                     [--nfs-export <export>]
                                                     [--nfs-version <version>]
                                                     [--s3-flavor <flavor>]
                                                     [--s3-region <region>]
                                                     [--s3-bucket <bucket>]
                                                     [--s3-endpoint <endpoint>]
                                                     [--s3-access-key-id <access-key-id>]
                                                     [--s3-secret-key-id <secret-key-id>]
                                                     [--s3-cert-verify <cert-verify>]
                                                     [--swift-auth-url <auth-url>]
                                                     [--swift-auth-version <auth-version>]
                                                     [--swift-user-name <user-name>]
                                                     [--swift-api-key <api-key>]
                                                     [--swift-domain <domain>]
                                                     [--swift-domain-id <domain-id>]
                                                     [--swift-tenant <tenant>]
                                                     [--swift-tenant-id <tenant-id>]
                                                     [--swift-tenant-domain
                                                     <tenant-domain>]
                                                     [--swift-tenant-domain-id
                                                     <tenant-domain-id>]
                                                     [--swift-trust-id <trust-id>]
                                                     [--swift-region <region>]
                                                     [--swift-internal <internal>]
                                                     [--swift-container <container>]
                                                     [--swift-cert-verify <cert-verify>]
                                                     [--azure-endpoint <endpoint>]
                                                     [--azure-container <container>]
                                                     [--azure-account-name <account-name>]
                                                     [--azure-account-key <account-key>]
```

```
                                        [--google-bucket <bucket>]
                                        [--google-credentials <credentials>]
```

`--nodes <nodes>`

A comma-separated list of node hostnames or IDs

`--tier {0,1,2,3}`

Storage tier

`--encoding <M>+<N>`

Storage erasure encoding mapping in the format:

- `M`: the number of data blocks
- `N`: the number of parity blocks

`--failure-domain {0,1,2,3,4}`

Storage failure domain

`--storage-type {local,nfs,s3,swift,azure,google}`

Storage type

`--stdin`

Use for setting registration password from `stdin`.

`--address <address>`

Address of the upstream backup storage

Storage parameters for the `nfs` storage type:

`--nfs-host <host>`

NFS hostname or IP address

`--nfs-export <export>`

Full path to the NFS export

`--nfs-version <version>`

NFS version (`3` or `4`)

Storage parameters for the `s3` storage type:

`--s3-flavor <flavor>` (optional)

Flavor name

`--s3-region <region>` (optional)

Set region for Amazon S3.

`--s3-bucket <bucket>`

Bucket name

`--s3-endpoint <endpoint>`

Endpoint URL

`--s3-access-key-id <access-key-id>`

Access key ID

`--s3-secret-key-id <secret-key-id>`

Secret key ID

`--s3-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the S3 endpoint

Storage parameters for the `swift` storage type:

`--swift-auth-url <auth-url>`

Authentication (keystone) URL

`--swift-auth-version <auth-version>` (optional)

Authentication protocol version

`--swift-user-name <user-name>`

User name

`--swift-api-key <api-key>`

API key (password)

`--swift-domain <domain>` (optional)

Domain name

`--swift-domain-id <domain-id>` (optional)

Domain ID

`--swift-tenant <tenant>` (optional)

Tenant name

`--swift-tenant-id <tenant-id>` (optional)

Tenant ID

`--swift-tenant-domain <tenant-domain>` (optional)

Tenant domain name

`--swift-tenant-domain-id <tenant-domain-id>` (optional)

Tenant domain ID

`--swift-trust-id <trust-id>` (optional)

Trust ID

`--swift-region <region>` (optional)

Region name

`--swift-container <container>` (optional)

Container name

`--swift-cert-verify <cert-verify>` (optional)

Allow self-signed certificate of the Swift endpoint (`true` or `false`)

Storage parameters for the `azure` storage type:

`--azure-endpoint <endpoint>`

Endpoint URL

`--azure-container <container>`

Container name

`--azure-account-name <account-name>`

Account name

`--azure-account-key <account-key>`

Account key

Storage parameters for the `google` storage type:

`--google-bucket <bucket>`

Google bucket name

`--google-credentials <credentials>`

Path to the file with Google credentials

Example:

```
# vinfra service backup cluster deploy-upstream \
--nodes eeb06dce-4cfd-4c89-bc7f-4689ea5c7058 --storage-type local \
--tier 0 --encoding 1+0 --failure-domain 0 --address upstream2.example.com
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| failed  | False                                |
| id      | 021a92bc-8ebc-4ecc-a397-fcd207a5872c |
| message |                                      |
| state   | new                                  |
+---------+--------------------------------------+
```

This command creates the upstream backup storage with the address `upstream2.example.com` on the local storage.

vinfra service backup cluster add-upstream

Add a new upstream to the reverse proxy backup storage:

```
usage: vinfra service backup cluster add-upstream --upstream-info-file
                                                  <upstream-info-file>
```

`--upstream-info-file <upstream-info-file>`

Path to the upstream information file

Example:

```
# vinfra service backup cluster add-upstream \
--upstream-info-file /root/upstream2.info
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| failed  | False                                |
| id      | 570363fc-ff87-4710-b28d-faba63019b34 |
| message |                                      |
| state   | new                                  |
+---------+--------------------------------------+
```

This command adds the new upstream backup storage to the reverse proxy backup storage.

vinfra service backup cluster process

Inspect and manipulate the backup storage process:

```
usage: vinfra service backup cluster process [--show | --retry]
                                             [--process-id <process-id>]
```

`--show`

Show the state of the backup storage process.

`--retry`

Retry a suspended backup storage process.

`--process-id <process-id>`

Backup storage process ID

Example:

```
# vinfra service backup cluster process --show \
--process-id 15bf7eb1-9fbd-436a-8936-33f7088c4933
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| failed  | False                                |
| id      | 15bf7eb1-9fbd-436a-8936-33f7088c4933 |
| message |                                      |
| state   | done                                 |
+---------+--------------------------------------+
```

This command shows the state of the backup storage process with the ID `ee7e60c5-5447-4177-8581-26657ac380c0`.

Managing general settings

Managing licenses

vinfra cluster license load

Load a license from a key.

```
usage: vinfra cluster license load <license-key>
```

`<license-key>`

License key to register.

Example:

```
# vinfra cluster license load A38600-3P6W74-RZSK58-Y9ZH05-2X7J48
+---------------+------------------------+
| Field         | Value                  |
+---------------+------------------------+
| capacity      | 7036767043584000       |
| expiration_ts | 1549583999             |
| free_size     | 7036766991361913       |
| keynumber     | VZSTOR.74418710.0000   |
| spla          | registered: false      |
|               | registration_url: null |
| status        | active                 |
| total_size    | 7036767043584000       |
| used_size     | 52222087               |
+---------------+------------------------+
```

This command installs the license from the key `A38600-3P6W74-RZSK58-Y9ZH05-2X7J48`.

vinfra cluster license show

Show details of the installed license:

```
usage: vinfra cluster license show
```

Example:

```
# vinfra cluster license show
+---------------+------------------------+
| Field         | Value                  |
+---------------+------------------------+
| capacity      | 7036767043584000       |
| expiration_ts | 1549583999             |
| free_size     | 7036766991361913       |
```

```
| keynumber     | VZSTOR.74418710.0000   |
| spla          | registered: false      |
|               | registration_url: null |
| status        | active                 |
| total_size    | 7036767043584000       |
| used_size     | 52222087               |
+---------------+------------------------+
```

This command shows the details of the currently installed license.

vinfra cluster license update

Update the installed license:

```
usage: vinfra cluster license update [--server <ka-server>]
```

`--server <ka-server>`

Hostname[:port] of the key administration server (default: `ka.parallels.com`)

Example:

```
vinfra cluster license update --server ka.parallels.com
+---------------+------------------------+
| Field         | Value                  |
+---------------+------------------------+
| capacity      | 7036767043584000       |
| expiration_ts | 1549583999             |
| free_size     | 7036766991361913       |
| keynumber     | VZSTOR.74418710.0000   |
| spla          | registered: false      |
|               | registration_url: null |
| status        | active                 |
| total_size    | 7036767043584000       |
| used_size     | 52222087               |
+---------------+------------------------+
```

This command shows the details of the currently installed license.

Managing updates

vinfra software-updates check-for-updates

Check for software updates:

```
usage: vinfra software-updates check-for-updates
```

Example:

```
# vinfra software-updates check-for-updates
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 80a06090-9d3d-4cb3-b7b0-b2ef8b9289f2 |
+---------+--------------------------------------+
```

This command creates a task to check if there are updates for the storage cluster.

Task outcome:

```
# vinfra task show 0143aec7-f9ce-4654-ad48-edb6f4104e22
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.business.models.software_updates.tasks.CheckSoftwareUpdate… |
| result  |                                                                        |
| state   | success                                                                |
| task_id | 80a06090-9d3d-4cb3-b7b0-b2ef8b9289f2                                   |
+---------+------------------------------------------------------------------------+
```

vinfra software-updates eligibility-check

Check nodes' update eligibility:

```
usage: vinfra software-updates eligibility-check
```

Example:

```
# vinfra software-updates eligibility-check
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 0143aec7-f9ce-4654-ad48-edb6f4104e22 |
+---------+--------------------------------------+
```

This command creates a task to check whether the nodes in the storage cluster are eligible for updates.

Task outcome:

```
# vinfra task show 0143aec7-f9ce-4654-ad48-edb6f4104e22
+---------+------------------------------------------------------------------+
| Field   | Value                                                            |
+---------+------------------------------------------------------------------+
| details |                                                                  |
| name    | backend.presentation.software_updates.tasks.EligibilityCheckTask |
| result  | cluster_has_releasing_nodes:                                     |
```



```
|         |   details: null                                                    |
|         |   exception: null                                                  |
|         |   message: null                                                    |
|         |   passed: true                                                     |
|         |   severity: critical                                               |
|         | cluster_unhealthy:                                                 |
|         |   details: null                                                    |
|         |   exception: null                                                  |
|         |   message: null                                                    |
|         |   passed: true                                                     |
|         |   severity: critical                                               |
|         | not_enough_space_on_agents:                                        |
|         |   details: null                                                    |
|         |   exception: null                                                  |
|         |   message: null                                                    |
|         |   passed: true                                                     |
|         |   severity: critical                                               |
|         | not_enough_space_on_mn:                                            |
|         |   details: null                                                    |
|         |   exception: null                                                  |
|         |   message: null                                                    |
|         |   passed: true                                                     |
|         |   severity: critical                                               |
|         | postgres_not_running:                                              |
|         |   details: null                                                    |
|         |   exception: null                                                  |
|         |   message: null                                                    |
|         |   passed: true                                                     |
|         |   severity: critical                                               |
| state   | success                                                            |
| task_id | 0143aec7-f9ce-4654-ad48-edb6f4104e22                               |
+---------+--------------------------------------------------------------------+
```

vinfra software-updates download

Download software updates:

```
usage: vinfra software-updates download
```

Example:

```
# vinfra software-updates download
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 2f930030-22de-4ce5-bf00-05328ee672f0 |
+---------+--------------------------------------+
```

This command creates a task to download updates.

Task outcome:


```
# vinfra task show 2f930030-22de-4ce5-bf00-05328ee672f0
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.business.models.software_updates.tasks.DownloadSoftwareUpd… |
| result  |                                                                        |
| state   | success                                                                |
| task_id | 2f930030-22de-4ce5-bf00-05328ee672f0                                   |
+---------+------------------------------------------------------------------------+
```

vinfra software-updates start

Start the software update procedure:

```
usage: vinfra software-updates start [--maintenance enabled={yes,no}[,key=value,…]]
                                     [--nodes <nodes>] [--skip-control-plane]
```

`--maintenance enabled={yes,no}[,key=value,…]>`

Specify maintenance parameters:

- `enabled`: enter maintenance during the upgrade (`yes` or `no`)
- comma-separated `key=value` pairs with keys (optional):
  - `on-fail`: choose how to proceed with the update if maintenance fails:
    - `stop` (default): stop the update if a node cannot enter maintenance mode. Nodes that have already been updated will remain so.
    - `skip`: skip and do not update nodes that cannot enter maintenance mode
    - `force`: forcibly update and reboot (if needed) all nodes even if they cannot enter maintenance mode. Using this option may result in downtime.
  - `compute-mode`: choose how to proceed with the update if a VM cannot be live migrated:
    - `strict`: stop the upgrade if a VM cannot be live migrated
    - `ignore`: ignore a VM that cannot be live migrated
    - `ignore_ext`: ignore a VM that cannot be or failed to be live migrated

`--nodes <nodes>`

A comma-separated list of node IDs or hostnames

`--skip-control-plane`

Update the cluster without updating the management panel.

Example:

```
# vinfra software-updates start --nodes node001,node002,node003 \
--maintenance enabled=yes,on-fail=skip,compute-mode=ignore
+---------+-------------------------------------+
```



```
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 0eae9159-7595-42a7-8feb-d04df3e295c7 |
+---------+--------------------------------------+
```

This command creates a task to start updating the nodes `node001`, `node002`, and `node003` and put them into maintenance. Those nodes that cannot enter maintenance will be skipped. Virtual machines that cannot be live migrated during maintenance will be ignored.

Task outcome:

```
# vinfra task show 0eae9159-7595-42a7-8feb-d04df3e295c7
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.business.models.software_updates.tasks.StartSoftwareUpdate… |
| result  |                                                                        |
| state   | running                                                                |
| task_id | 0eae9159-7595-42a7-8feb-d04df3e295c7                                   |
+---------+------------------------------------------------------------------------+
```

vinfra software-updates pause

Pause software updates:

```
usage: vinfra software-updates pause
```

Example:

```
# vinfra software-updates pause
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | b02a686b-3214-447e-a9b4-43698aa9388b |
+---------+--------------------------------------+
```

This command creates a task to pause updates.

Task outcome:

```
# vinfra task show b02a686b-3214-447e-a9b4-43698aa9388b
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.presentation.software_updates.tasks.PauseSoftwareUpdateTask |
| result  |                                                                        |
| state   | success                                                                |
```

```
| task_id | b02a686b-3214-447e-a9b4-43698aa9388b                              |
+---------+-------------------------------------------------------------------+
```

vinfra software-updates resume

Resume the software update procedure:

```
usage: vinfra software-updates resume
```

Example:

```
# vinfra software-updates resume
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 35323989-bdbc-4826-94c3-70ed7d06969d |
+---------+--------------------------------------+
```

This command creates a task to resume the update.

Task outcome:

```
# vinfra task show 35323989-bdbc-4826-94c3-70ed7d06969d
+---------+-------------------------------------------------------------------------+
| Field   | Value                                                                   |
+---------+-------------------------------------------------------------------------+
| details |                                                                         |
| name    | backend.presentation.software_updates.tasks.ResumeSoftwareUpdate… |
| result  |                                                                         |
| state   | success                                                                 |
| task_id | 35323989-bdbc-4826-94c3-70ed7d06969d                                    |
+---------+-------------------------------------------------------------------------+
```

vinfra software-updates cancel

Cancel software updates:

```
usage: vinfra software-updates cancel [--maintenance-mode
                                       {exit,exit-keep-resources,hold}]
```

`--maintenance-mode {exit,exit-keep-resources,hold}`

Maintenance mode:

- `exit`: exit maintenance for the node and return evacuated resources back on the node
- `exit-keep-resources` (default): exit maintenance for the node but keep evacuated resources on another node
- `hold`: do not exit maintenance

Example:


```
# vinfra software-updates cancel exit
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 7aeb20ba-1f9f-4f28-9790-086428d3e18e |
+---------+--------------------------------------+
```

This command creates a task to cancel the update and return the node to operation. The evacuated resources from this node will be moved back to it.

Task outcome:

```
# vinfra task show 7aeb20ba-1f9f-4f28-9790-086428d3e18e
+---------+------------------------------------------------------------------------+
| Field   | Value                                                                  |
+---------+------------------------------------------------------------------------+
| details |                                                                        |
| name    | backend.presentation.software_updates.tasks.CancelSoftwareUpdate… |
| result  |                                                                        |
| state   | success                                                                |
| task_id | 7aeb20ba-1f9f-4f28-9790-086428d3e18e                                   |
+---------+------------------------------------------------------------------------+
```

vinfra software-updates status

Check software update status:

```
usage: vinfra software-updates status
```

Example:

```
# vinfra software-updates status
+--------------------------+-------------------------------------------+
| Field                    | Value                                     |
+--------------------------+-------------------------------------------+
| available_storage_release | release: '758'                           |
|                          | version: 3.5.0                            |
| last_check_datetime      | 2019-12-17T13:25:41.991763+00:00          |
| nodes                    | - current_storage_release:                |
|                          |     release: '758'                        |
|                          |     version: 3.5.0                        |
|                          |   downloaded_storage_release: null        |
|                          |   host: man-hci7-1.vstoragedomain         |
|                          |   id: 51cc14d4-eec6-433e-a7b1-e1c5c7f9555e |
|                          |   orig_hostname: man-hci7-1               |
|                          |   status: uptodate                        |
| services                 | []                                        |
| status                   | uptodate                                  |
| tasks                    | - errors:                                 |
|                          |     message: None                         |
```

```
|                           |     nodes: []                              |
|                           |   id: 8cf880d1-6648-450b-b96c-c87c27b9e181 |
|                           |   name: StartSoftwareUpdateTask            |
|                           |   params:                                  |
|                           |     force: false                           |
|                           |     skip: false                            |
+---------------------------+--------------------------------------------+
```

This command shows the node update status.

Managing domains

vinfra domain create

Create a new domain:

```
usage: vinfra domain create [--description <description>]
                            [--enable | --disable] <name>
```

`--description <description>`
: Domain description

`--enable`
: Enable domain

`--disable`
: Disable domain

`<name>`
: Domain name

Example:

```
# vinfra domain create mydomain
+----------------+----------------------------------+
| Field          | Value                            |
+----------------+----------------------------------+
| description    |                                  |
| enabled        | True                             |
| id             | ed408d00561c4a398f933c29e87cadab |
| name           | domain1                          |
| projects_count | 0                                |
+----------------+----------------------------------+
```

This command creates and enables the domain `mydomain`.

vinfra domain list

List all available domains:


```
usage: vinfra domain list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra domain list
+----------------------------------+----------+---------+--------------------+
| id                               | name     | enabled | description        |
+----------------------------------+----------+---------+--------------------+
| default                          | Default  | True    | The default domain |
| 24986479ee3246048d3ef2a065ea99f5 | mydomain | True    |                    |
+----------------------------------+----------+---------+--------------------+
```

This command lists domains used in the compute cluster.

vinfra domain show

Display information about a domain:

```
usage: vinfra domain show <domain>
```

`<domain>`

Domain ID or name

Example:

```
# vinfra domain show mydomain
+---------------+----------------------------------+
| Field         | Value                            |
+---------------+----------------------------------+
| description   |                                  |
| enabled       | True                             |
| id            | 24986479ee3246048d3ef2a065ea99f5 |
| name          | mydomain                         |
| projects_count | 0                               |
+---------------+----------------------------------+
```

This command shows the details of the domain `mydomain`.

vinfra domain set

Modify an existing domain:

```
usage: vinfra domain set [--description <description>] [--enable | --disable]
                         [--name <name>] <domain>
```

`--description <description>`

Domain description

`--enable`

Enable domain

`--disable`

Disable domain

`--name <name>`

Domain name

`<domain>`

Domain ID or name

Example:

```
# vinfra domain set mydomain --description "A custom domain"
+----------------+----------------------------------+
| Field          | Value                            |
+----------------+----------------------------------+
| description    | A custom domain                  |
| enabled        | True                             |
| id             | 24986479ee3246048d3ef2a065ea99f5 |
| name           | mydomain                         |
| projects_count | 0                                |
+----------------+----------------------------------+
```

This command adds the description for the domain `mydomain`.

vinfra domain delete

Delete a domain:

```
usage: vinfra domain delete <domain>
```

`<domain>`

Domain ID or name

Example:

```
# vinfra domain delete mydomain
Operation successful
```

This command deletes the domain `mydomain`.

Managing domain users

vinfra domain user list-available-roles

List available user roles:

```
usage: vinfra domain user list-available-roles [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
#  vinfra domain user list-available-roles
+---------------+---------------+-----------------------------------+-----------+
| id            | name          | description                       | scope     |
+---------------+---------------+-----------------------------------+-----------+
| abgw          | ABGW          | Can create and manage Acronis     | - system  |
|               |               | Backup Gateway.                   |           |
| admin         | Administrator | Can perform all management        | - system  |
|               |               | operations.                       |           |
| cluster       | Cluster       | Can create cluster, join nodes    | - system  |
|               |               | to cluster, and manage (assign    |           |
|               |               | and release) disks.               |           |
| compute       | Compute       | Can create and manage compute     | - system  |
|               |               | cluster.                          |           |
| domain_admin  | Domain Admin  | Can manage users, projects and    | - domain  |
|               |               | all resources in a domain.        |           |
| image_upload  | Image Upload  | Can manage compute images.        | - domain  |
| iscsi         | Block Storage | Can create and manage iSCSI       | - system  |
|               |               | targets, LUNs and CHAP users.     |           |
| login         | Login         | Can login in web UI.              | []        |
| network       | Network       | Can modify network settings       | - system  |
|               |               | and roles.                        |           |
| nfs           | NFS           | Can create and manage NFS.        | - system  |
| project_admin | Project Admin | Can manage virtual objects        | - project |
|               |               | inside a project.                 |           |
| s3            | S3            | Can create and manage S3 cluster. | - system  |
| ssh           | SSH           | Can add and remove SSH keys for   | - system  |
|               |               | cluster nodes access.             |           |
| updates       | Updates       | Can install updates.              | - system  |
| viewer        | Viewer        | Viewer role (read only)           | - system  |
+---------------+---------------+-----------------------------------+-----------+
```

This command lists all available user roles.

vinfra domain user create

Create a new domain user:

```
usage: vinfra domain user create [--email <email>] [--description <description>]
                                 [--assign <project> <role>]
                                 [--assign-domain <domain> <roles>]
                                 [--domain-permissions <domain_permissions>]
                                 [--system-permissions <system_permissions>]
                                 [--enable | --disable] --domain <domain> <name>
```

`--email <email>`

User email

`--description <description>`

User description

`--assign <project> <role>`

Assign a user to a project with one or more permission sets. Specify this option multiple times to assign the user to multiple projects.

- `<project>`: project ID or name
- `<role>`: user role in the project (`project_admin`)

`--assign-domain <domain> <roles>`

Assign a user to a domain with one or more permission sets. Specify this option multiple times to assign the user to multiple domains. This option is only valid for service accounts.

- `<domain>`: domain ID or name
- `<roles>`: a comma-separated list of service account roles (`compute`)

`--domain-permissions <domain_permissions>`

A comma-separated list of domain permissions. View the list of available domain permissions using `vinfra domain user list-available-roles | grep domain`.

`--enable`

Enable user

`--disable`

Disable user

`--domain <domain>`

Domain name or ID

`<name>`

User name

Example:

```
#  vinfra domain user create myuser --domain mydomain \
--domain-permissions domain_admin
Password:
+--------------------+----------------------------------+
```



```
| Field              | Value                              |
+--------------------+------------------------------------+
| assigned_domains   | []                                 |
| assigned_projects  | []                                 |
| description        |                                    |
| domain_id          | 2929ff42b1e64884a05dea3011862aed |
| domain_permissions | - domain_admin                     |
| email              |                                    |
| enabled            | True                               |
| id                 | a9c67c6acf1f4df1818fdeeee0b4bd5e |
| name               | myuser                             |
| role               | domain_admin                       |
| system_permissions | []                                 |
| tags               | []                                 |
+--------------------+------------------------------------+
```

This command creates and enables a new administrator account `myuser` within the domain `mydomain`. It also sets password for the new user.

vinfra domain user list

List all users in a domain:

```
usage: vinfra domain user list [--long] --domain <domain>
                               [--limit <num>] [--marker <user>]
                               [--name <name>] [--id <id>]
                               [--tags <tag>[,<tag>,…]]
```

`--long`

Enable access and listing of all fields of objects.

`--domain <domain>`

Domain name or ID

`--limit <num>`

The maximum number of users to list. To list all users, set the option to -1.

`--marker <user>`

List users after the marker.

`--name <name>`

List users with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a user with the specified ID or list users using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--tags <tag>[,<tag>,…]`

List projects with the specified tags (comma-separated) or use a filter. Supported filter operators: `any`, `not_any`. The filter format is `<operator>:<value1>[,<value2>,…]`.

Example:

```
# vinfra domain user list --domain mydomain -c id -c name -c enabled \
-c domain_permissions -c assigned_projects
+-----------+--------+---------+-------------------+-------------------+
| id        | name   | enabled |domain_permissions | assigned_projects |
+-----------+--------+---------+-------------------+-------------------+
| a9c6<...> | myuser | True    |- domain_admin     | []                |
+-----------+--------+---------+-------------------+-------------------+
```

This command lists users in the domain `mydomain`.

vinfra domain user show

Display information about a domain user:

```
usage: vinfra domain user show --domain <domain> <user>
```

`--domain <domain>`

Domain ID or name

`<user>`

User ID or name

Example:

```
# vinfra domain user show myuser --domain mydomain
+--------------------+----------------------------------+
| Field              | Value                            |
+--------------------+----------------------------------+
| assigned_domains   | []                               |
| assigned_projects  | []                               |
| description        |                                  |
| domain_id          | 2929ff42b1e64884a05dea3011862aed |
| domain_permissions | - domain_admin                   |
| email              |                                  |
| enabled            | True                             |
| id                 | a9c67c6acf1f4df1818fdeeee0b4bd5e |
| name               | myuser                           |
| role               | domain_admin                     |
| system_permissions | []                               |
| tags               | []                               |
+--------------------+----------------------------------+
```

This command shows the details of the user `myuser` from the domain `mydomain`.

vinfra domain user set

Modify the parameters of a domain user:

```
usage: vinfra domain user set [--password] [--email <email>]
                              [--description <description>]
                              [--assign <project> <role>]
                              [--assign-domain <domain> <roles>]
                              [--unassign-domain <domain>]
                              [--domain-permissions <domain_permissions>]
                              [--system-permissions <system_permissions>]
                              [--enable | --disable] [--name <name>]
                              --domain <domain> <user>
```

`--password`

Request the password from stdin

`--email <email>`

User email

`--description <description>`

User description

`--assign <project> <role>`

Assign a user to a project with one or more permission sets. Specify this option multiple times to assign the user to multiple projects.

- `<project>`: project ID or name
- `<role>`: user role in the project (`project_admin`)

`--assign-domain <domain> <roles>`

Assign a user to a domain with one or more permission sets. Specify this option multiple times to assign the user to multiple domains. This option is only valid for service accounts.

- `<domain>`: domain ID or name
- `<roles>`: a comma-separated list of service account roles (`compute`)

`--domain-permissions <domain_permissions>`

A comma-separated list of domain permissions. View the list of available domain permissions using `vinfra domain user list-available-roles | grep domain`.

`--enable`

Enable user

`--disable`

Disable user

`--name <name>`

User name

`--domain <domain>`

Domain name or ID

`<user>`

User ID or name

Example:

```
# vinfra domain user set myuser --domain mydomain \
--assign myproject project_admin
+--------------------+-------------------------------------------------+
| Field              | Value                                           |
+--------------------+-------------------------------------------------+
| assigned_domains   | []                                              |
| assigned_projects  | - project_id: d1c4d6198fb940e6b971cf306571ebbd |
|                    |   role: project_admin                           |
| description        |                                                 |
| domain_id          | 2929ff42b1e64884a05dea3011862aed                |
| domain_permissions | []                                              |
| email              |                                                 |
| enabled            | True                                            |
| id                 | a9c67c6acf1f4df1818fdeeee0b4bd5e                |
| name               | myuser                                          |
| role               | project_member                                  |
| system_permissions | []                                              |
| tags               | []                                              |
+--------------------+-------------------------------------------------+
```

This command assigns the user `myuser` from the domain `mydomain` to the project `myproject` as a project administrator.

vinfra domain user delete

Remove a domain user:

```
usage: vinfra domain user delete --domain <domain> <user>
```

`--domain <domain>`

Domain ID or name

`<user>`

User ID or name

Example:

```
# vinfra domain user delete myuser --domain mydomain
Operation successful
```

This command deletes the user `myuser` from the domain `mydomain`.

Managing domain projects

vinfra domain project create

Create a new domain project:

```
usage: vinfra domain project create [--description <description>]
                                    [--enable | --disable] --domain <domain>
                                    <name>
```

`--description <description>`

Project description

`--enable`

Enable project

`--disable`

Disable project

`--domain <domain>`

Domain name or ID

`<name>`

Project name

Example:

```
# vinfra domain project create myproject --domain mydomain \
--description "A custom project"
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | A custom project                 |
| domain_id   | 9f7e68938fe946a2a862e360bbe40d98 |
| enabled     | True                             |
| id          | d1c4d6198fb940e6b971cf306571ebbd |
| name        | myproject                        |
| tags        | []                               |
+-------------+----------------------------------+
```

This command creates and enables the project `myproject` within the domain `mydomain` and adds a description to it.

vinfra domain project list

List all projects in a domain:

```
usage: vinfra domain project list [--long] --domain <domain>
                                  [--limit <num>] [--marker <project>]
                                  [--name <name>] [--id <id>]
                                  [--tags <tag1>[,<tag2>,…]]
```

`--long`

Enable access and listing of all fields of objects.

`--domain <domain>`

Domain name or ID

`--limit <num>`

The maximum number of projects to list. To list all projects, set the option to -1.

`--marker <project>`

List projects after the marker.

`--name <name>`

List projects with the specified name or use a filter. Supported filter operator: `contains`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--id <id>`

Show a project with the specified ID or list projects using a filter. Supported filter operator: `in`. The filter format is `<operator>:<value1>[,<value2>,…]`.

`--tags <tag>[,<tag>,…]`

List projects with the specified tags (comma-separated) or use a filter. Supported filter operators: `any`, `not_any`. The filter format is `<operator>:<value1>[,<value2>,…]`.

Example:

```
# vinfra domain project list --domain mydomain
+-----------+-----------+---------+------------------+-----------+
| id        | name      | enabled | description      | domain_id |
+-----------+-----------+---------+------------------+-----------+
| d1c4<...> | myproject | True    | A custom project | 9f7e<...> |
+-----------+-----------+---------+------------------+-----------+
```

This command lists projects in the domain `mydomain`.

vinfra domain project show

Show details of a domain project:

```
usage: vinfra domain project show --domain <domain> <project>
```

`--domain <domain>`

Domain name or ID

`<project>`

Project ID or name

Example:

```
#  vinfra domain project show myproject --domain mydomain
+---------------+----------------------------------+
| Field         | Value                            |
+---------------+----------------------------------+
| description   | A custom project                 |
| domain_id     | 9f7e68938fe946a2a862e360bbe40d98 |
| enabled       | True                             |
| id            | d1c4d6198fb940e6b971cf306571ebbd |
| members_count | 0                                |
| name          | myproject                        |
| tags          | []                               |
+---------------+----------------------------------+
```

This command shows the details of the project `myproject` from the domain `mydomain`.

vinfra domain project set

Modify an existing project:

```
usage: vinfra domain project set [--description <description>]
                                 [--enable | --disable] [--name <name>]
                                 --domain <domain> <project>
```

`--description <description>`

Project description

`--enable`

Enable project

`--disable`

Disable project

`--name <name>`

Project name

`--domain <domain>`

Domain name or ID

`<project>`

Project ID or name

Example:

```
# vinfra domain project set myproject --domain mydomain --disable
+-------------+----------------------------------+
```

```
| Field       | Value                            |
+-------------+----------------------------------+
| description | A custom project                 |
| domain_id   | 9f7e68938fe946a2a862e360bbe40d98 |
| enabled     | False                            |
| id          | d1c4d6198fb940e6b971cf306571ebbd |
| name        | myproject                        |
| tags        | []                               |
+-------------+----------------------------------+
```

This command disables the project `myproject` from the domain `mydomain`.

vinfra domain project user list

List users of a project:

```
usage: vinfra domain project user list [--long] --domain <domain> <project>
```

`--long`

Enable access and listing of all fields of objects.

`--domain <domain>`

Domain name or ID

`<project>`

Project ID or name

Example:

```
# vinfra domain project user list myproject --domain mydomain
+----------------------------------+--------+-------------+---------------+
| id                               | name   | description | role          |
+----------------------------------+--------+-------------+---------------+
| eb0203e6b8a641d8be5b54b2f3fc9f47 | myuser |             | project_admin |
+----------------------------------+--------+-------------+---------------+
```

This command lists users of the project `myproject` within the domain `mydomain`.

vinfra domain project user remove

Remove a user from a project:

```
usage: vinfra domain project user remove --user <user> --domain <domain> <project>
```

`--user <user>`

User name or ID

`--domain <domain>`

Domain name or ID

`<project>`

Project ID or name

Example:

```
# vinfra domain project user remove myproject --domain mydomain --user myuser
Operation successful
```

This command removes the user `myuser` from the project `myproject` within the domain `mydomain`.

vinfra domain project delete

Delete a domain project:

```
usage: vinfra domain project delete --domain <domain> <project>
```

`--domain <domain>`

Domain name or ID

`<project>`

Project ID or name

Example:

```
# vinfra domain project delete myproject --domain mydomain
Operation successful
```

This command deletes the project `myproject` from the domain `mydomain`.

Managing domain properties

vinfra domain properties create

Create a property sheet for the domain:

```
usage: vinfra domain properties create --key <key> --data <data>
                                       [--access <access>] <domain>
```

`--key <key>`

Key name

`--data <data>`

Property sheet. Should be a valid JSON object.

`--access <access>`

Access type:

- `pub`: grant read access to all users (authentication is not required)
- `auth`: grant read access to authenticated users
- `domain`: grant read access to domain users

The superadmin and domain admin have write access.

`<domain>`

Domain name or ID

Example 1:

```
# vinfra domain properties create --key myproperty mydomain \
--data '{"key1": "value1", "key2": "value2"}' --access pub
Operation successful.
```

This command creates a property sheet with the key `myproperty` that is accessible by all users.

Example 2:

```
# vinfra domain properties create --key allow_live_resize mydomain \
--data '{"enabled":true}'
Operation successful.
```

This command creates the property `allow_live_resize` that enables CPU and RAM hot plug for virtual machines within the domain `mydomain`.

vinfra domain properties keys list

List keys and access rights of all domain property sheets:

```
usage: vinfra domain properties keys list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra domain properties keys list
+---------+---------------------+
| domain  | keys                |
+---------+---------------------+
| domain1 | - allow_live_resize |
|         | - myproperty        |
| Default | - allow_live_resize |
+---------+---------------------+
```

This command lists existing property sheets in all domains.

vinfra domain properties show

Show details of a domain property sheet:

```
usage: vinfra domain properties show --key <key> <domain>
```

`--key <key>`

Key name

`<domain>`

Domain name or ID

Example:

```
# vinfra domain properties show --key myproperty mydomain
+--------+--------------+
| Field  | Value        |
+--------+--------------+
| data   | key1: value1 |
|        | key2: value2 |
| domain | mydomain     |
| key    | myproperty   |
+--------+--------------+
```

This command shows the details of the property sheet with the key `myproperty`.

vinfra domain properties access set

Update access rights of the domain property sheet:

```
usage: vinfra domain properties access set [--access <access>]
                                           [--key <key>] <domain>
```

`--key <key>`

Key name

`--access <access>`

Access type:

- `pub`: grant read access to all users (authentication is not required)
- `auth`: grant read access to authenticated users
- `domain`: grant read access to domain users

The superadmin and domain admin have write access.

`<domain>`

Domain name or ID

Example:


```
# vinfra domain properties access set --key myproperty mydomain --access domain
Operation successful.
```

This command restricts read access for the property sheet with the key `myproperty` to domain users.

vinfra domain properties update

Update a domain property sheet:

```
usage: vinfra domain properties update --key <key> --data <data>
                                       [--access <access>] <domain>
```

`--key <key>`

Key name

`--data <data>`

Property sheet. Should be a valid JSON object.

`--access <access>`

Access type:

- `pub`: grant read access to all users (authentication is not required)
- `auth`: grant read access to authenticated users
- `domain`: grant read access to domain users

The superadmin and domain admin have write access.

`<domain>`

Domain name or ID

Example:

```
# vinfra domain properties update --key allow_live_resize mydomain \
--data '{"enabled": false}'
Operation successful.
```

This command changes the property `allow_live_resize` to disable CPU and RAM hot plug for virtual machines within the domain `mydomain`.

vinfra domain properties delete

Delete a domain property sheet:

```
usage: vinfra domain properties delete --key <key> <domain>
```

`--key <key>`

Key name

`<domain>`

Domain name or ID

Example:

```
# vinfra domain properties delete --key myproperty mydomain
Operation successful.
```

This command deletes the property sheet with the key `myproperty`.

Managing SSH keys

vinfra cluster sshkey add

Add an SSH public key from a file:

```
usage: vinfra cluster sshkey add <file>
```

`<file>`

SSH public key file

Example:

```
# vinfra cluster sshkey add id_rsa.pub
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 100a54ce-0bf5-4bc0-8e46-2e8b952343e6 |
+---------+--------------------------------------+
```

This command creates a task to add a public SSH key from the file `mykey.pub` to the list of trusted keys.

Task outcome:

```
# vinfra task show 100a54ce-0bf5-4bc0-8e46-2e8b952343e6
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| args    | - admin                                                    |
|         | - 1                                                        |
| kwargs  | key: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAA<...>              |
|         |   user@example.com                                         |
| name    | backend.presentation.nodes.ssh.tasks.CreateSshKeyTask      |
| result  | id: 6a2fb834-4bc6-4597-ae74-7cacf96b7c75                   |
|         | key: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAA<...>              |
|         |   user@example.com                                         |
|         | label: user@example.com                                    |
| state   | success                                                    |
```

```
| task_id | 100a54ce-0bf5-4bc0-8e46-2e8b952343e6                 |
+---------+------------------------------------------------------+
```

vinfra cluster sshkey list

Show the list of added SSH public keys:

```
usage: vinfra cluster sshkey list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra cluster sshkey list
+---------------------+----------------------------------+------------------+
| id                  | key                              | label            |
+---------------------+----------------------------------+------------------+
| 8ccf7f1b-6a53-<...> | ssh-rsa AAAAB3NzaC1yc2EAAAA<...> | user@example.com |
|                     | user@example.com                 |                  |
+---------------------+----------------------------------+------------------+
```

This command lists trusted SSH keys.

vinfra cluster sshkey delete

Remove an SSH public key from storage cluster nodes:

```
usage: vinfra cluster sshkey delete <sshkey>
```

`<sshkey>`

SSH key value

Example:

```
# vinfra cluster sshkey delete 8ccf7f1b-6a53-4d74-99ce-c410d51a9921
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 053802b2-b4c3-454d-89e2-6d6d312dd2ed |
+---------+--------------------------------------+
```

This command creates a task to delete the SSH key with the ID `8ccf7f1b-6a53-4d74-99ce-c410d51a9921`.

Task outcome:

```
# vinfra task show 053802b2-b4c3-454d-89e2-6d6d312dd2ed
+---------+-------------------------------------------------------+
| Field   | Value                                                 |
+---------+-------------------------------------------------------+
| args    | - admin                                               |
|         | - 1                                                   |
|         | - 8ccf7f1b-6a53-4d74-99ce-c410d51a9921                |
| kwargs  | {}                                                    |
| name    | backend.presentation.nodes.ssh.tasks.RemoveSshKeyTask |
| state   | success                                               |
| task_id | 053802b2-b4c3-454d-89e2-6d6d312dd2ed                  |
+---------+-------------------------------------------------------+
```

Managing external DNS servers

vinfra cluster settings dns show

Display DNS servers:

```
usage: vinfra cluster settings dns show
```

Example:

```
# vinfra cluster settings dns show
+------------------+----------------------------------+
| Field            | Value                            |
+------------------+----------------------------------+
| dhcp_nameservers | 10.10.0.10,10.10.0.11,10.37.130.2 |
| nameservers      | 10.10.0.11,10.10.0.10            |
+------------------+----------------------------------+
```

This command lists the currently used DNS servers: both internal (obtained via DHCP) and external (static set by the user).

vinfra cluster settings dns set

Set DNS servers:

```
usage: vinfra cluster settings dns set --nameservers <nameservers>
```

`--nameservers <nameservers>`

A comma-separated list of DNS servers

Example:

```
# vinfra cluster settings dns set --nameservers 8.8.8.8
+------------------+--------------+
```

```
| Field            | Value         |
+------------------+---------------+
| dhcp_nameservers | - 10.10.0.10  |
|                  | - 10.10.0.11  |
|                  | - 10.37.130.2 |
| nameservers      | - 8.8.8.8     |
+------------------+---------------+
```

This command sets the external DNS server to `8.8.8.8`.

Configuring management node high availability

vinfra cluster ha create

Create a HA configuration:

```
usage: vinfra cluster ha create --virtual-ip <network:ip>
                                --nodes <nodes> [--force]
```

`--virtual-ip <network:ip>`

HA configuration mapping in the format:

- `network`: network to include in the HA configuration (must include at least one of these traffic types: Internal management, Admin panel, Self-service panel, or Compute API).
- `ip`: virtual IP address that will be used in the HA configuration.

Specify this option multiple times to create a HA configuration for multiple networks.

`--nodes <nodes>`

A comma-separated list of node IDs or hostnames

`--force`

Skip checks for minimal hardware requirements

Example:

```
# vinfra cluster ha create --virtual-ip Private:10.37.130.200 \
--virtual-ip Public:10.94.129.79 --nodes 94d58604-6f30-4339-8578-adb7903b7277,\
f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4,7d7d37b8-4c06-4f1a-b3a6-4b54257d70ce
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 80a00e55-335d-4d41-bac4-5fee4791d423 |
+---------+--------------------------------------+
```

This command creates a task to create a management node HA cluster from nodes with the IDs `94d58604-6f30-4339-8578-adb7903b7277`, `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`, and `7d7d37b8-4c06-4f1a-b3a6-4b54257d70ce`.

The command must specify the network with the traffic type `Internal management` as well as one with the traffic type `Admin panel`.

Important
After the HA cluster has been created, the admin panel will only be accessible at the provided public IP address. Log in to said address via SSH to continue managing Virtuozzo Hybrid Infrastructure with the `vinfra` CLI tool. You may also need to set the `VINFRA_PASSWORD` environment variable again, because you will access different HA cluster nodes on each log in where it may not have been set.

Task outcome:

```
# vinfra task show 80a00e55-335d-4d41-bac4-5fee4791d423
+---------+-----------------------------------------------------+
| Field   | Value                                               |
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.presentation.ha.tasks.CreateHaConfigTask    |
| result  | compute_task_id: c5125024-5472-4420-b8b6-e03971ab952c |
|         | ha_cluster_location:                                |
|         | - https://10.94.129.79:8888                         |
|         | nodes:                                              |
|         | - id: 94d58604-6f30-4339-8578-adb7903b7277          |
|         |   ipaddr: 10.37.130.118                             |
|         |   is_primary: false                                 |
|         | - id: f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4          |
|         |   ipaddr: 10.37.130.134                             |
|         |   is_primary: true                                  |
|         | - id: 7d7d37b8-4c06-4f1a-b3a6-4b54257d70ce          |
|         |   ipaddr: 10.37.130.246                             |
|         |   is_primary: false                                 |
|         | primary_node_location: https://10.94.62.243:8888    |
|         | virtual_ips:                                        |
|         | - ip: 10.94.129.79                                  |
|         |   roles_set: 5f0adc1d-c10f-46c1-b7b8-dd1aacab613b   |
|         | - ip: 10.37.130.200                                 |
|         |   roles_set: 5a0401b5-9b42-4d8b-8372-71c747230033   |
| state   | success                                             |
| task_id | 80a00e55-335d-4d41-bac4-5fee4791d423                |
+---------+-----------------------------------------------------+
```

vinfra cluster ha update

Update the HA configuration:

```
usage: vinfra cluster ha update [--virtual-ip <network:ip>]
                                [--nodes <nodes>] [--force]
```

`--virtual-ip <network:ip>`

HA configuration mapping in the format:

- `network`: network to include in the HA configuration (must include at least one of these traffic types: Internal management, Admin panel, Self-service panel, or Compute API).
- `ip`: virtual IP address that will be used in the HA configuration.

Specify this option multiple times to create a HA configuration for multiple networks.

`--nodes <nodes>`

A comma-separated list of node IDs or hostnames

`--force`

Skip checks for minimal hardware requirements

Example:

```
# vinfra cluster ha update --nodes 94d58604-6f30-4339-8578-adb7903b7277,\
f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4,4b83a87d-9adf-472c-91f0-782c47b2d5f1
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 565e9146-254b-4f7a-a2ff-b7119c95baa9 |
+---------+--------------------------------------+
```

This command creates a task to update the management node HA configuration, that is, include the nodes with the IDs `94d58604-6f30-4339-8578-adb7903b7277`, `f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4`, and `4b83a87d-9adf-472c-91f0-782c47b2d5f1`.

Task outcome:

```
# vinfra task show 565e9146-254b-4f7a-a2ff-b7119c95baa9
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.presentation.ha.tasks.UpdateHaConfigTask           |
| result  | compute_task_id: 84994caf-3a02-43ea-b904-48632f0379c7      |
|         | ha_cluster_location:                                       |
|         | - https://10.94.129.79:8888                                |
|         | nodes:                                                     |
|         | - id: f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4                 |
|         |   ipaddr: 10.37.130.134                                    |
|         |   is_primary: true                                         |
|         | - id: 4b83a87d-9adf-472c-91f0-782c47b2d5f1                 |
|         |   ipaddr: 10.37.130.127                                    |
|         |   is_primary: false                                        |
|         | - id: 94d58604-6f30-4339-8578-adb7903b7277                 |
|         |   ipaddr: 10.37.130.118                                    |
|         |   is_primary: false                                        |
|         | primary_node_location: https://10.94.62.243:8888           |
|         | virtual_ips:                                               |
|         | - ip: 10.94.129.79                                         |
|         |   roles_set: 5f0adc1d-c10f-46c1-b7b8-dd1aacab613b          |
```



```
|         | - ip: 10.37.130.200                                    |
|         |   roles_set: 5a0401b5-9b42-4d8b-8372-71c747230033      |
| state   | success                                                |
| task_id | 565e9146-254b-4f7a-a2ff-b7119c95baa9                   |
+---------+--------------------------------------------------------+
```

vinfra cluster ha show

Display the HA configuration:

```
usage: vinfra cluster ha show
```

Example:

```
# vinfra cluster ha show
+----------------------+-------------------------------------------------+
| Field                | Value                                           |
+----------------------+-------------------------------------------------+
| ha_cluster_location  | - https://10.94.129.79:8888                     |
| nodes                | - id: 94d58604-6f30-4339-8578-adb7903b7277      |
|                      |   ipaddr: 10.37.130.118                         |
|                      |   is_primary: false                             |
|                      | - id: f59dabdb-bd1c-4944-8af2-26b8fe9ff8d4      |
|                      |   ipaddr: 10.37.130.134                         |
|                      |   is_primary: true                              |
|                      | - id: 4b83a87d-9adf-472c-91f0-782c47b2d5f1      |
|                      |   ipaddr: 10.37.130.127                         |
|                      |   is_primary: false                             |
| primary_node_location | https://10.94.62.243:8888                      |
| virtual_ips          | - ip: 10.37.130.200                             |
|                      |   roles_set: 5a0401b5-9b42-4d8b-8372-71c747230033 |
|                      | - ip: 10.94.129.79                              |
|                      |   roles_set: 5f0adc1d-c10f-46c1-b7b8-dd1aacab613b |
+----------------------+-------------------------------------------------+
```

This command shows the management node HA cluster configuration.

vinfra cluster ha delete

Delete the HA configuration:

```
usage: vinfra cluster ha delete
```

Example:

```
# vinfra cluster ha delete
+---------+-------------------------------------+
| Field   | Value                               |
```

```
+---------+--------------------------------------+
| task_id | c1f3e9c3-0a7b-455a-96d4-cef3b7e86e62 |
+---------+--------------------------------------+
```

This command creates a task to delete the management node HA cluster.

Task outcome:

```
# vinfra task show c1f3e9c3-0a7b-455a-96d4-cef3b7e86e62
+---------+-----------------------------------------------------+
| Field   | Value                                               |
+---------+-----------------------------------------------------+
| details |                                                     |
| name    | backend.presentation.ha.tasks.DeleteHaConfigTask |
| result  |                                                     |
| state   | success                                             |
| task_id | c1f3e9c3-0a7b-455a-96d4-cef3b7e86e62             |
+---------+-----------------------------------------------------+
```

Managing cluster backups

vinfra cluster backup create

Create a backup:

```
usage: vinfra cluster backup create
```

Example:

```
# vinfra cluster backup create
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | e4b4f891-cc3a-4308-a321-d76265ef7b5b |
+---------+--------------------------------------+
```

This command creates a task to back up the storage cluster.

Task outcome:

```
# vinfra task show e4b4f891-cc3a-4308-a321-d76265ef7b5b
+---------+------------------------------------------------------------+
| Field   | Value                                                      |
+---------+------------------------------------------------------------+
| details |                                                            |
| name    | backend.presentation.backups.tasks.BackupManagedNodeTask |
| result  | status: finished                                           |
| state   | success                                                    |
```



```
| task_id | e4b4f891-cc3a-4308-a321-d76265ef7b5b                    |
+---------+---------------------------------------------------------+
```

vinfra cluster backup show

Show backup information:

```
usage: vinfra cluster backup show
```

Example:

```
# vinfra cluster backup show
+----------------------+-----------------------------+
| Field                | Value                       |
+----------------------+-----------------------------+
| last_backup_date     | 2019-08-21T15:41:24+00:00   |
| last_backup_location | /mnt/vstorage/webcp/backup/ |
| ready                | True                        |
| tasks                | []                          |
+----------------------+-----------------------------+
```

This command shows the details of the last cluster backup and the ID of the ongoing backup task, if any.

Managing storage tier encryption

vinfra cluster settings encryption show

Display storage tier encryption:

```
usage: vinfra cluster settings encryption show
```

Example:

```
# vinfra cluster settings encryption show
+-------+-------+
| Field | Value |
+-------+-------+
| tier0 | False |
| tier1 | False |
| tier2 | False |
| tier3 | False |
+-------+-------+
```

This command shows encryption status of each storage tier.

vinfra cluster settings encryption set

Set storage tier encryption:

```
usage: vinfra cluster settings encryption set [--tier-enable {0,1,2,3}]
                                              [--tier-disable {0,1,2,3}]
```

`--tier-enable {0,1,2,3}`

Enable encryption for storage tiers. This option can be used multiple times.

`--tier-disable {0,1,2,3}`

Disable encryption for storage tiers. This option can be used multiple times.

Example:

```
# vinfra cluster settings encryption set --tier-enable 2
+-------+-------+
| Field | Value |
+-------+-------+
| tier0 | False |
| tier1 | False |
| tier2 | True  |
| tier3 | False |
+-------+-------+
```

This command enables encryption for the storage tier 2.

Managing automatic storage disk configuration

vinfra cluster settings automatic-disk-replacement show

Show automatic storage disk configuration:

```
usage: vinfra cluster settings automatic-disk-replacement show
```

Example:

```
# vinfra cluster settings automatic-disk-replacement show
+-------+-------+
| Field | Value |
+-------+-------+
| tier0 | True  |
| tier1 | False |
| tier2 | True  |
| tier3 | False |
+-------+-------+
```

This command shows the status of automatic disk configuration for each storage tier.

vinfra cluster settings automatic-disk-replacement set

Change automatic storage disk configuration:

```
usage: vinfra cluster settings automatic-disk-replacement set [--tier0 {on,off}]
                                                              [--tier1 {on,off}]
                                                              [--tier2 {on,off}]
                                                              [--tier3 {on,off}]
```

`--tier0 {on,off}`

Enable or disable automatic storage disk configuration for tier 0

`--tier1 {on,off}`

Enable or disable automatic storage disk configuration for tier 1

`--tier2 {on,off}`

Enable or disable automatic storage disk configuration for tier 2

`--tier3 {on,off}`

Enable or disable automatic storage disk configuration for tier 3

Example:

```
# vinfra cluster settings automatic-disk-replacement set \
--tier0 on --tier1 on --tier2 on --tier3 on
+-------+-------+
| Field | Value |
+-------+-------+
| tier0 | True  |
| tier1 | True  |
| tier2 | True  |
| tier3 | True  |
+-------+-------+
```

This command enables automatic storage disk configuration for all storage tiers.

Managing alerts

vinfra cluster alert list

List alert log entries:

```
usage: vinfra cluster alert list [--long] [--all]
```

`--long`

Enable access and listing of all fields of objects.

`--all`

Show both enabled and disabled alerts.

Example:

```
# vinfra cluster alert list --all
+----+-----------------+--------------------+----------+---------+
| id | type            | datetime           | severity | enabled |
+----+-----------------+--------------------+----------+---------+
|  1 | Network warning | 2018-08-30T18:02:14 | warning  | True    |
|  2 | Network warning | 2018-08-30T18:02:14 | warning  | True    |
|  3 | Network warning | 2018-08-30T18:02:14 | warning  | True    |
|  4 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
|  5 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
|  6 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
|  7 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
|  8 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
|  9 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
| 10 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
| 11 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
| 12 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
| 13 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
| 14 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
| 15 | Network warning | 2018-08-31T13:02:15 | warning  | True    |
+----+-----------------+--------------------+----------+---------+
```

This command lists all alerts in the log and shows whether they are enabled or disabled.

vinfra cluster alert show

Show details of the specified alert log entry:

```
usage: vinfra cluster alert show <alert>
```

`<alert>`

Alert ID

Example:

```
# vinfra cluster alert show 1
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| _type        | undefined_speed                      |
| cluster_id   |                                      |
| cluster_name |                                      |
| datetime     | 2018-08-30T18:02:14.855302+00:00     |
| details      | host: node001.vstoragedomain.        |
| enabled      | True                                 |
| group        | node                                 |
```



```
| host         | node001.vstoragedomain.              |
| id           | 1                                    |
| message      | Network interface "eth1" on node     |
|              | "node001.vstoragedomain." has an     |
|              | undefined speed                      |
| node_id      | 4f96acf5-3bc8-4094-bcb6-4d1953be7b55 |
| object_id    | eth1                                 |
| severity     | warning                              |
| suspended    |                                      |
| type         | Network warning                      |
+--------------+--------------------------------------+
```

This command shows the details of alert with ID 1.

vinfra cluster alert delete

Remove an entry from the alert log:

```
usage: vinfra cluster alert delete <alert>
```

`<alert>`

Alert ID

Example:

```
# vinfra cluster alert delete 1
+--------------+--------------------------------------+
| Field        | Value                                |
+--------------+--------------------------------------+
| _type        | undefined_speed                      |
| cluster_id   |                                      |
| cluster_name |                                      |
| datetime     | 2018-08-30T18:02:14.855302+00:00     |
| details      | host: node001.vstoragedomain.        |
| enabled      | True                                 |
| group        | node                                 |
| host         | node001.vstoragedomain.              |
| id           | 1                                    |
| message      | Network interface "eth1" on node     |
|              | "node001.vstoragedomain." has an     |
|              | undefined speed                      |
| node_id      | 4f96acf5-3bc8-4094-bcb6-4d1953be7b55 |
| object_id    | eth1                                 |
| severity     | warning                              |
| suspended    |                                      |
| type         | Network warning                      |
+--------------+--------------------------------------+
```

This command deletes the alert with the ID 1 from the log.

Managing audit log

vinfra cluster auditlog list

List all audit log entries:

```
usage: vinfra cluster auditlog list [--long]
```

`--long`

Enable access and listing of all fields of objects.

Example:

```
# vinfra cluster auditlog list
+----+----------+--------------------+-----------------------+---------------+
| id | username | type               | activity              | timestamp     |
+----+----------+--------------------+-----------------------+---------------+
|  1 | admin    | LoginUser          | User login            | <...>08:33:44 |
|  2 | admin    | CreateVLAN         | Create VLAN           | <...>12:34:18 |
|  3 | admin    | RemoveNetworkIface | Delete interface      | <...>13:26:40 |
|  4 | admin    | CreateNetworkRoles… | Create custom role set | <...>15:06:03 |
|  5 | admin    | RemoveNetworkRoles… | Remove custom role set | <...>15:39:31 |
|  6 | admin    | CreateNetworkRole  | Create custom role    | <...>15:58:50 |
|  7 | admin    | RemoveNetworkRole  | Remove custom role    | <...>16:20:22 |
+----+----------+--------------------+-----------------------+---------------+
```

This command lists the audit log entries.

vinfra cluster auditlog show

Show details of an audit log entry:

```
usage: vinfra cluster auditlog show <auditlog>
```

`<auditlog>`

Audit log ID

Example:

```
# vinfra cluster auditlog show 1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| activity     | User login                       |
| cluster_id   |                                  |
| cluster_name |                                  |
| component    | Users                            |
```

```
| details      | []                                |
| id           | 1                                 |
| message      | User "admin" login                |
| node_id      |                                   |
| result       | success                           |
| task_id      | 5686556295049300                  |
| timestamp    | 2018-09-07T08:33:44.175797+00:00 |
| type         | LoginUser                         |
| username     | admin                             |
+--------------+-----------------------------------+
```

This command shows the details of the audit log entry with the ID 1.

Managing cluster password

vinfra cluster password show

Show the storage cluster password:

```
usage: vinfra cluster password show
```

Example:

```
# vinfra cluster password show
+----------+----------+
| Field    | Value    |
+----------+----------+
| id       | 1        |
| name     | cluster1 |
| password | aR2oRG   |
+----------+----------+
```

This command shows the storage cluster password.

vinfra cluster password reset

Set a new storage cluster password:

```
usage: vinfra cluster password reset
```

Example:

```
# vinfra cluster password reset
Password:
+----------+----------+
| Field    | Value    |
+----------+----------+
| id       | 1        |
```

```
| name     | cluster1 |
| password | 1q2w3e   |
+----------+----------+
```

This command sets a new password for the storage cluster.

Sending problem reports

Generate and send a problem report:

```
usage: vinfra cluster problem-report [--email <email>]
                                     [--description <description>] [--send]
```

`--email <email>`

Contact email address

`--description <description>`

Problem description

`--send`

Generate the problem report archive and send it to the technical support team

Example:

```
# vinfra cluster problem-report --email test@example.com --description "Test report" --
send
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 8bcfb92f-f02b-4de8-8e44-3426047630e3 |
+---------+--------------------------------------+
```

This commands creates a task to send a problem report with the description “Test report” to the technical support team and use test@example.com as a reply-to address. Note the problem report ID in the task details. You will need to mention it in the support ticket.

Task outcome:

```
+---------+-------------------------------------------------------------+
| Field   | Value                                                       |
+---------+-------------------------------------------------------------+
| details |                                                             |
| name    | backend.presentation.reports.tasks.ReportProblemTask        |
| result  | id: '1001923113'                                            |
|         | path: /var/cache/problem-reports/report-<...>.391329.tar.gz |
| state   | success                                                     |
| task_id | 37d5c13a-001c-4789-8242-96825a17deda                        |
+---------+-------------------------------------------------------------+
```

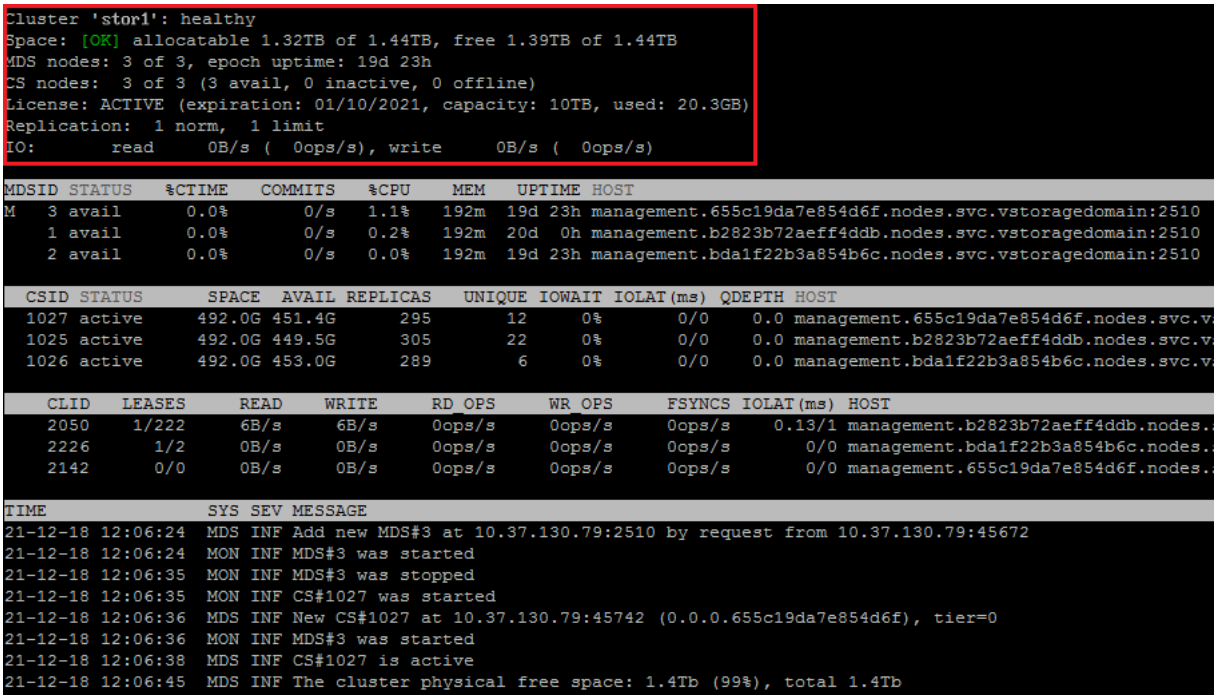
Monitoring the storage cluster

Monitoring the storage cluster is very important because it allows you to check the status and health of all computers in the cluster and react as necessary.

The main command for monitoring is `vstorage -c <cluster_name> top`. It invokes a text user interface that you can control with keys (press **h** for help).

Monitoring general storage cluster parameters

By monitoring general parameters, you can get detailed information about all components of the storage cluster, its overall status and health. To display this information, use the `vstorage -c <cluster_name> top` command. For example:



The command above shows detailed information about the `stor1` cluster. The general parameters (highlighted in red) are the following:

Cluster

Overall status of the cluster:

Healthy

All chunk servers in the cluster are active.

Unknown

There is not enough information about the cluster state (for example, because the master MDS server was elected a while ago).

Degraded

Some of the chunk servers in the cluster are inactive.

Failure

The cluster has too many inactive chunk servers; the automatic replication is disabled.

SMART warning

One or more physical disks attached to cluster nodes are in pre-failure condition. For details, refer to "Monitoring physical disks" (p. 296).

Space

Amount of disk space in the cluster:

Free

Free physical disk space in the cluster.

Allocatable

Amount of logical disk space available to clients. Allocatable disk space is calculated on the basis of the current replication parameters and free disk space on chunk servers. It may also be limited by license.

Note

For more information on monitoring and understanding disk space usage in clusters, refer to "Understanding disk space usage" (p. 286).

MDS nodes

Number of active MDS servers as compared to the total number of MDS servers configured for the cluster.

Epoch time

Time elapsed since the MDS master server election.

CS nodes

Number of active chunk servers as compared to the total number of chunk servers configured for the cluster.

In parentheses, you can see the additional information on these chunk servers:

- Active chunk servers (avail.) that are currently up and running in the cluster.
- Inactive chunk servers (inactive) that are temporarily unavailable. A chunk server is marked as inactive during its first 5 minutes of inactivity.
- Offline chunk servers (offline) that have been inactive for more than 5 minutes. A chunk server changes its state to offline after 5 minutes of inactivity. Once the state is changed to offline, the cluster starts replicating data to restore the chunks that were stored on the offline chunk server.

License

Key number under which the license is registered on the Key Authentication server and license state.

Replication

Replication settings. The normal number of chunk replicas and the limit after which a chunk gets blocked until recovered.

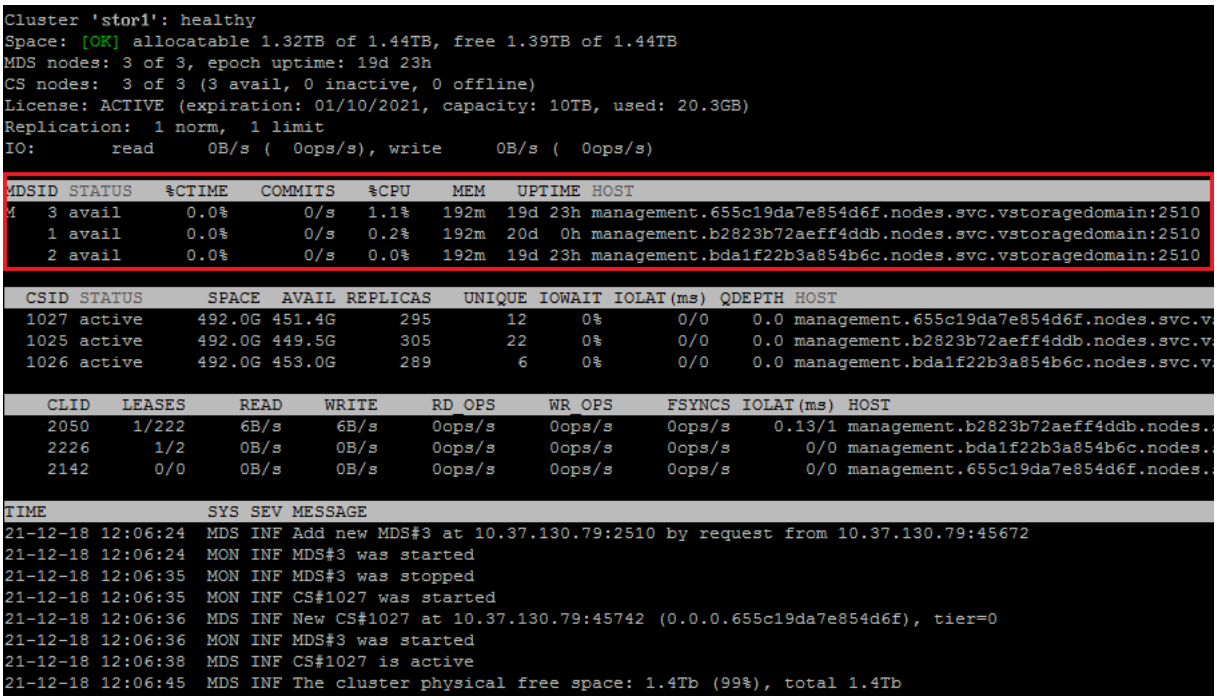
IO

Disk IO activity in the cluster:

- Speed of read and write I/O operations, in bytes per second.
- Number of read and write I/O operations per second.

Monitoring metadata servers

MDS servers are a critical component of any storage cluster, and monitoring the health and state of MDS servers is a crucial task. To monitor MDS servers, use the `vstorage -c <cluster_name> top` command. For example:



The command above shows detailed information about the `stor1` cluster. The monitoring parameters for MDS servers (highlighted in red) are the following:

MDSID

MDS server identifier (ID).

The letter “M” before ID, if present, means that the given server is the master MDS server.

STATUS

MDS server status.

%CTIME

Total time the MDS server spent writing to the local journal.

COMMITS

Local journal commit rate.

%CPU

MDS server activity time.

MEM

Amount of physical memory the MDS server uses.

UPTIME

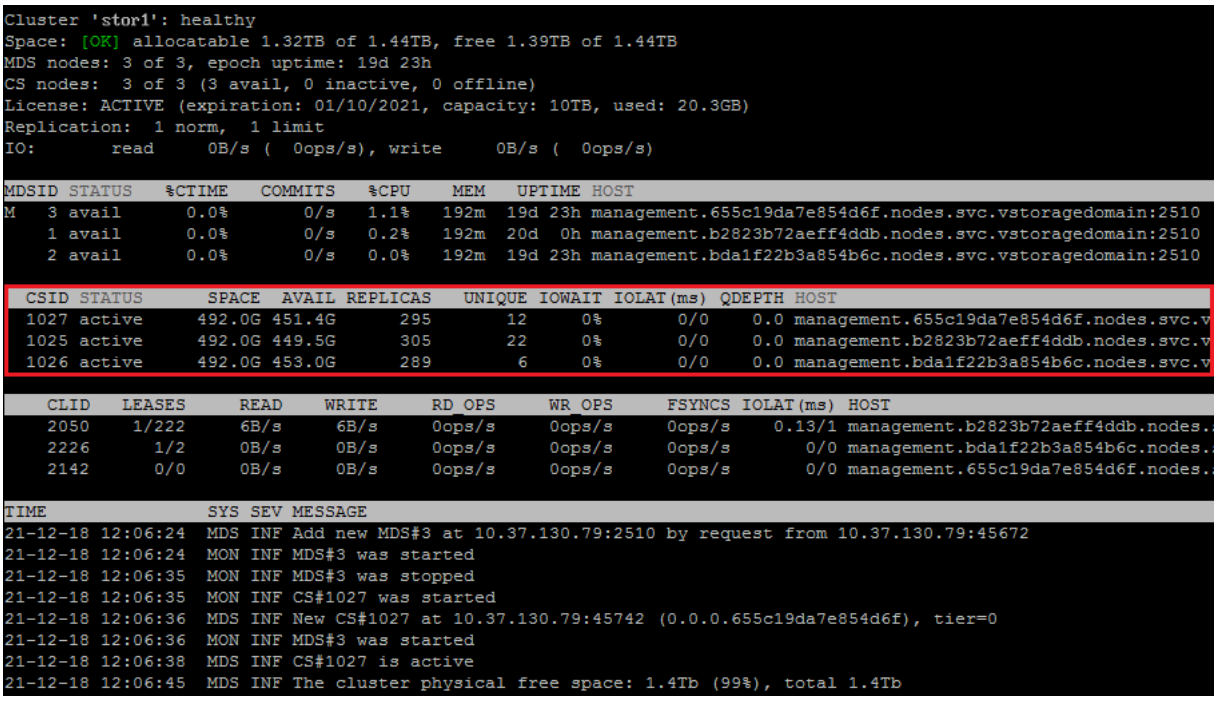
Time elapsed since the last MDS server start.

HOST

MDS server hostname or IP address.

Monitoring chunk servers

By monitoring chunk servers, you can keep track of the disk space available in the storage cluster. To monitor chunk servers, use the `vstorage -c <cluster_name> top` command. For example:



The command above shows detailed information about the `stor1` cluster. The monitoring parameters for chunk servers (highlighted in red) are the following:

CSID

Chunk server identifier (ID).

STATUS

Chunk server status:

active

The chunk server is up and running.

failed

The chunk server process is running but a problem has occured with the CS disk.

inactive

The chunk server is temporarily unavailable. A chunk server is marked as inactive during its first 5 minutes of inactivity.

offline

The chunk server is inactive for more than 5 minutes. After the chunk server goes offline, the cluster starts replicating data to restore the chunks that were stored on the affected chunk server.

dropped

The chunk server was removed by the administrator.

maintenance

The node that the chunk server is located on is in maintenance.

ill

The chunk server experiences slowdown and degrades the cluster performance. The chunk server is isolated from the cluster I/O.

SPACE

Total amount of disk space on the chunk server.

AVAIL

Available disk space on the chunk server.

REPLICAS

Number of replicas stored on the chunk server.

UNIQUE

Number of chunks that do not have replicas.

IOWAIT

Percentage of time spent waiting for I/O operations being served.

IOLAT

Average/maximum time, in milliseconds, the client needed to complete a single IO operation during the last 20 seconds.

QDEPTH

Average chunk server I/O queue depth.

HOST

Chunk server hostname or IP address.

FLAGS

The following flags may be shown for active chunk servers:

J

The CS uses a write journal.

C

Checksumming is enabled for the CS. Checksumming lets you know when a third party changes the data on the disk.

D

Direct I/O, the normal state for a CS without a write journal.

c

The chunk server's write journal is clean, there is nothing to commit from the write journaling SSD to the HDD where the CS is located.

Understanding disk space usage

Usually, you get the information on how disk space is used in your cluster with the `vstorage top` command. This command displays the following disk-related information: total space, free space, and allocatable space. For example:

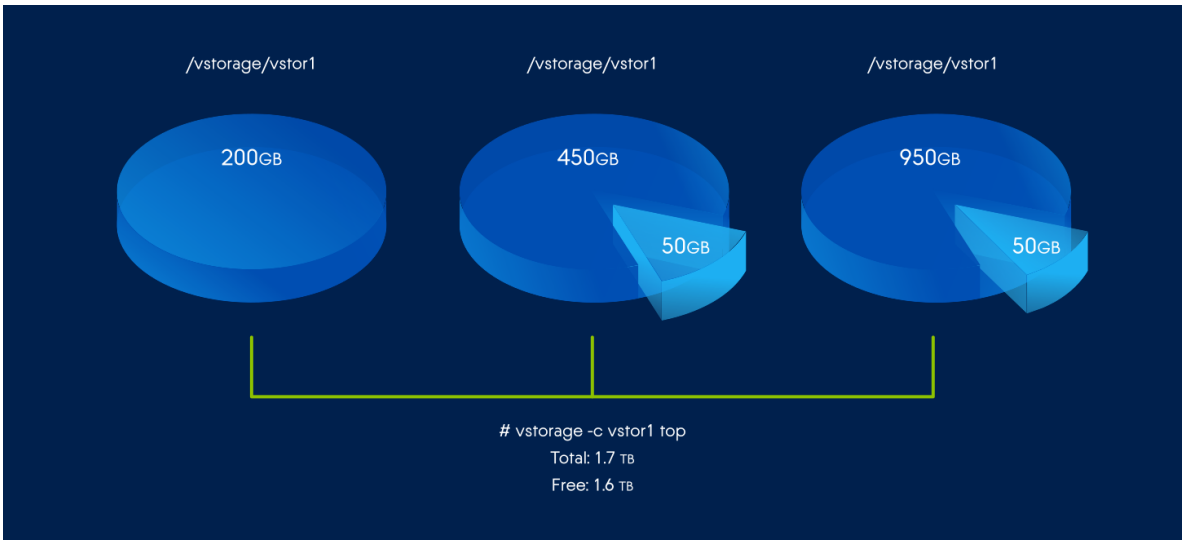
```
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 180GB of 200GB, free 1.6TB of 1.7TB
<...>
```

In this command output:

- `1.7TB` is the total disk space in the `stor1` cluster. The total disk space is calculated on the basis of used and free disk space on all partitions in the cluster. Used disk space includes the space occupied by all data chunks and their replicas plus the space occupied by any other files stored on the cluster partitions.

  Let us assume that you have a 100 GB partition and 20 GB on this partition are occupied by some files. Now if you set up a chunk server on this partition, this will add 100 GB to the total disk space of the cluster, though only 80 GB of this disk space will be free and available for storing data chunks.
- `1.6TB` is the free disk space in the `stor1` cluster. Free disk space is calculated by subtracting the disk space occupied by data chunks and any other files on the cluster partitions from the total disk space.

  For example, if the amount of free disk space is 1.6 TB and the total disk space is 1.7 TB, this means that about 100 GB on the cluster partitions are already occupied by some files.



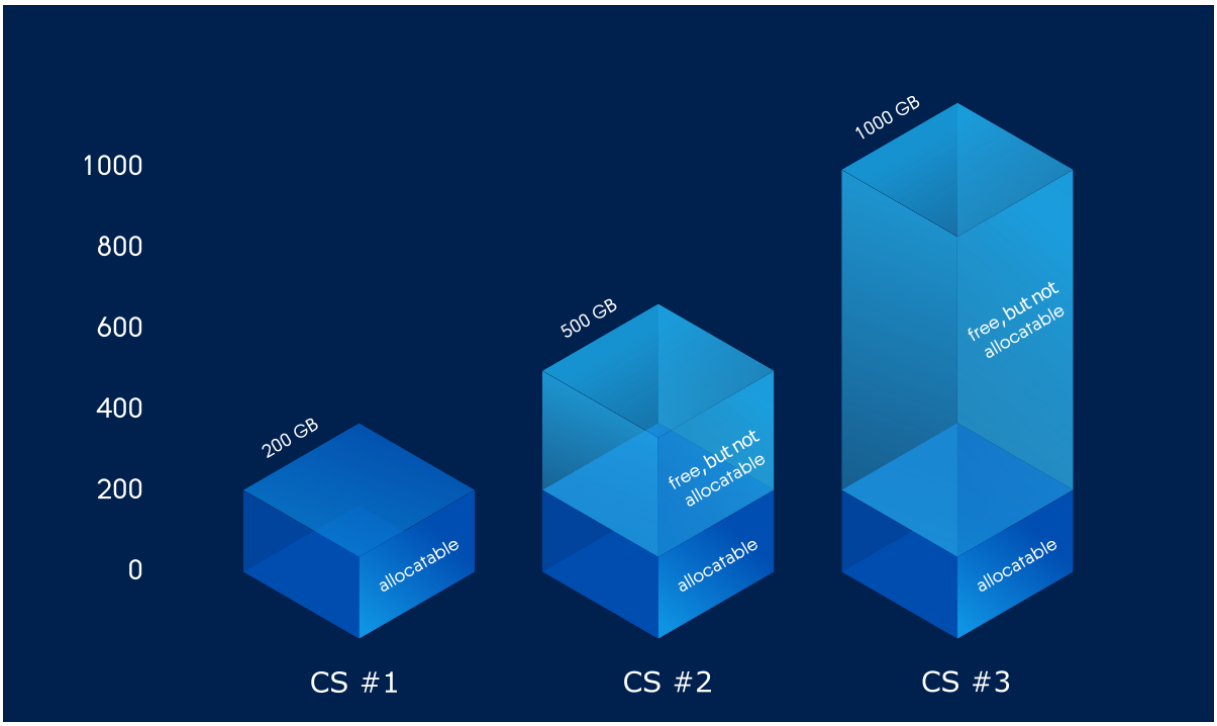
- `allocatable 180GB of 200GB` is the amount of free disk space that can used for storing data chunks. Refer to "Understanding allocatable disk space" (p. 287) for details.

Understanding allocatable disk space

When monitoring disk space information in the cluster, you also need to pay attention to the space reported by the `vstorage top` utility as *allocatable*. Allocatable space is the amount of disk space that is free and can be used for storing user data. Once this space runs out, no data can be written to the cluster.

Calculation of allocatable disk space is illustrated on the following example:

- The cluster has 3 chunk servers. The first chunk server has 200 GB of disk space, the second one — 500 GB, and the third one — 1 TB.
- The default replication factor of 3 is used in the cluster, meaning that each data chunk must have 3 replicas stored on three different chunk servers.



In this example, the available disk space is 200 GB, which equals the amount of disk space on the smallest chunk server:

```
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 180GB of 200GB, free 1.6TB of 1.7TB
<...>
```

In this cluster configuration each server is set to store one replica for each data chunk. So once the disk space on the smallest chunk server (200 GB) runs out, no more chunks in the cluster can be created until a new chunk server is added or the replication factor is decreased.

If the replication factor changes to 2, the `vstorage top` command will report the available disk space as 700 GB:

```
# vstorage set-attr -R /mnt/vstorage replicas=2:1
# vstorage -c stor1 top
connected to MDS#1
Cluster 'stor1': healthy
Space: [OK] allocatable 680GB of 700GB, free 1.6TB of 1.7TB
<...>
```

The available disk space has increased because now only 2 replicas are created for each data chunk and new chunks can be made even if the smallest chunk server runs out of space (in this case, replicas will be stored on a bigger chunk server).

Allocatable disk space may also be limited by license.

Viewing space occupied by data chunks

To view the total amount of disk space occupied by all user data in the cluster, run the `vstorage top` command and press the **V** key on your keyboard. Once you do this, your command output should look like the following:

```
# vstorage -c stor1 top
Cluster 'stor1': healthy
Space: [OK] allocatable 1.32TB of 1.44TB, free 1.39TB of 1.44TB
MDS nodes: 3 of 3, epoch uptime: 19d 23h, cluster version: 128
CS nodes:  3 of 3 (3 avail, 0 inactive, 0 offline), storage version: 128
License: ACTIVE (expiration: 01/10/2021, capacity: 10TB, used: 20.3GB)
Replication:  1 norm,  1 limit
Chunks: [OK] 323 (100%) healthy,  0 (0%) standby,  0 (0%) degraded,  0 (0%) urgent,
             0 (0%) blocked,  0 (0%) pending,  0 (0%) offline,  0 (0%) replicating,
             0 (0%) overcommitted,  0 (0%) deleting,  0 (0%) void
FS:  20.3GB in 757 files, 757 inodes,  244 file maps,  323 chunks,  889 chunk replicas
IO:       read     0B/s (  0ops/s), write     0B/s (  0ops/s)
IO total: read   37.1GB ( 473Kops), write  133.7GB ( 4.7Mops)
Repl IO:  read     0B/s, write:     0B/s
Sync rate:   0ops/s, datasync rate:   0ops/s
IO QDEPTH: 0.0 aver, 0.0 max
<...>
```

The **FS** field shows the size of all user data in the cluster without consideration for replicas.

Exploring chunk states

The following is a list of all possible chunk states.

Healthy

Number and percentage of chunks that have enough active replicas. The normal state of chunks.

Offline

Number and percentage of chunks all replicas of which are offline. Such chunks are completely inaccessible for the cluster and cannot be replicated, read from or written to. All requests to an offline chunk are frozen until a CS that stores that chunk's replica goes online.

Get offline chunk servers back online as fast as possible, to avoid losing data.

Blocked

Number and percentage of chunks that have fewer active replicas than the set minimum amount. Write requests to a blocked chunk are frozen until it has at least the set minimum amount of replicas. Read requests to blocked chunks are allowed, however, as they still have some active replicas left. Blocked chunks have a higher replication priority than degraded chunks.

Having blocked chunks in the cluster increases the risk of losing data, so postpone any maintenance on working cluster nodes and get offline chunk servers back online as fast as possible.

Degraded

Number and percentage of chunks whose active replicas are few, but not below the set minimum. Such chunks can be read from and written to. However, in the latter case, a degraded chunk becomes urgent.

Replicating

Number and percentage of chunks which are being replicated. Write operations on such chunks are frozen until replication ends.

Void

Number and percentage of chunks that have been allocated but never used yet. Such chunks contain no data. It is normal to have some void chunks in the cluster.

Pending

Number and percentage of chunks that must be replicated immediately. For a write request from client to a chunk to complete, the chunk must have at least the set minimum amount of replicas. If it does not, the chunk is blocked and the write request cannot be completed. As blocked chunks must be replicated as soon as possible, the cluster places them in a special high-priority replication queue and reports them as pending.

Urgent

Number and percentage of chunks which are degraded and have non-identical replicas. Replicas of a degraded chunk may become non-identical if some of them are not accessible during a write operation. As a result, some replicas happen to have the new data while some still have the old data. The latter are dropped by the cluster as fast as possible. Urgent chunks do not affect information integrity as the actual data is stored in at least the set minimum amount of replicas.

Overcommitted

Number and percentage of chunks that have more replicas than normal. Usually these chunks appear after the normal number of replicas has been lowered or a lot of data has been deleted. Extra replicas are eventually dropped, however, this process may slow down during replication.

Deleting

Number and percentage of chunks queued for deletion.

Monitoring disk health

Important
This functionality is disabled in clusters deployed on virtual machines.

You can monitor node disks by using the `vstorage-disks-monitor` service. This service runs on every management node and queries chunk server (CS) metrics from the Prometheus service for further analysis. `vstorage-disks-monitor` detects CSes that experiences slowdown and marks them as ill (slow). To avoid degrading the cluster performance, slow CSes are fenced from the cluster I/O.

The service also calculates disk health, in percent, based on each metric weight. Weights can be configured in the `/etc/disks-monitor/analyzers.yml` configuration file. The service logs are stored in `/var/log/disks-monitor/disks-monitor.log`.

The service can work in two modes:

- As a daemon if you use the `vstorage-disks-monitor sidecar` command
- As a tool for listing disk statuses and alerts if you run `vstorage-disks-monitor health` and `vstorage-disks-monitor alerts`

You can disable fencing ill CSes by running the `vstorage-disks-monitor sidecar --fencing.enable` command.

Disk-related metrics in Prometheus

The Prometheus service stores the following disk-related metrics:

CS-related metrics	
`csd_io_op_time_seconds`	Mean time per I/O request
`master:mdsd_cs_status`	CS status on master MDS
Disk-related metrics in `/proc/diskstats`	
`node_disk_read_time_seconds`	Total time, in seconds, spent on read requests
`node_disk_reads_completed`	Total number of completed read requests
`node_disk_write_time_seconds`	Total time, in seconds, spent on write requests
`node_disk_writes_completed`	Total number of completed write requests
S.M.A.R.T. metrics	
`smart_device_smart_healthy`	S.M.A.R.T. status is healthy
`smart_reallocated_sector_ct`	Total number of reallocated disk sectors (05)

`smart_reported_uncorrect`	Total number of errors that could not be recovered using hardware ECC (187)
`smart_command_timeout`	Total number of aborted operations due to a timeout (188)
`smart_current_pending_sector`	Total number of unstable sectors (197)
`smart_offline_uncorrectable`	Total number of uncorrectable errors when reading/writing a sector (198)
`smart_media_wearout_indicator`	Media Wearout Indicator for SSD (233)
`smart_nvme_intel_wear_leveling`	Media Wearout Indicator for Intel NVME (233)
`smart_scsi_read_errors_uncorrected`	Total number of uncorrectable errors when reading a sector
`smart_scsi_reallocated_sector_ct`	Total number of reallocated disk sectors
`smart_scsi_verify_errors_uncorrected`	Total number of uncorrectable errors when verifying a sector
`smart_scsi_write_errors_uncorrected`	Total number of uncorrectable errors when writing a sector
Kernel SCSI errors	
`kernel_scsi_failures_total`	Total number of SCSI failures reported by the kernel
Disk health metric from `vstorage-disks-monitor`	
`diskmon_cs_disk_health`	Disk health reported by the `vstorage-disks-monitor` service. Possible values are 0.0–1.0. The 1.0 value means that the disk is 100% healthy.

Calculating disk health

The core part of calculating disk health are analyzers. Each analyzer calculates disks health based on its own algorithm. The overall disk health is a product of disk health values from all of the analyzers.

For example:

- According to the S.M.A.R.T. attributes, the disk health is 0.9.
- The slow disk analyzer reports that the disk health is 0.4.
- The slow CS analyzer reports that the disk health is 0.5.
- According to SCSI errors, the disk health is 1.0.

The overall disk health is calculated as 0.9*0.4*0.5*1.0 and equals 0.18 or 18%.

S.M.A.R.T. attributes

The following table contains the S.M.A.R.T. attributes that affect the health value:

ID	S.M.A.R.T. attribute	Weight[1]	Limit[2], in percent
05	Reallocated Sectors Count	2	70
187	Reported Uncorrectable Errors	1	70
188	Command Timeout	1	20
197	Current Pending Sectors Count	2	70
198	Offline uncorrectable Sectors Count	2	70
233	Media Wearout Indicator	1	100

The disk health is calculated by using the following formula:

```
Disk health (%) = K * Π (100% - D)
```

where:

- `K` is the reduction coefficient. A disk is considered less healthy if it reports more then one type of S.M.A.R.T. errors. The coefficient formula is `0.8^({Number of S.M.A.R.T. attributes with error} - 1)`. Possible values are 0–1.
- `Π` is a product of minimums calculated for each critical S.M.A.R.T. attribute.
- 100% is the initial health of the disk.
- `D` is a minimum from the limit and attribute value with its weight. Its formula is `(min(limit, attribute_value * weight))`.
- `limit` is a limit of each critical S.M.A.R.T. attribute.
- `attribute_value` is the current attribute value.
- `weight` is weight of each critical S.M.A.R.T. attribute.

For example:

- Reallocated Sectors Count: attribute value = 30, weight = 2, limit = 70
- Command Timeout: attribute value = 23, weight = 1, limit = 20
- K = 0.8 * (2–1) = 0.8

The S.M.A.R.T. disk health is calculated as follows: 0.8 * (100% – (min(30*2, 70))) * (100% - min(23*1, 20))) = 0.8 * 0.4 * 0.8 = 0.256 (26%)

---

[1] Defines by how many percent the attribute value decreases the total disk health value.

[2] Defines the maximum impact the attribute may have on the total disk health.

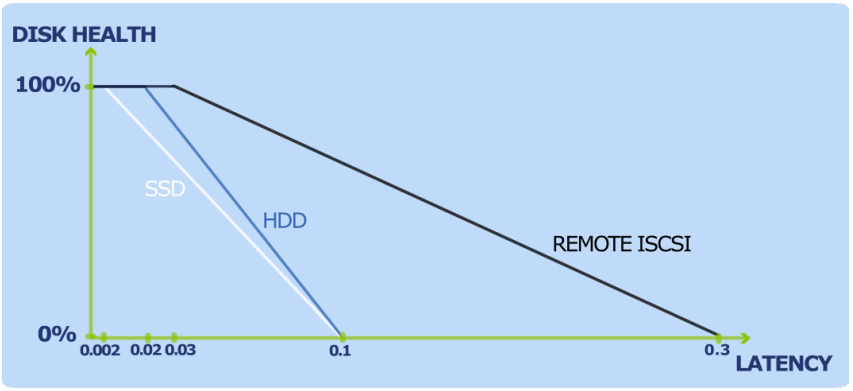
Slow disk and slow CS analyzers

Slow disk and CS analyzers calculate disk health according to the average I/O latency over time (15 minutes).

The following table shows the default thresholds:

Analyzer	OK latency[1], in seconds	FATAL latency[2], in seconds
Slow CS	0.03	0.3
Slow HDD Disk	0.02	0.1
Slow SSD Disk	0.002	0.1

If disk latency is less than **OK latency**, the disk health is considered to be 100%. If disk latency exceeds **FATAL latency**, the disk health is considered to be 0%. Disk latency that lies within these two thresholds will vary linearly from 100% to 0%.



When disk health becomes 0%, the service generates an alert and marks this CS as ill. Such a CS does not trigger automatic replication but is no longer available for chunk allocation.

SCSI errors

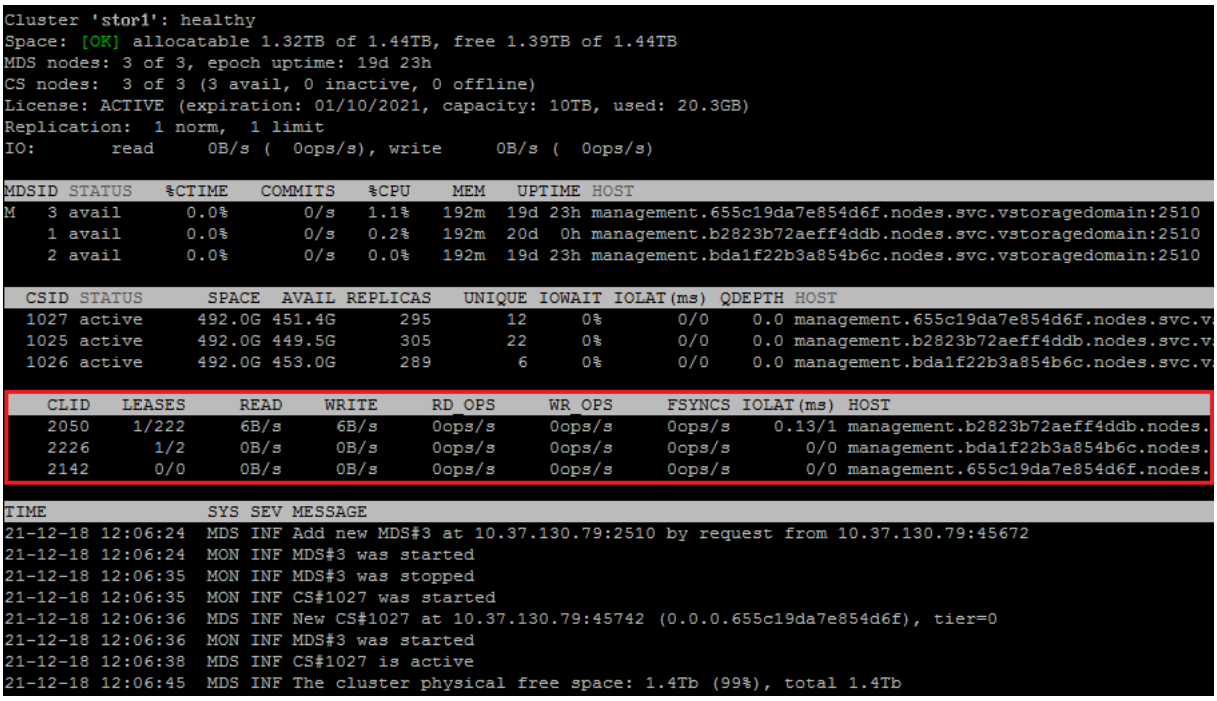
By default, each SCSI failure decrease the overall disk heath by 4%. The maximum health impact is set to 70.

Monitoring clients

By monitoring clients, you can check the status and health of servers that you use to access virtual machines. To monitor clients, use the `vstorage -c <cluster_name> top` command. For example:

[1] Maximum latency value to consider disk health to be 100%.

[2] Latency value to consider disk health to be 0%.



The command above shows detailed information about the `stor1` cluster. The monitoring parameters for clients (highlighted in red) are the following:

CLID

Client identifier (ID).

LEASES

Average number of files opened for reading/writing by the client and not yet closed, for the last 20 seconds.

READ

Average rate, in bytes per second, at which the client reads data, for the last 20 seconds.

WRITE

Average rate, in bytes per second, at which the client writes data, for the last 20 seconds.

RD_OPS

Average number of read operations the client made per second, for the last 20 seconds.

WR_OPS

Average number of write operations the client made per second, for the last 20 seconds.

FSYNCS

Average number of sync operations the client made per second, for the last 20 seconds.

IOLAT

Average/maximum time, in milliseconds, the client needed to complete a single IO operation, for the last 20 seconds.

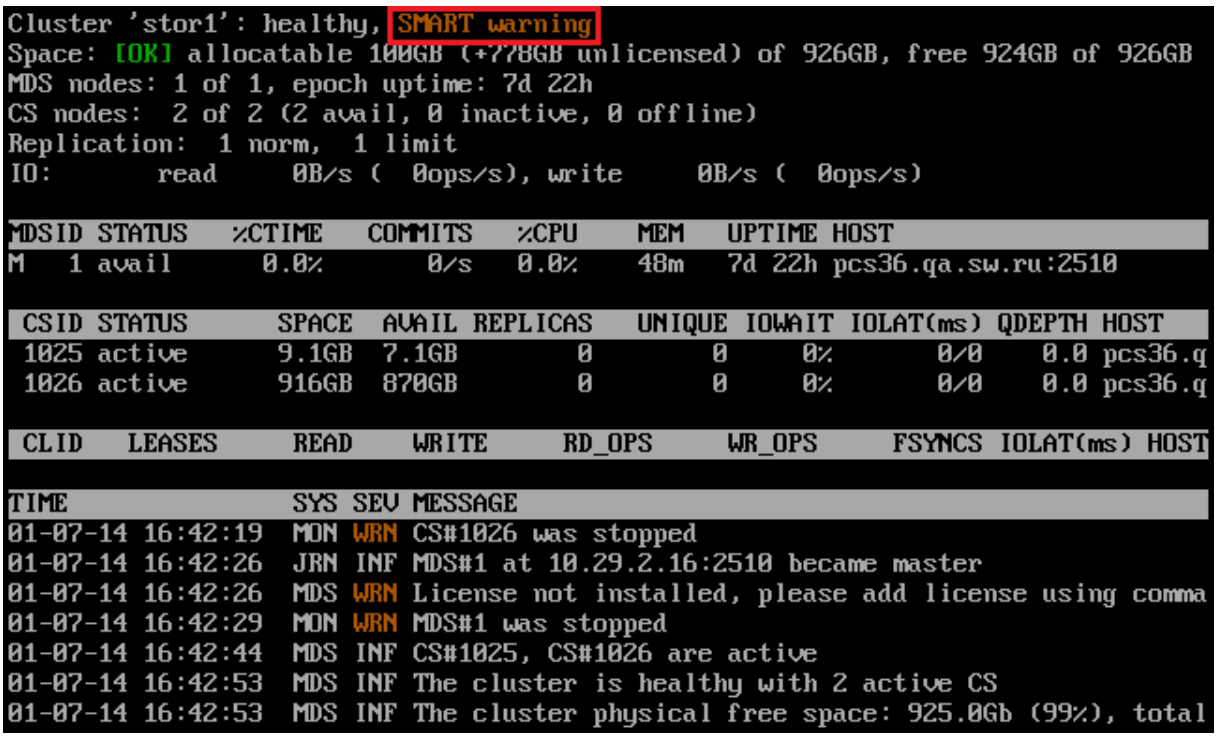
HOST

Client hostname or IP address.

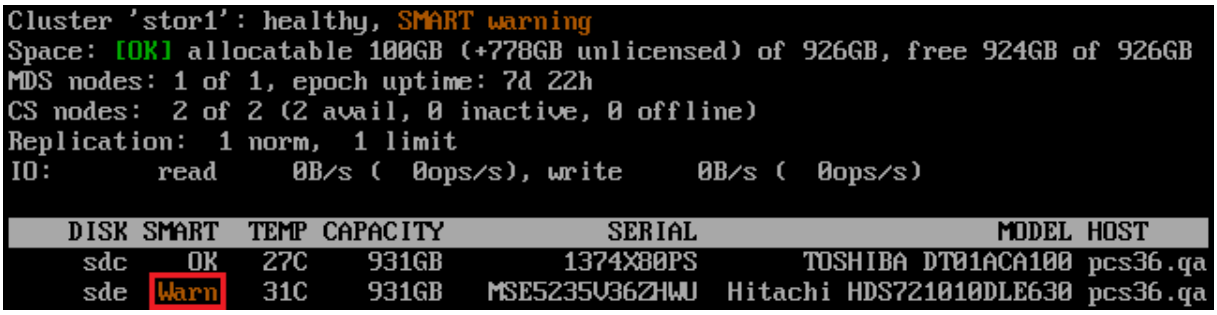
Monitoring physical disks

The S.M.A.R.T. status of physical disks is monitored by the `smartctl` tool installed along with Virtuozzo Hybrid Infrastructure. For it to work, S.M.A.R.T. functionality must be enabled in the node's BIOS. The tool is run every 10 minutes as a `cron` job also added during installation. The `smartctl` tool polls all physical disks attached to nodes in the cluster, including caching and journaling SSDs, and reports the results to the MDS server.

You can view disk poll results for the last 10 minutes in the output of the `vstorage top` command. For example:



If the **SMART warning** message is shown in the main table, one of the physical disks is in pre-failure condition according to S.M.A.R.T. Press **d** to switch to the disks table to see more details. For example:



The disks table shows the following parameters:

DISK

Disk name assigned by operating system.

SMART

Disk's S.M.A.R.T. status:

OK

The disk is healthy.

Warn

The disk is in pre-failure condition. Pre-failure condition means that at least one of these S.M.A.R.T. counters is nonzero:

- Reallocated Sector Count
- Reallocated Event Count
- Current Pending Sector Count
- Offline Uncorrectable

TEMP

Disk temperature in Celsius.

CAPACITY

Disk capacity.

SERIAL

Disk serial number.

MODEL

Disk model.

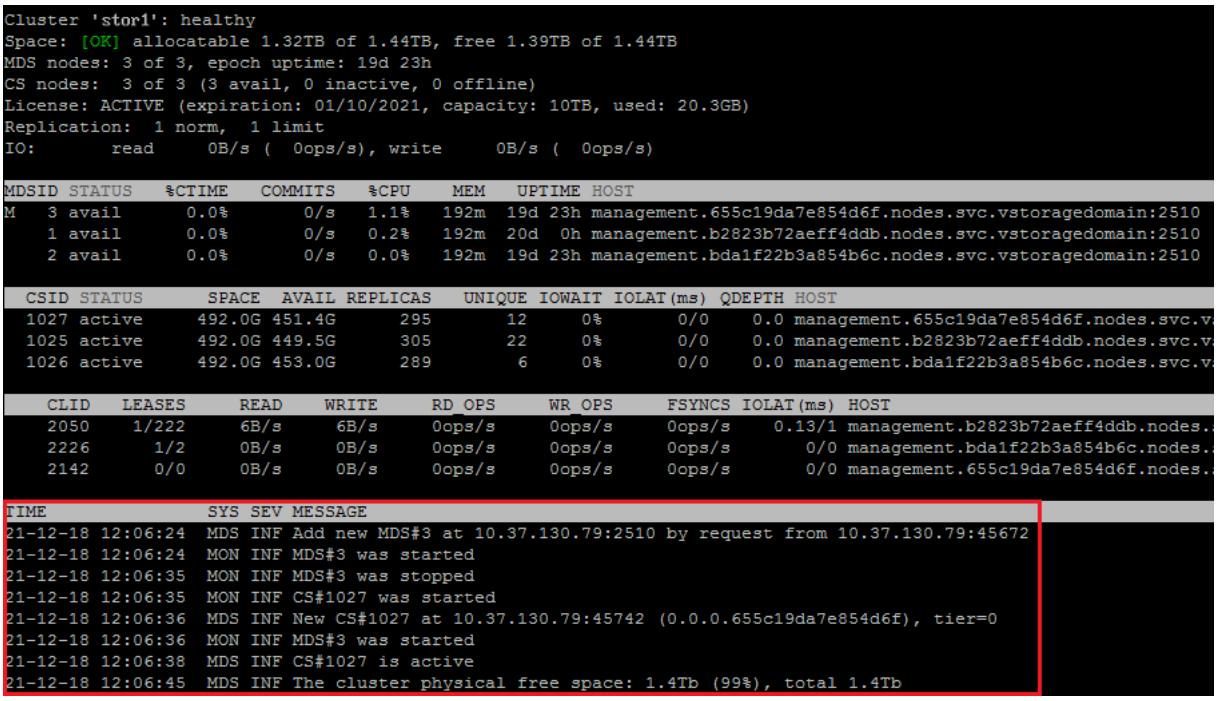
HOST

Disk's host address.

To disable S.M.A.R.T. disk monitoring, delete the corresponding `cron` job.

Monitoring event logs

You can use the `vstorage -c <cluster_name> top` utility to monitor significant events happening in the storage cluster. For example:



The command above shows the latest events in the `stor1` cluster. The information on events (highlighted in red) is given in a table with the following columns:

TIME

Time of event.

SYS

Component of the cluster where the event happened (e.g., MDS for an MDS server or JRN for local journal).

SEV

Event severity.

MESSAGE

Event description.

The following table lists basic events displayed when you run the `vstorage top` utility.

Basic events

Event	Severity	Description
MDS#<N> (<addr>:<port>) lags behind for more than 1000 rounds	JRN err	Generated by the MDS master server when it detects that MDS#<N> is stale. This message may indicate that some MDS server is very slow and lags behind.
MDS#<N> (<addr>:<port>) didn't accept commits for *M* sec	JRN err	Generated by the MDS master server if MDS#<N> did not accept commits for *M* seconds. MDS#<N> gets marked as stale. This message may indicate that the MDS service on MDS#<N> is

Event	Severity	Description
		experiencing a problem. The problem may be critical and should be resolved as soon as possible.
MDS#<N> (<addr>:<port>) state is outdated and will do a full resync	JRN err	Generated by the MDS master server when MDS#<N> will do a full resync. MDS#<N> gets marked as stale. This message may indicate that some MDS server was too slow or disconnected for such a long time that it is not really managing the state of metadata and has to be resynchronized. The problem may be critical and should be resolved as soon as possible.
MDS#<N> at <addr>:<port> became master	JRN info	Generated every time a new MDS master server is elected in the cluster. Frequent changes of MDS masters may indicate poor network connectivity and may affect the cluster operation.
The cluster is healthy with *N* active CS	MDS info	Generated when the cluster status changes to healthy or when a new MDS master server is elected. This message indicates that all chunk servers in the cluster are active and the number of replicas meets the set cluster requirements.
The cluster is degraded with *N* active, *M* inactive, *K* offline CS	MDS warn	Generated when the cluster status changes to degraded or when a new MDS master server is elected. This message indicates that some chunk servers in the cluster are • inactive, that is, do not send any registration messages, or • offline, that is, have been inactive for longer than `mds.wd.offline_tout`, which is 5 min by default.
The cluster failed with *N* active, *M* inactive, *K* offline CS (mds.wd.max_offline_cs=<n>)	MDS err	Generated when the cluster status changes to failed or when a new MDS master server is elected. This message indicates that the number of offline chunk servers exceeds `mds.wd.max_offline_cs`, which is 2 by default. When the cluster fails, the automatic replication is not scheduled any more. So the cluster administrator must take action to either repair failed chunk servers or increase `mds.wd.max_offline_cs`. Setting this value to 0 disables the failed mode completely.
The cluster is filled up to <N>%	MDS info/warn	Shows the current space usage in the cluster. A warning is generated if the disk space consumption equals or exceeds 80%. It is important to have spare disk space for data replicas if one of the chunk servers fails.
Replication started, N chunks are	MDS info	Generated when the cluster starts automatic data replication to recover the missing replicas.

Event	Severity	Description
queued		
Replication completed	MDS info	Generated when the cluster finishes automatic data replication.
CS#<N> has reported hard error on *path*	MDS warn	Generated when the chunk server CS#<N> detects disk data corruption. You are recommended to check the hardware for errors and replace corrupted disks as soon as possible.
CS#<N> has not registered during the last *T* sec and is marked as inactive/offline	MDS warn	Generated when the chunk server CS#<N> has been unavailable for a while. In this case, the chunk server first gets marked as inactive. After 5 minutes, the state is changed to offline, which starts automatic replication of data to restore the replicas that were stored on the offline chunk server.
Failed to allocate *N* replicas for '*path*' by request from <addr>:<port> - *K* out of *M* chunks servers are available	MDS warn	Generated when the cluster cannot allocate chunk replicas, for example, when it runs out of disk space.
Failed to allocate *N* replicas for '*path*' by request from <addr>:<port> since only *K* chunk servers are registered	MDS warn	Generated when the cluster cannot allocate chunk replicas because not enough chunk servers are registered in the cluster.

Monitoring replication parameters

When you configure replication parameters, keep in mind that the new settings do not come into effect immediately. For example, increasing the default replication parameter for data chunks may take some time to complete, depending on the new value of this parameter and the number of data chunks in the cluster.

To check that the new replication parameters have been successfully applied to your cluster, do the following:

1. Run the `vstorage -c <cluster_name> top` command.
2. Press **V** to display additional information about the cluster. Typical command output may look like this:

```
# vstorage -c stor1 top
Cluster 'stor1': healthy
```



```
Space: [OK] allocatable 448.6GB of 492.0GB, free 1.39TB of 1.44TB
MDS nodes: 3 of 3, epoch uptime: 20d  0h
CS nodes:  3 of 3 (3 avail, 0 inactive, 0 offline)
License: ACTIVE (expiration: 01/10/2021, capacity: 10TB, used: 20.3GB)
Replication:  3 norm,  2 limit
Chunks: [Warning] 187 (57%) healthy,  0 (0%) standby,  0 (0%) degraded,  135 (41%)
urgent,
             0 (0%) blocked,  0 (0%) pending,  0 (0%) offline,  1 (0%) replicating,
             0 (0%) overcommitted,  0 (0%) deleting,  0 (0%) void
IO:       read     0B/s (  0ops/s), write  106KB/s (  7ops/s)
<...>
```

3. Check the **Chunks** field for the following:
   - When decreasing the replication parameters, look for chunks that are in the overcommitted or deleting state. If the replication process is complete, no chunks with these states should be present in the output.
   - When increasing the replication parameters, look for chunks that are in the blocked or urgent state. If the replication process is complete, no chunks with these states should be present in the output. Besides, when the process is still in progress, the value of the healthy parameter is less than 100%.

   For more information on available chunk statuses, refer to "Exploring chunk states" (p. 289).

Accessing storage clusters via iSCSI

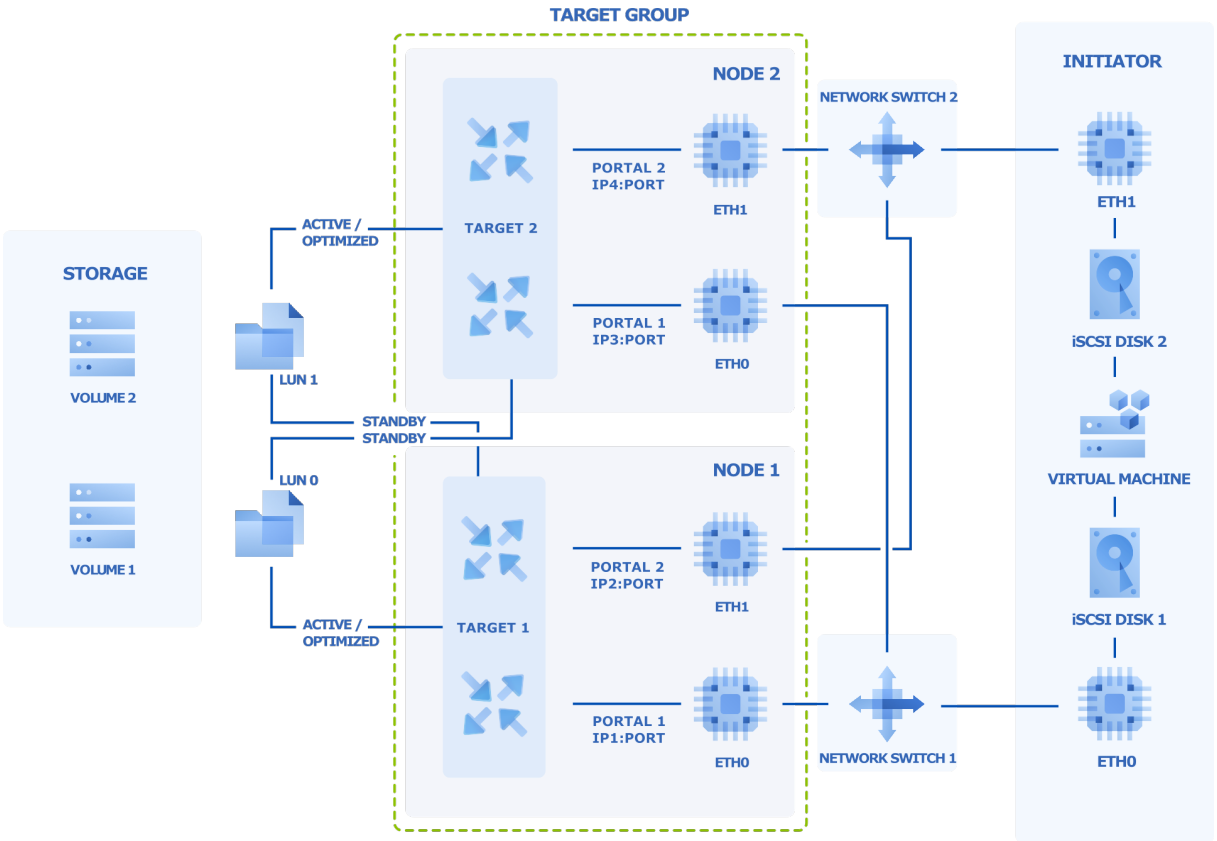
Virtuozzo Hybrid Infrastructure allows you to export cluster disk space to external operating systems and third-party virtualization solutions, in the form of LUN block devices over iSCSI in a SAN-like manner.

In Virtuozzo Hybrid Infrastructure, you can create groups of redundant targets running on different storage nodes. To each target group you can attach multiple storage volumes with their own redundancy provided by the storage layer. These volumes are exported by targets as LUNs.

Each node in a target group can host a single target for that group. If one of the nodes in a target group fails along with its targets, healthy targets from the same group continue to provide access to the LUNs previously serviced by the failed targets.

You can create multiple target groups on same nodes. A volume, however, may only be attached to one target group at any moment in time.

The figure below shows a typical setup for exporting Virtuozzo Hybrid Infrastructure disk space via iSCSI.



The figure shows two volumes located on redundant storage provided by Virtuozzo Hybrid Infrastructure. The volumes are attached as LUNs to a group of two targets running on Virtuozzo Hybrid Infrastructure nodes. Each target has two portals, one per network interface, with the iSCSI traffic type. This makes a total of four discoverable endpoints with different IP addresses. Each target provides access to all LUNs attached to the group.

Targets work in the ALUA mode, so one path to the volume is preferred and considered Active/Optimized while the other is Standby. The Active/Optimized path is normally chosen by the initiator (Explicit ALUA). If the initiator cannot do so (either does not support it or times out), the path is chosen by the storage itself (Implicit ALUA).

Network interfaces **eth0** and **eth1** on each node are connected to different switches for redundancy. The initiator, for example, VMware ESXi, is connected to both switches as well and provides volumes as iSCSI disks 1 and 2 to a VM via different network paths.

If the Active/Optimized path becomes unavailable for some reason (for example, the node with the target or network switch fails), the Standby path through the other target will be used instead to connect to the volume. When the Active/Optimized path is restored, it will be used again.

iSCSI workflow overview

The typical workflow of exporting volumes via iSCSI is as follows:

1. Assign the network with the traffic type **iSCSI** to a network interface on each node that you will add to a target group. Refer to "Managing node network interfaces" (p. 50).
2. Create a target group on chosen nodes, providing details for target WWNs and portals. Targets will be created automatically and added to the group. Target portals will be created on specified network interfaces and ports. Refer to "Creating target groups" (p. 304).
3. Create volumes and attach them to the target group. Refer to "Managing iSCSI volumes" (p. 307).
4. Optionally, enable CHAP authorization for the target group, create CHAP accounts, and assign them to the target group. Refer to "Managing CHAP accounts" (p. 314).
5. Optionally, enable ACL authorization for the target group, create a list of initiators that will be allowed to access only specific LUNs. Initiators not on the list will be able to access all LUNs in the target group. Refer to "Managing LUN views" (p. 315).
6. Start the target group. Refer to "Starting and stopping target groups" (p. 305).
7. Connect initiators to targets by using standard tools of your operating system or product, for example, `iscsiadm`. Use the `vstorage-target session-list` command to view iSCSI sessions active on a node in a target group.

Configuring CLI tool

Before you can use the `vstorage-target` CLI tool to export volumes via iSCSI, set it up, as described below. Perform these steps on each node where you plan to run iSCSI targets.

1. Create a configuration file `/etc/vstorage/iscsi/config.json` with at least these mandatory parameters:

```
{
  "ClusterName": "cluster1",
  "VolumesRoot": "/vols/iscsi/vols",
}
```


Where `ClusterName` is the name of your storage cluster and `VolumesRoot` is the path to the directory for iSCSI volumes.

You can also set these optional parameters:

- `"PcsLogLevel"`: the log level, ranges from 1 (log errors only) to 7 (log all, including debug messages).
- `"LogPath"`: the path to log files, the default is `/var/log/vstorage` (the log will be saved to `vstorage-target.log`).
- `"GetTimeout"`: the timeout for the initiator's command to read target port group status, the default is 3000 ms.

2. Enable the target monitor service:

```
# systemctl start vstorage-target-monitor.service
# systemctl enable vstorage-target-monitor.service
```

3. Create the iSCSI volume directory if it does not exist:

```
# mkdir -p /mnt/vstorage/vols/iscsi/
```

If you modify the configuration file later, restart the TCM monitor service to apply changes:

```
# systemctl restart vstorage-target-monitor.service
```

Managing target groups

This section explains how to create and manage groups of iSCSI targets.

Creating target groups

Before you create any target groups, assign the network with the iSCSI traffic type to a network interface on each node that you will add to a target group.

To create a target group, you will need a configuration file with a list of nodes to add to the group, as well as target WWNs and portals. For example:

```
[
  {
    "NodeId": "01baeabee73e4a0d",
    "WWN": "iqn.2013-10.com.vstorage:test1",
    "Portals": [
      {
        "Addr": "192.168.10.11",
        "Port": 3025
      }
    ]
  },
  {
    "NodeId": "0d90158e9d2444e1",
```



```
    "WWN": "iqn.2013-10.com.vstorage:test2",
    "Portals": [
      {
        "Addr": "192.168.10.12",
        "Port": 3025
      }
    ]
  },
  {
    "NodeId": "a9eca47661a64031",
    "WWN": "iqn.2013-10.com.vstorage:test3",
    "Portals": [
      {
        "Addr": "192.168.10.13",
        "Port": 3025
      }
    ]
  }
]
```

In this configuration file:

- `NodeId` is a node identifier that you can obtain from `/etc/vstorage/host_id` on a node.
- `WWN` is a target world wide name, an IQN, for example, **iqn.2013-10.com.vstorage:test1** (you can only customize the last part after the colon).
- `Portals` is one or more target portals, IP address and port combinations that the target will be accessible at. The IP address `Addr` belongs to a public network interface on the node that handles the "iSCSI" traffic type. The port `Port` is optional and defaults to 3260, if omitted.

Once you have the configuration file, for example, `tg1.json`, you can create the target group with the `vstorage-target tg-create` command. For example, to create an iSCSI target group, run:

```
# vstorage-target tg-create -name tg1 -targets tg1.json -type ISCSI
{
  "Id": "3d8364f5-b830-4211-85af-3a19d30ebac4"
}
```

When you run the command, targets are created on the nodes specified in the configuration file and joined to the target group, target portals are created on the specified network interfaces and ports.

Starting and stopping target groups

When you create a target group, its targets are initially stopped. You can start them with the `vstorage-target tg-start` command. For example:

```
# vstorage-target tg-start -id 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command starts all targets in the group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

All targets in a group can either be running or stopped. So if you add targets to a group of running targets, the new targets will be started automatically.

To stop a target group, use the `vstorage-target tg-stop` command. For example:

```
# vstorage-target tg-stop -id 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command stops all targets in the group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

Listing target groups

You can list target groups with the `vstorage-target tg-list` command, which displays basic information about groups. For example:

```
# vstorage-target tg-list
[
  {
    "Id": "3d8364f5-b830-4211-85af-3a19d30ebac4",
    "Name": "tg1",
    "Type": "ISCSI",
    "Running": true,
    "ACL": false,
    "ChapAuth": false,
    "CHAP": {},
    "Mode": 0
  },
  {
    "Id": "78c3b51e-fd9a-485b-91ce-bc0a8171c89d",
    "Name": "tg2",
    "Type": "ISCSI",
    "Running": false,
    "ACL": false,
    "ChapAuth": false,
    "CHAP": {},
    "Mode": 0
  }
]
```

To print complete information about all target groups, use `vstorage-target tg-list -all`.

Printing details of target groups

To print the details of a specific target group, use the `vstorage-target tg-status` command. For example:

```
# vstorage-target tg-status -id faeacacd-eba6-416c-9a7f-b5ba9e372e16
```

This command prints the complete details of the target group with the ID `faeacacd-eba6-416c-9a7f-b5ba9e372e16`. One parameter to pay attention to is `NodeState`. It indicates whether a node is in sync with the target group, that is, aware of its current configuration. The following states can be shown:

- `synced`: the node is in sync with the target group.
- `syncing`: the node is syncing with the target group.
- `failed`: the node failed to sync with the target group (refer to the `Error` parameter for details).
- `offline`: the node is offline.
- `disabled`: the node is disabled and its target is offline.

Managing persistent reservations of target groups

SCSI-2 reservations allow initiators to gain exclusive access to a LUN and prevent other initiators from making changes to that LUN at the same time. Such reservations are typically released by the initiator after changes have been made to the LUN. They are, however, also released on initiator failures or logical unit resets. SCSI-3 introduces persistent reservations that remain in case of failures or resets and are released by the initiator when needed. They also allow multiple initiators to communicate with the LUN in a controlled manner.

In Virtuozzo Hybrid Infrastructure, persistent reservations are used mostly to support Microsoft Hyper-V nodes working in Failover Clusters.

SCSI persistent reservations are enabled by default. You can enable and disable them, for all volumes in the target group. Do the following:

```
# vstorage-target tg-pr -id <tg_ID> -enable
# vstorage-target tg-pr -id <tg_ID> -disable
```

Where `<tg_ID>` is the ID of the target group for which persistent reservations are set.

Note
For persistent reservations to work, the `vstorage-target-manager` service must be running on all MDS nodes.

Deleting target groups

To delete a target group, use the `vstorage-target tg-delete` command. For example:

```
# vstorage-target tg-delete -id 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command deletes the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

Managing iSCSI volumes

This section describes how to create and manage volumes to be exported via iSCSI.

Creating iSCSI volumes

To create a volume, use the `vstorage-target vol-create` command. For example:

```
# vstorage-target vol-create -name vol1 -size 1G \
-vstorage-attr "replicas=3:2 failure-domain=host tier=0"
{
  "Id": "3277153b-5296-49c5-9b66-4c200ddb343d"
}
```

This command creates a 1 GB volume named `vol1` on storage tier 0 with 3:2 replication and host as failure domain.

Listing and printing details of iSCSI volumes

To list volumes, use the `vstorage-target vol-list` command. For example:

```
# vstorage-target vol-list
[
  "3277153b-5296-49c5-9b66-4c200ddb343d",
  "a12110d5-cbbc-498a-acdd-a8567286f927",
  "d5cc3c13-cfb4-4890-a20d-fb80e2a56278"
]
```

Use `vstorage-target vol-stat -all` to print details of all volumes. To print details of a specific volume, run `vstorage-target vol-stat -id <vol_ID>`.

Attaching iSCSI volumes to target groups

To attach a volume to a target group, use the `vstorage-target tg-attach` command. A volume cannot be attached to multiple target groups at the same time. For example:

```
# vstorage-target tg-attach -id 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-volume 3277153b-5296-49c5-9b66-4c200ddb343d -lun 0 -node bbfd0e7a26b1406d
```

This command attaches the volume with the ID `3277153b-5296-49c5-9b66-4c200ddb343d` to a target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4` as LUN 0. LUN ID numbering must start with 0. The same command sets the `PREFERRED` bit to the node with the ID `bbfd0e7a26b1406d`. The default Active/Optimized path will go via this node.

Setting the Active/Optimized path for iSCSI volumes

To set an Active/Optimized path for an iSCSI volume, use the `vstorage-target vol-set` command. It will only work if the specified node is `STABLE`.

Note
Make sure the new preferred node is reachable by the initiator.

```
# vstorage-target vol-set -id 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-pref-node bbfd0e7a26b1406d
```

This command sets the Active/Optimized path for the volume with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4` to the node with the ID `bbfd0e7a26b1406d`.

Viewing iSCSI volume ALUA information

To view the ALUA information for an iSCSI volume, use the `vstorage-target vol-info` command. For example:

```
# vstorage-target vol-info -id 3d8364f5-b830-4211-85af-3a19d30ebac4
Volume ID:      3d8364f5-b830-4211-85af-3a19d30ebac4
Name:           vol1
Size:           1073741824
Used:           1073152
Serial:         d2be0e84fd7f
Attrs:          map[]
TG:             4708b908-8c2d-444c-91b1-a1e18a96d4fc
LUN:            0

                              *** Node #0 ***
                        -------------------------
    NodeId:                     bbfd0e7a26b1406d
     State:                              synced
      TPGs:                       vstorage_tpg_0
      ALUA:                              active
 Preferred:                                   1
      WWNs:  iqn.2014-06.com.vstorage:target1 [2]
   Portals:                        10.37.130.61
```

This command shows the ALUA details for the volume with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

Viewing and setting iSCSI volume parameters

To view and set volume parameters, for example, redundancy mode, failure domain, or tier, use the commands `vstorage-target vol-attr get` and `vstorage-target vol-attr set`. For example:

```
# vstorage-target vol-attr get -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278
{
  "chunk-size": "268435456",
  "client-ssd-cache": "1",
  "failure-domain": "host",
  "replicas": "3:2",
```

```
  "tier": "0"
}
# vstorage-target vol-attr set -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278 \
-vstorage-attr "replicas=2:1 tier=1"
```

The first command shows the parameters of the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278`. The second command sets the redundancy mode to 2 replicas and the tier to 1 for this volume.

Increasing iSCSI volume size

To increase the size of a volume, use the `vstorage-target vol-grow` command. For example:

```
# vstorage-target vol-grow -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278 -size 2G
```

This command expands the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278` to 2 GB.

Setting iSCSI volume limits

To set read/write limits for a volume, use the `vstorage-target vol-limits` command. For example:

```
# vstorage-target vol-limits -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278 \
-read-bps 10485760 -write-bps 10485760
```

This command sets read/write speed for the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278` to 10485760 bytes per second.

Detaching iSCSI volumes from target groups

To detach a volume from a target group, use the `vstorage-target tg-detach` command. LUN 0 must be detached last. For example:

```
# vstorage-target tg-detach -id 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-volume d5cc3c13-cfb4-4890-a20d-fb80e2a56278
```

This command detaches the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278` from the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

Deleting iSCSI volumes

To delete a volume, use the `vstorage-target vol-delete` command. You cannot delete volumes attached to target groups. For example:

```
# vstorage-target vol-delete -id d5cc3c13-cfb4-4890-a20d-fb80e2a56278
```

This command deletes the volume with the ID `d5cc3c13-cfb4-4890-a20d-fb80e2a56278`.

Managing nodes in target groups

This section describes how to manage nodes in relation to target groups.

Adding nodes to target groups

To add a node to a target group, create a configuration file with details about target WWN and portal. The target will be created automatically on the added node. One node can be added to multiple target groups, and the same network interfaces on it can be used simultaneously by multiple targets from different groups.

For example:

```
# vstorage-target node-add -node bbfd0e7a26b1406d \
-tg 3d8364f5-b830-4211-85af-3a19d30ebac4 -targets target.json
```

This command adds the node with the ID `bbfd0e7a26b1406d` to the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`. It also creates a target on it according to the `target.json` configuration file that looks as follows:

```
[
  {
    "NodeId": "bbfd0e7a26b1406d",
    "WWN": "iqn.2013-10.com.vstorage:test2",
    "Portals": [
      {
        "Addr": "10.94.104.89",
        "Port": 3260
      }
    ]
  }
]
```

Listing nodes in target groups

You can list all nodes in all target groups and their detailed information by using the `vstorage-target node-list` command. For example:

```
# vstorage-target node-list
[
  {
    "ID": "bbfd0e7a26b1406d",
    "Status": "STABLE",
    "Enabled": true,
    "MonitorOnline": true,
    "Version": "7.10.32",
    "Address": "10.94.104.89:40135",
```

```
    "ActiveVolumes": [
      "0937f0e3-91a9-4dfc-8c10-6202bdc792c8"
    ]
  },
]
```

Enabling and disabling nodes in target groups

To enable or disable a node in all target groups it belongs to, use the `vstorage-target node-set` command. Enabling a node starts its targets, while disabling a node stops its targets and moves the active path to another node. These operations are also performed when the node exits and enters maintenance.

For example, to enable a node with the ID `bbfd0e7a26b1406d`, run:

```
# vstorage-target node-set -node bbfd0e7a26b1406d -enable
```

Before disabling a node, make sure there are other `STABLE` nodes where the Active/Optimized path can be moved. Otherwise, an I/O error will occur.

To disable a node with the ID `bbfd0e7a26b1406d`, run:

```
# vstorage-target node-set -node bbfd0e7a26b1406d -disable
```

You can check the node status with the `vstorage-target node-list` command, refer to "Listing nodes in target groups" (p. 311).

Deleting nodes from target groups

To delete a node from a target group, use the `vstorage-target node-del` command. You can only delete a node if it is not on the Active/Optimized path. Otherwise, you need to move the A/O path to another node either by disabling the node (refer to "Enabling and disabling nodes in target groups" (p. 312)) or manually (refer to "Setting the Active/Optimized path for iSCSI volumes" (p. 308)).

```
# vstorage-target node-del -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-node bbfd0e7a26b1406d
```

This command deletes the node with the ID `bbfd0e7a26b1406d` from the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

Managing targets and portals

This section describes how to create and manage targets. The optimal way is to create a single target per node.

Creating targets

Typically, targets are created automatically when you create target groups or add nodes to them. However, as you can delete target(s) from a node without removing the node from a target group, you can also create target(s) on such a node again. Use the `vstorage-target target-create` command. For example:

```
# vstorage-target target-create -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-json target.json
```

This command creates a target based on the `target.json` configuration file in the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`. The configuration file lists target details like the node to create the target on, WWN, and portal. For example:

```
{
  "NodeId": "bbfd0e7a26b1406d",
  "WWN": "iqn.2013-10.com.vstorage:test22",
  "Portals": [
    {
      "Addr": "10.94.104.90",
      "Port": 3260
    }
  ]
}
```

Adding and removing target portals

To add a portal to a target, use the `vstorage-target target-portal add` command. For example:

```
# vstorage-target target-portal add -wwn iqn.2013-10.com.vstorage:test2 \
-addr 10.94.104.90 -tg 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command adds a portal with the IP address 10.94.104.90 and default port 3260 to the target with the IQN `iqn.2013-10.com.vstorage:test2` in the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4`.

To delete a portal from a target, use the `vstorage-target target-portal del` command. For example:

```
# vstorage-target target-portal del -wwn iqn.2013-10.com.vstorage:test2 \
-addr 10.94.104.90 -tg 3d8364f5-b830-4211-85af-3a19d30ebac4
```

This command deletes the portal created before.

Deleting targets

To delete a target from a target group (as well as the node it is on), use the `vstorage-target target-delete` command. For example:

```
# vstorage-target target-delete -tg 3d8364f5-b830-4211-85af-3a19d30ebac4 \
-wwn iqn.2013-10.com.vstorage:test22
```

This command deletes the target with the IQN `iqn.2013-10.com.vstorage:test22` from the target group with the ID `3d8364f5-b830-4211-85af-3a19d30ebac4` and from the node it is located on.

A node that has no targets left on it is deleted from the target group.

Managing CHAP accounts

The Challenge-Handshake Authentication Protocol (CHAP) provides a way to restrict access to targets and their LUNs by requiring a user name and a password from the initiator. CHAP accounts apply to entire target groups.

To use CHAP, enable it for the target group:

```
# vstorage-target tg-auth -enable-chap -id <tg_ID>
```

Creating and listing CHAP accounts

To create a CHAP account, use the `vstorage-target account-create` command. For example:

```
# vstorage-target account-create -user user1 -desc "User for TG1"
Enter Password:
```

The password must be 12 to 16 characters long.

To list existing CHAP accounts and their details, use the `vstorage-target account-list` command.

Changing CHAP account details

To change the password or description of a CHAP account, use the `vstorage-target account-set` command. For example:

```
# vstorage-target account-set description -user user1 -desc "A new description"
# vstorage-target account-set password -user user1
Enter Password:
```

Assigning CHAP accounts to target groups

To assign a CHAP account to a target group, use the `vstorage-target tg-chap` command. For example:

```
# vstorage-target tg-chap set -id faeacacd-eba6-416c-9a7f-b5ba9e372e16 \
-user user1
```

To remove an assignment, run:

```
# vstorage-target tg-chap del -id faeacacd-eba6-416c-9a7f-b5ba9e372e16 \
-user user1
```

Deleting CHAP accounts

To delete an unused CHAP account, use the `vstorage-target account-delete` command. For example:

```
# vstorage-target account-delete -user user1
```

Managing LUN views

LUN views provide a way to create and manage an access control list (ACL) that limits access to chosen LUNs for specific initiators. Initiators not on the list have access to all LUNs in iSCSI target groups.

To use ACL-based authorization, enable it for the target group:

```
# vstorage-target tg-auth -enable-acl -id <tg_ID>
```

Creating LUN views

To create a LUN view for an initiator, use the commands `vstorage-target tg-initiator add` or `vstorage-target view-add`. The former command adds an initiator to the target group's ACL and creates a view for it. The latter command is used to add views to initiators that are already on the ACL.

For example:

```
# vstorage-target tg-initiator add -alias initiator2 -luns 0,1 \
-tg ee764519-80e3-406e-b637-8d63712badf1 -wwn iqn.1994-05.com.redhat:1535946874d
```

This command adds the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d` to the ACL of the target group with the ID `ee764519-80e3-406e-b637-8d63712badf1` and creates a view allowing it to access the LUNs with the IDs 0 and 1.

Another example:

```
# vstorage-target view-add -tg faeacacd-eba6-416c-9a7f-b5ba9e372e16 -lun 2 \
-map 2 -wwn iqn.1994-05.com.redhat:1535946874d
```

This command adds a view for the same initiator allowing it to access LUN 2 as well.

Listing LUN views

To list LUN views for an initiator, use the `vstorage-target view-list` command. For example:

```
# vstorage-target view-list -tg ee764519-80e3-406e-b637-8d63712badf1 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command lists views for the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d`.

Changing LUN view details

To change LUN views for an initiator, use the `vstorage-target view-set` command. For example:

```
# vstorage-target view-set -luns 1 -tg ee764519-80e3-406e-b637-8d63712badf1 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command allows the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d` to access only LUN 1. Essentially, it deletes all LUN views for it excluding specified.

Deleting LUN views

To delete a LUN view for an initiator, use the `vstorage-target view-del` command.

```
# vstorage-target view-del -lun 1 -tg ee764519-80e3-406e-b637-8d63712badf1 \
-wwn iqn.1994-05.com.redhat:1535946874d
```

This command deletes the view for LUN 1 for the initiator with the IQN `iqn.1994-05.com.redhat:1535946874d`.

Advanced tasks

This chapter describes miscellaneous configuration and management tasks.

Managing guest tools

This section explains how to install and uninstall the guest tools. This functionality is required for "Running commands in virtual machines without network connectivity" (p. 321).

Installing guest tools

To be able to install the guest tools in virtual machines, you first need to create and upload compute images from the supplied guest tools ISO files located in `/usr/share/vz-guest-tools/`. Execute the following commands on one of the compute nodes:

- For Linux guest tools:

```
# vinfra service compute image create vz-guest-tools-lin \
--file /usr/share/vz-guest-tools/vz-guest-tools-lin.iso --os-distro linux
Uploading image to server [Elapsed Time: 0:00:05] ...
```

- For Windows guest tools:

```
# vinfra service compute image create vz-guest-tools-win \
--file /usr/share/vz-guest-tools/vz-guest-tools-win.iso --os-distro windows
Uploading image to server [Elapsed Time: 0:00:09] ...
```

Next, you need to attach the created image to a VM and run the guest tools installer. The steps differ for new and already existing VMs and are described in the following subsections.

Installing guest tools in new virtual machines

When you create a new VM, you can attach the guest tools image to it and install the guest tools after the operating system. To do this, perform the following steps on a compute node:

1. Create a new VM with the guest tools image. For example, to create a Linux VM `centos`, run:

```
# vinfra service compute server create centos --network id=private \
--flavor medium --volume source=blank,size=64,boot-index=0,type=disk \
--volume source=image,id=centos7,size=3,boot-index=1,type=cdrom \
--volume source=image,id=vz-guest-tools-lin,size=1,boot-index=2,type=cdrom
```

Note
Round up the size of volumes to be created from images. For example, if the OS distribution image is 2.6 GB, use `size=3`.

In this example, the first volume is a blank virtual HDD, the second volume is the OS distribution image `centos7`, and the third volume is the guest tools image `vz-guest-tools-lin`. Make sure to specify the correct boot order by means of the `boot-index` parameter.

2. Log in to the virtual machine and install an operating system in it.
3. Run guest tools installer inside the VM:
   - Inside a Linux VM, create a mount point for the optical drive with the guest tools image and run the installer:

     ```
     # mkdir /mnt/cdrom
     # mount /dev/sr1 /mnt/cdrom
     # bash /mnt/cdrom/install
     ```

   - Inside a Windows VM, launch the installer in the AutoPlay window if autorun is enabled. Otherwise open the optical drive in Explorer and run `setup.exe`.

     After installing guest tools, restart the VM.

Note
Guest tools rely on the QEMU guest agent that is installed alongside the tools. The agent service must be running for the tools to work.

Installing guest tools in existing virtual machines

The steps you need to perform to install the guest tools in existing VMs depend on the guest OS type. They are described in the following subsections.

Installing guest tools in existing Linux virtual machines

To install the guest tools in an existing Linux virtual machine, do the following:

1. Create a volume from the uploaded guest tools image. For example:

   ```
   # vinfra service compute volume create vz-guest-tools-lin-vol \
   --storage-policy default --size 1 --image vz-guest-tools-lin
   ```

2. Attach the guest tools volume to the virtual machine. For example:

   ```
   # vinfra service compute server volume attach \
   --server centos vz-guest-tools-lin-vol
   +--------+--------------------------------------+
   | Field  | Value                                |
   +--------+--------------------------------------+
   | device | /dev/sr1                             |
   | id     | 1a40012a-7976-47a1-81f1-ff498cba90af |
   +--------+--------------------------------------+
   ```

3. Log in to the virtual machine, create a mount point for the optical drive with the guest tools image, and then run the installer:

```
# mkdir /mnt/cdrom
# mount /dev/sr1 /mnt/cdrom
# bash /mnt/cdrom/install
```

Note
Guest tools rely on the QEMU guest agent that is installed alongside the tools. The agent service must be running for the tools to work.

Installing guest tools in existing Windows virtual machines

To install the guest tools in an existing Windows virtual machine, do the following:

1. Power off the Windows VM. For example, to stop the `win10` VM, run:

   ```
   # vinfra service compute server stop win10
   ```

2. Convert its system volume to a template image. You will need the volume ID that you can obtain with `vinfra service compute volume list`. For example, to use the `win10` VM boot volume, run:

   ```
   # vinfra service compute volume list | grep win10
   | 7116d747-a1e1-4200-bd4a-25cc51ef006c | win10/<...>/Boot volume   | <...> |
   | ef2f1979-7811-4df6-9955-07e2fc942858 | win10/<...>/CD/DVD volume | <...> |
   # vinfra service compute volume upload-to-image \
   7116d747-a1e1-4200-bd4a-25cc51ef006c | grep id
   | id               | 79da5239-b2bb-4779-ada2-46cb8da8ba0e
   ```

3. Create a new Windows VM from the template, attaching the guest tools image to it during creation. For example:

   ```
   # vinfra service compute server create newvm --network id=private \
   --flavor medium --volume source=image,id=79da5239-b2bb-4779-ada2-46cb8da8ba0e,\
   size=64,boot-index=0,type=disk --volume source=image,id=vz-guest-tools-win,\
   size=1,boot-index=1,type=cdrom
   ```

   Note
   The size of volume to be created from a template image must be equal to or greater than the minimum volume size specified in the image metadata. You can learn the minimum volume size by using `vinfra service compute image show <image_id> | grep min_disk`.

   In this example, the first volume is the template of the original VM's system disk and the second volume is the guest tools image. Make sure to specify the correct boot order by means of the `boot-index` parameter.

4. Once the image is mounted inside the Windows VM, launch the installer in the AutoPlay window if autorun is enabled. Otherwise open the optical drive in Explorer and run `setup.exe`.
   After installing guest tools, restart the VM.

Note
Guest tools rely on the QEMU guest agent that is installed alongside the tools. The agent service must be running for the tools to work.

Uninstalling guest tools

The steps you need to perform to remove guest tools depend on the guest OS and are described in the following sections.

Uninstalling guest tools from Linux virtual machines

To uninstall the guest tools from a Linux guest, log in to the virtual machine, and then do the following:

1. Remove the packages:
   a. On RPM-based systems (CentOS and other):

      ```
      # yum remove dkms-vzvirtio_balloon prl_nettool qemu-guest-agent-vz \
      vz-guest-udev
      ```

   b. On DEB-based systems (Debian and Ubuntu):

      ```
      # apt-get remove vzvirtio-balloon-dkms prl-nettool qemu-guest-agent-vz \
      vz-guest-udev
      ```

      If any of the packages listed above are not installed on your system, the command will fail. In this case, exclude these packages from the command and run it again.
2. Remove the files:

   ```
   # rm -f /usr/bin/prl_backup /usr/share/qemu-ga/VERSION \
   /usr/bin/install-tools \
   /etc/udev/rules.d/90-guest_iso.rules /usr/local/bin/fstrim-static \
   /etc/cron.weekly/fstrim
   ```

3. Reload the udev rules:

   ```
   # udevadm control --reload
   ```

After removing guest tools, restart the virtual machine.

Uninstalling guest tools from Windows virtual machines

To uninstall the guest tools for Windows, log in to the virtual machine, and then do the following:

1. Remove the QEMU device drivers from the device manager.

   Important
   Do not remove the VirtIO/SCSI hard disk driver and NetKVM network driver. Without the former, the VM will not boot; without the latter, the VM will lose network connectivity.

2. Uninstall the QEMU guest agent and guest tools from the list of installed applications.
3. Stop and delete **Guest Tools Monitor**:

   ```
   > sc stop VzGuestToolsMonitor
   > sc delete VzGuestToolsMonitor
   ```

4. Unregister **Guest Tools Monitor** from **Event Log**:

   ```
   > reg delete HKLM\SYSTEM\CurrentControlSet\services\eventlog\Application\\
   VzGuestToolsMonitor
   ```

5. Delete the autorun registry key for **RebootNotifier**:

   ```
   > reg delete HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v \
   VzRebootNotifier
   ```

6. Delete the `C:\Program Files\Qemu-ga\` directory.

   If `VzGuestToolsMonitor.exe` is locked, close all the Event Viewer windows. If it remains locked, restart the `eventlog` service:

   ```
   > sc stop eventlog
   > sc start eventlog
   ```

After removing the guest tools, restart the virtual machine.

Running commands in virtual machines without network connectivity

If a VM cannot access a network for some reason, you can still run commands in it from the node the VM resides on. The VM in question must have the guest tools installed in it (refer to "Managing guest tools" (p. 317)).

You will need the VM ID that you can obtain with `vinfra service compute server list`. You can also use a virsh domain name that you can get using `virsh list`.

Running commands in Linux virtual machines

To run an arbitrary command inside a Linux VM and receive the output to your console, use the `virsh x-exec` command. For example:


```
# virsh x-exec 1d45a54b-0e20-4d5e-8f11-12c8b4f300db /usr/bin/bash -c 'lsblk'
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0         7:0    0 945.9M  1 loop
loop1         7:1    0     5G  1 loop
├─live-rw   253:0    0     5G  0 dm   /
└─live-base 253:1    0     5G  1 dm
loop2         7:2    0    32G  0 loop
└─live-rw   253:0    0     5G  0 dm   /
sda           8:0    0    64G  0 disk
sdc           8:32   0     1G  1 disk
sr0          11:0    1     2G  0 rom  /run/initramfs/live
```

To copy a file to a Linux VM, use the `virsh x-exec` and `cat` commands. For example:

```
# virsh x-exec 1d45a54b-0e20-4d5e-8f11-12c8b4f300db \
--shell 'cat > test.file' < /home/test.file
```

To get a file from a Linux VM, use the `virsh x-exec` and `cat` commands as well. For example:

```
# virsh x-exec 1d45a54b-0e20-4d5e-8f11-12c8b4f300db \
--shell 'cat /home/test.file' > test.file
```

Running commands in Windows virtual machines

To run an arbitrary command inside a Windows VM and receive the output to your console, use the `virsh x-exec` command. For example:

```
# virsh x-exec bbf4a6ec-865f-4e2c-ac21-8639d1bfb85c --shell dir c:\\
 Volume in drive C has no label.
 Volume Serial Number is D0BE-A8D1

 Directory of c:\

06/10/2009  01:42 PM                24 autoexec.bat
06/10/2009  01:42 PM                10 config.sys
07/13/2009  06:37 PM    <DIR>          PerfLogs
11/12/2018  07:45 AM    <DIR>          Program Files
11/12/2018  07:55 AM    <DIR>          test
11/12/2018  06:23 AM    <DIR>          Users
11/12/2018  07:53 AM    <DIR>          Windows
               2 File(s)             34 bytes
               5 Dir(s)  59,329,495,040 bytes free
```

To copy a file to a Windows VM, use the `virsh x-exec` and `prl_cat` commands. For example:

```
# virsh x-exec bbf4a6ec-865f-4e2c-ac21-8639d1bfb85c \
--shell '%programfiles%\\qemu-ga\\prl_cat' 'c:\test\test.file' < /home/test.file
```

To get a file from a Windows VM, use the `virsh x-exec` and `type` commands. For example:


```
# virsh x-exec bbf4a6ec-865f-4e2c-ac21-8639d1bfb85c \
--shell type 'c:\test\test.file' > test.file
```

Setting virtual machine CPU model

Virtual machines are created with the host CPU model by default. If nodes in the compute cluster have different CPUs, live migration of VMs between compute nodes may not work or applications inside VMs that depend on particular CPUs may not function properly. To avoid this, you can find out which CPU model offers compatibility across all nodes in the compute cluster and manually set it as the compute cluster default. In this case, however, the compute cluster CPU model will be the least advanced one and compute nodes will lose CPU capabilities of a more advanced processor.

To set the compute cluster CPU model, do the following:

1. Run `virsh capabilities` on each compute node, to print an XML document with information on node's CPU.
2. On one of the nodes, create an XML file, for example, `cpu-compare.xml`. Add the `<cpu>` sections from all of the nodes to this file. For example:

```
<cpu>
  <arch>x86_64</arch>
  <model>SandyBridge-IBRS</model>
  <vendor>Intel</vendor>
  <counter name='tsc' frequency='2099999000' scaling='yes'/>
  <topology sockets='1' cores='16' threads='2'/>
  <feature policy='require' name='vme'/>
  <feature policy='require' name='ds'/>
  <feature policy='require' name='acpi'/>
  <feature policy='require' name='ss'/>
  <feature policy='require' name='ht'/>
  <feature policy='require' name='tm'/>
  <feature policy='require' name='pbe'/>
  <feature policy='require' name='dtes64'/>
  <feature policy='require' name='monitor'/>
  <feature policy='require' name='ds_cpl'/>
  <feature policy='require' name='vmx'/>
  <feature policy='require' name='smx'/>
  <feature policy='require' name='est'/>
  <feature policy='require' name='tm2'/>
  <feature policy='require' name='xtpr'/>
  <feature policy='require' name='pdcm'/>
  <feature policy='require' name='pcid'/>
  <feature policy='require' name='dca'/>
  <feature policy='require' name='arat'/>
  <feature policy='require' name='md-clear'/>
  <feature policy='require' name='stibp'/>
  <feature policy='require' name='ssbd'/>
  <feature policy='require' name='xsaveopt'/>
  <feature policy='require' name='pdpe1gb'/>
```

```
  <feature policy='require' name='invtsc'/>
</cpu>
<cpu>
  <arch>x86_64</arch>
  <model>Skylake-Server</model>
  <vendor>Intel</vendor>
  <counter name='tsc' frequency='2099999000' scaling='yes'/>
  <topology sockets='1' cores='16' threads='2'/>
  <feature name='ds'/>
  <feature name='acpi'/>
  <feature name='ss'/>
  <feature name='ht'/>
  <feature name='tm'/>
  <feature name='pbe'/>
  <feature name='dtes64'/>
  <feature name='monitor'/>
  <feature name='ds_cpl'/>
  <feature name='vmx'/>
  <feature name='smx'/>
  <feature name='est'/>
  <feature name='tm2'/>
  <feature name='xtpr'/>
  <feature name='pdcm'/>
  <feature name='dca'/>
  <feature name='tsc_adjust'/>
  <feature name='cmt'/>
  <feature name='intel-pt'/>
  <feature name='pku'/>
  <feature name='md-clear'/>
  <feature name='stibp'/>
  <feature name='arch-capabilities'/>
  <feature name='xsaves'/>
  <feature name='invtsc'/>
  <feature name='rdctl-no'/>
  <feature name='ibrs-all'/>
  <feature name='skip-l1dfl-vmentry'/>
  <feature name='mds-no'/>
  <pages unit='KiB' size='4'/>
  <pages unit='KiB' size='2048'/>
  <pages unit='KiB' size='1048576'/>
</cpu>
```

3. Compare the CPU features by using `virsh cpu-baseline`. For example:

```
# virsh cpu-baseline cpu-compare.xml | grep model
<model fallback='allow'>SandyBridge</model>
```

   The command will print the most compatible CPU model across all nodes. In the example, it is SandyBridge.

4. Set this CPU model for the compute cluster. For example:

```
# vinfra service compute set --cpu-model SandyBridge
```

Take note of the following:

- For the list of supported CPU models, run "vinfra service compute show" (p. 85).
- Changing CPU model affects only new VMs (that is, those created after the change).

Creating Linux templates

If you do not have a ready Linux template, you can build one with the `diskimage-builder` tool. The disk image is created with only the root user that has neither password nor SSH keys. You can use the `user data` and `cloud-init` methods to perform initial configuration tasks on VMs that will be deployed from the disk image, for example, create custom user accounts. For more options to customize a VM during boot, refer to the cloud-init documentation.

To create a template and deploy a VM from it, do the following:

1. Install the `diskimage-builder` package:

   ```
   # yum install diskimage-builder
   ```

2. For the RHEL 7 guest OS, download the cloud image from the Red Hat Customer Portal (login required) and execute:

   ```
   # export DIB_LOCAL_IMAGE=<path_to_rhel7_image>
   ```

3. Execute the `disk-image-create` command to build a disk image with installed `cloud-init` for the desired Linux guest. For example:

   ```
   # disk-image-create vm centos7 -t qcow2 -o centos7
   ```

   where

   - `centos7` is the name of a guest OS. Can be one of the following: `centos6`, `centos7`, `debian`, `rhel7`, or `ubuntu`.

     By default, using the `ubuntu` element will create a disk image for Ubuntu 16.04. To build the Ubuntu 18.04 disk image, add the `DIB_RELEASE=bionic` to the command: `DIB_RELEASE=bionic disk-image-create vm ubuntu -t qcow2 -o ubuntu18`.
   - `-o` sets the name for the resulting disk image file.

4. Upload the created disk image by using the `vinfra` tool to the compute cluster:

   ```
   # vinfra service compute image create centos7-image --os-distro centos7 \
   --disk-format qcow2 --file centos7.qcow2
   ```

   where

- `centos7-image` is the name of a new image.
- `centos7` is the OS distribution. Can be one of the following: `centos6`, `centos7`, `debian9`, `rhel7`, `ubuntu16.04`, and `ubuntu18.04`.
- `centos7.qcow2` is the QCOW2-image created on step 3.

5. Create the `user-data` configuration file with a custom user account:

```
# cat <<EOF > user-data
#cloud-config
user: myuser
password: password
chpasswd: {expire: False}
ssh_pwauth: True
EOF
```

   where `myuser` is the name of a custom user and `password` is a password for the account.

6. Launch the deployment of a VM from the disk image by using the configuration file as user data:

```
# vinfra service compute server create centos7-vm --flavor medium \
--network public --user-data user-data --volume source=image,\
id=centos7-image,size=10
```

   where

   - `centos7-vm` is the name of a new VM.
   - `user-data` is the configuration file created in step 5.
   - `centos7-image` is the image added to the compute cluster in step 4.

   For more information on managing compute objects via the `vinfra` tool, refer to "Managing the compute cluster" (p. 82).

Connecting to OpenStack command-line interface

For managing the compute cluster, you can also use the OpenStack command-line client, which is automatically installed along with the Virtuozzo Hybrid Infrastructure.

To connect to and be able to use the OpenStack CLI, do the following:

1. Locate the node with the management role in the admin panel. Open the **Infrastructure** > **Nodes** screen. The management node runs the **Admin panel** service.
2. Access the management node via SSH and log in as the service user. For example:

```
# ssh node001.vstoragedomain
# su - vstoradmin
```

3. Generate the admin OpenRC script that sets environment variables:

```
# kolla-ansible post-deploy
```

   The command will create the `/etc/kolla/admin-openrc.sh` bash script:


```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=vstorage-service-user
export OS_PASSWORD=<password>
export OS_AUTH_URL=https://<MN_IP_address>:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_TYPE=password
export OS_INSECURE=true
export PYTHONWARNINGS="ignore:Unverified HTTPS request is being made"
export NOVACLIENT_INSECURE=true
export NEUTRONCLIENT_INSECURE=true
export CINDERCLIENT_INSECURE=true
export OS_PLACEMENT_API_VERSION=1.22
```

By default, the script is created to authorize OpenStack commands in the `admin` project under the `vstorage-service-user` user for managing the compute cluster with administrative privileges.

4. To perform administrative actions, run this script:

Important
You need to run the script each session.

```
# source /etc/kolla/admin-openrc.sh
```

If you want to work in another project under another user, you need to make changes to the `admin-openrc.sh` script. For example, to authorize OpenStack commands in the `myproject` project under the `myuser` user within the `mydomain` domain, do the following:

1. Copy the script to the chosen directory with a new name. For example:

```
# cp /etc/kolla/admin-openrc.sh /root/myscript.sh
```

2. Open the copied script for editing and change the first five variables as follows:

```
export OS_PROJECT_DOMAIN_NAME=mydomain
export OS_USER_DOMAIN_NAME=mydomain
export OS_PROJECT_NAME=myproject
export OS_USERNAME=myuser
export OS_PASSWORD=<myuser_password>
```

Leave other variables as is and save your changes.

3. Run the modified script:

Important
You need to run the script each session.

```
# source /root/myscript.sh
```

Now you can work in the project you have authorized in by executing OpenStack commands with the `--insecure` option. For example:

```
# openstack --insecure server list
+---------------------+------+--------+-----------------------+-------+--------+
| ID                  | Name | Status | Networks              | Image | Flavor |
+---------------------+------+--------+-----------------------+-------+--------+
| 32b0f95d-477f-<...> | vm1  | ACTIVE | private=192.168.128.87 |       | tiny   |
+---------------------+------+--------+-----------------------+-------+--------+
```

To learn how to secure OpenStack API traffic, refer to "Securing OpenStack API traffic with SSL" (p. 329).

Setting a DNS name for the compute API

By means of the **Compute API** traffic type, Virtuozzo Hybrid Infrastructure exposes a public endpoint that listens to OpenStack API requests. By default, it points to the IP address of the management node (or to its virtual IP address if high availability is enabled).

In some cases, you need to modify all public endpoints to use the domain name resolvable to the management node IP address (or its virtual IP), for example, to secure OpenStack API traffic with an SSL certificate without the `subjectAltName` field or to make the Kubernetes service access the compute API via the DNS name.

You can modify all public endpoints to use the domain name when creating the compute cluster or afterwards using the `--endpoint-hostname` option (refer to "vinfra service compute create" (p. 82) or "Changing compute cluster parameters" (p. 89)). For example, to use `dns-name.example` for public endpoints, execute:

```
# vinfra service compute set --endpoint-hostname dns-name.example
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 534391a2-946a-4406-8dc0-756f161cd595 |
+---------+--------------------------------------+
```

Wait until the task is complete:

```
# vinfra task show 534391a2-946a-4406-8dc0-756f161cd595
+---------+-------------------------------------------------------------------+
| Field   | Value                                                             |
+---------+-------------------------------------------------------------------+
| details |                                                                   |
| name    | backend.presentation.compute.tasks.ReconfigureComputeClusterTask |
| result  |                                                                   |
| state   | success                                                           |
| task_id | 534391a2-946a-4406-8dc0-756f161cd595                              |
+---------+-------------------------------------------------------------------+
```


To check that the given domain name is used instead of the management node IP address, do the following:

1. Generate or regenerate the admin OpenRC script:

```
# kolla-ansible post-deploy
```

2. Run the script:

```
# source /etc/kolla/admin-openrc.sh
```

3. List the public endpoints:

```
# openstack --insecure endpoint list | grep public
| 5a845b4b<...> | <...> | https://dns-name.example:8780                    |
| 7d901686<...> | <...> | https://dns-name.example:8776/v2/%(tenant_id)s   |
| 44aa0f53<...> | <...> | https://dns-name.example:8774/v2.1/%(tenant_id)s |
| 0e6d3a39<...> | <...> | https://dns-name.example:9292                    |
| 0b906e51<...> | <...> | https://dns-name.example:9696                    |
| 1b68ac7c<...> | <...> | https://dns-name.example:8776/v3/%(tenant_id)s   |
| d80af756<...> | <...> | https://dns-name.example:8004/v1/%(tenant_id)s   |
| d0e8c7da<...> | <...> | https://dns-name.example:5000/v3                 |
```

Securing OpenStack API traffic with SSL

Note
Only one SSL certificate can be added and applied for both the admin panel and OpenStack API.

Traffic to and from the public endpoint that listens to OpenStack API requests can be secured with an SSL certificate. However, as domain names are not used by default, the certificate will need a `subjectAltName` field containing the aforementioned management node IP address. If it does not have such a field, you will need to modify the public endpoint to use a domain name that you have a certificate for.

To secure public OpenStack API traffic with SSL, do the following:

1. In the admin panel, upload the SSL certificate and private key, on the **Settings** > **Management node** > **SSL access** screen.
2. On the client side, place the CA certificate file to the operating system's trusted bundle:

```
# cp ca.pem /etc/pki/ca-trust/source/anchors/
# update-ca-trust extract
```

   Alternatively, you can append the `--os-cacert ca.pem` option to each OpenStack client call.
3. If your certificate does not have the `subjectAltName` field, modify all public endpoints to use the domain name for which you have the certificate for, as described in "Setting a DNS name for the compute API" (p. 328). This domain name must resolve to the management node IP address (or to its virtual IP address if high availability is enabled).

4. In your OpenRC script, change `OS_AUTH_URL` to the same domain name and remove all parameters related to insecure access. For example:

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=<ADMIN_PASSWORD>
export OS_AUTH_URL=https://<DOMAIN_NAME>:5000/v3
export OS_IDENTITY_API_VERSION=3
```

Now you can run OpenStack commands without the `--insecure` option.

Using metering for compute resources

You can collect usage data of compute resources using Gnocchi. This time series database processes and stores measurement data of compute resources and provides access to it via REST API or the command-line tool.

Measurements can be sampled from such compute resources as virtual machines, VM disks and interfaces, compute networks, volumes, etc. All resources are being revised: if any attribute of a resource changes, this change is recorded in the history of the resource. For a VM, for example, you can measure the amount of allocated memory and virtual CPUs, as well as the memory and CPU usage.

An entity storing aggregates composed of a timestamp and value is called a metric. A metric is attached to a specific resource and associated to an archive policy. A policy defines how long aggregates are kept in a metric and how they are computed (minimum, maximum, average, etc.).

Two default archive policies, `low` and `ceilometer-low-rate`, are used for storing metrics. These policies imply that all computed aggregates are kept for one month with 5-minute granularity. The differences between them are the following:

- `ceilometer-low-rate` is used for cumulative metrics and stores only mean values and the average of delta values per interval
- `low` is used for gauge metrics and stores minimum, maximum, and mean values, standard deviation, as well as the sum and number of all measurements

The following metrics are available for aggregation:

Compute resource metrics

Metric	Type	Resource type	Description
`memory`	gauge	instance	Amount of RAM allocated to the VM, in megabytes
`memory.usage`	gauge	instance	Percentage of RAM used by the VM
`vcpus`	gauge	instance	Number of virtual CPUs allocated to the

Metric	Type	Resource type	Description
			VM
cpu	cumulative	instance	Amount of CPU time used by the VM, in nanoseconds
disk.device.read.requests	cumulative	instance_disk	Number of read requests
disk.device.write.requests	cumulative	instance_disk	Number of write requests
disk.device.read.bytes	cumulative	instance_disk	Amount of data read, in bytes
disk.device.write.bytes	cumulative	instance_disk	Amount of data written, in bytes
network.incoming.bytes	cumulative	instance_ network_ interface	Incoming network traffic, in bytes
network.outgoing.bytes	cumulative	instance_ network_ interface	Outgoing network traffic, in bytes
network.incoming.packets	cumulative	instance_ network_ interface	Incoming network traffic, in packets
network.outgoing.packets	cumulative	instance_ network_ interface	Outgoing network traffic, in packets
image.size	gauge	image	Size of the uploaded image, in bytes
volume.size	gauge	volume	Size of the volume, in gigabytes
snapshot.size	gauge	volume	Size of the volume snapshot, in gigabytes
magnum.cluster	gauge	coe_cluster	Number of Container Orchestration Engine (COE), that is, Kubernetes clusters
bandwidth	delta	network	Incoming and outgoing network traffic for a physical compute network, in bytes

Cumulative metrics are polled every 5 minutes and increase over time, delta metrics are polled every 5 minutes and change over time, and gauge metrics are updated on events and show fluctuating values.

Enabling the metering service

Note
If you have already installed metering services via the admin panel, skip this section.

You can enable metering services in your compute cluster by doing one of the following:

- If you have no compute cluster yet, deploy it and enable metering by adding the `--enable-metering` option to the `vinfra service compute create` command. For example:

```
# vinfra service compute create --nodes <node1_id>[,<node2_id>,…] \
--enable-metering
```

- If you have already created the compute cluster, use the following command:

```
# vinfra service compute set --enable-metering
```

Note
The metering service will only take into account compute objects created after it has been enabled.

These commands open port 8041 and enable two Gnocchi services: `gnocchi-api`, an HTTP server, and `gnocchi-metricd`, a metric daemon.

Using Gnocchi command-line tool

After enabling metering, you can get access to your compute resource metrics either via REST API or by using the Gnocchi command-line tool. To be able to use the tool, connect to the OpenStack command-line interface to authorize further `gnocchi` commands (refer to "Connecting to OpenStack command-line interface" (p. 326)):

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

Now you can use the `gnocchi` command with the `--insecure` option. You can see the full list of `gnocchi` commands in the OpenStack documentation.

Viewing resources

To view the existing resources, use the `gnocchi resource list` command. For example:

```
# gnocchi --insecure resource list -c id -c type -c project_id
+--------------+---------------------------+------------+
| id           | type                      | project_id |
+--------------+---------------------------+------------+
| 238597c7<...> | volume                    | c1bf1<...> |
| 3c78558f<...> | instance                  | c1bf1<...> |
| 44f1896f<...> | instance_network_interface | c1bf1<...> |
| 880e9efc<...> | instance_disk             | c1bf1<...> |
+--------------+---------------------------+------------+
```

The output shows that the compute cluster hosts one virtual machine with one NIC and one disk that is also present as a volume.

Viewing metrics for resources

To check all the available metrics for the resources, use the `gnocchi metric list` command. For example:

```
# gnocchi --insecure metric list
+-------------+-------+-------------------------------+---------+-------------+
| id          | <...> | name                          | unit    | resource_id |
+-------------+-------+-------------------------------+---------+-------------+
| 243c7a<...> | <...> | disk.root.size                | GB      | 3c7855<...> |
| 365e45<...> | <...> | network.outgoing.packets      | packet  | 44f189<...> |
| 4fbd3e<...> | <...> | disk.device.read.requests     | request | 880e9e<...> |
| 54519f<...> | <...> | compute.instance.booting.time | sec     | 3c7855<...> |
| 5e1406<...> | <...> | disk.device.write.bytes       | B       | 880e9e<...> |
| 66a96c<...> | <...> | vcpus                         | vcpu    | 3c7855<...> |
| 722ea9<...> | <...> | memory                        | MB      | 3c7855<...> |
| 7c961a<...> | <...> | disk.device.write.requests    | request | 880e9e<...> |
| 87e9fb<...> | <...> | network.incoming.packets      | packet  | 44f189<...> |
| 9d5632<...> | <...> | disk.device.read.bytes        | B       | 880e9e<...> |
| b8be8f<...> | <...> | cpu                           | ns      | 3c7855<...> |
| c1961b<...> | <...> | disk.ephemeral.size           | GB      | 3c7855<...> |
| c9b61e<...> | <...> | volume.size                   | GB      | 238597<...> |
| d06a58<...> | <...> | network.outgoing.bytes        | B       | 44f189<...> |
| e2d998<...> | <...> | network.incoming.bytes        | B       | 44f189<...> |
| eaac2b<...> | <...> | memory.usage                  | MB      | 3c7855<...> |
+-------------+-------+-------------------------------+---------+-------------+
```

Viewing measures for metrics

To view measures for a particular metric, use the `gnocchi measures show` command. For example:

```
# gnocchi --insecure measures show cpu \
--resource-id 3c78558f-08bf-47e2-ba3e-bdb13e7b25bb
+---------------------------+-------------+-------------+
| timestamp                 | granularity |       value |
+---------------------------+-------------+-------------+
| 2019-12-11T17:05:00+03:00 |       300.0 |  2.2756e+11 |
| 2019-12-11T17:10:00+03:00 |       300.0 |  2.8897e+11 |
| 2019-12-11T17:15:00+03:00 |       300.0 |  3.7367e+11 |
| 2019-12-11T17:20:00+03:00 |       300.0 |    4.64e+11 |
| 2019-12-11T17:25:00+03:00 |       300.0 |  7.6104e+11 |
+---------------------------+-------------+-------------+
```

By default, the `mean` aggregation method is used. To obtain how much CPU time is consumed per interval, use the `--aggregation rate:mean` option:

```
# gnocchi --insecure measures show cpu --aggregation rate:mean \
--resource-id 3c78558f-08bf-47e2-ba3e-bdb13e7b25bb
+---------------------------+-------------+---------------+
```

```
| timestamp                 | granularity |         value |
+---------------------------+-------------+---------------+
| 2019-12-11T17:10:00+03:00 |       300.0 | 61410000000.0 |
| 2019-12-11T17:15:00+03:00 |       300.0 | 84700000000.0 |
| 2019-12-11T17:20:00+03:00 |       300.0 | 90330000000.0 |
| 2019-12-11T17:25:00+03:00 |       300.0 |    2.9704e+11 |
| 2019-12-11T17:30:00+03:00 |       300.0 |      3.64e+11 |
+---------------------------+-------------+---------------+
```

Viewing outgoing traffic usage

The `bandwidth` metric that can show outgoing traffic usage is not available by default. To be able to use it, you need to configure it first. Do the following:

1. Create a meter by using the `openstack network meter create` command. For example, to create the meter `outgoing_traffic` in the project `project1` within the domain `domain1`, run:

   ```
   # openstack --insecure network meter create outgoing_traffic --share \
   --project project1 --project-domain domain1
   ```

2. Create a rule that includes all outgoing network traffic by specifying the CIDR 0.0.0.0/0 and the egress traffic direction. For example:

   ```
   # openstack --insecure network meter rule create outgoing_traffic --egress \
   --include --remote-ip-prefix 0.0.0.0/0
   ```

3. Create a rule that excludes the outgoing network traffic of your physical compute network by specifying its CIDR. For example, for the physical compute network with the CIDR 10.10.10.0/24, run:

   ```
   # openstack --insecure network meter rule create outgoing_traffic --egress \
   --exclude --remote-ip-prefix 10.10.10.0/24
   ```

Now, you can proceed to creating virtual routers and assigning floating IP addresses to virtual machines in the self-service panel. The `bandwidth` metric will only account outgoing VM traffic via virtual routers. You can view measures for this metric as follows:

1. Find out the `resource_id` for the `bandwidth` metric. For example:

   ```
   # gnocchi --insecure metric list | grep bandwidth
   | dce6d500-37c9<…> | low | bandwidth | B | cfc0cbb3-ff05-4b25-be7c-ab7f872e4180 |
   ```

2. View measures for the resource. For example:

   ```
   # gnocchi --insecure measures show --resource-id cfc0cbb3-ff05-4b25-be7c-ab7f872e4180
   bandwidth
   +---------------------------+-------------+---------+
   | timestamp                 | granularity |   value |
   +---------------------------+-------------+---------+
   ```

```
| 2021-10-03T02:40:00+03:00 |       300.0 |  1045.0 |
| 2021-10-03T02:50:00+03:00 |       300.0 |  1907.0 |
| 2021-10-03T02:55:00+03:00 |       300.0 | 15932.0 |
+---------------------------+-------------+---------+
```

Changing retention period for metrics

Note
Extending retention period increases the Gnocchi database size insignificantly compared to its default value.

You can change the retention period for metrics through an archive policy. By default, the `ceilometer-low-rate` and `low` policies are used to store metrics. Note that you cannot change the metering granularity for these policies.

To show the details of a particular archive policy, use the `gnocchi archive-policy show` command. For example:

```
# gnocchi --insecure archive-policy show ceilometer-low-rate
+---------------------+---------------------------------------+
| Field               | Value                                 |
+---------------------+---------------------------------------+
| aggregation_methods | rate:mean, mean                       |
| back_window         | 0                                     |
| definition          | - points: 8640, granularity: 0:05:00, |
|                     |   timespan: 30 days, 0:00:00          |
| name                | ceilometer-low-rate                   |
+---------------------+---------------------------------------+
```

In the output:

- 8640 points shows how many aggregates will be retained
- the granularity of 5 minutes defines the time between two aggregates
- the timespan of 30 days specifies the retention period for aggregates

To sum it up, metrics with the `ceilometer-low-rate` policy will keep 8640 computed aggregates for one month with 5-minute granularity.

To change the policy definition, use the `gnocchi archive-policy update` command. To calculate the correct number of points required for the desired timespan, refer to this formula: points = timespan \ granularity. For example, to keep aggregates for 2 months with the 5-minute granularity, specify 17280 points:

```
# gnocchi --insecure archive-policy update ceilometer-low-rate \
-d points:17280,granularity:0:05:00,timespan:60d
+---------------------+---------------------------------------+
| Field               | Value                                  |
+---------------------+---------------------------------------+
```



```
| aggregation_methods | rate:mean, mean                          |
| back_window         | 0                                        |
| definition          | - points: 17280, granularity: 0:05:00, |
|                     |   timespan: 60 days, 0:00:00           |
| name                | ceilometer-low-rate                      |
+---------------------+------------------------------------------+
```

Viewing resource usage per project

To get usage of compute resources allocated to all virtual machines that belong to a particular project, use the command `vinfra service compute quotas show --usage <project_id>`. For example:

```
# vinfra service compute quotas show 6ef6f48f01b640ccb8ff53117b830fa3 --usage
+---------------------------------+-------+
| Field                           | Value |
+---------------------------------+-------+
| compute.cores.limit             | 20    |
| compute.cores.used              | 2     |
| compute.ram.limit               | 40960 |
| compute.ram.used                | 4096  |
| k8saas.cluster.limit            | 10    |
| k8saas.cluster.used             | 0     |
| lbaas.loadbalancer.limit        | 10    |
| lbaas.loadbalancer.used         | 0     |
| network.floatingip.limit        | 10    |
| network.floatingip.used         | 0     |
| storage.gigabytes.default.limit | 1024  |
| storage.gigabytes.default.used  | 66    |
+---------------------------------+-------+
```

The output shows that VMs included in the project with the ID `62af79f31ae5488aa33077d02af48282` were allocated 2 vCPUs, 4 GB of RAM, and 66 GB of disk space.

Alternatively, to collect resource usage per project for a specific period of time, you can use the `gnocchi` tool. For example, to aggregate usage data for the project with the ID `75521ab61d1f4e9090aac5836c219492` from 12:00 PM July 18, 2021, to 12:00 PM July 19, 2021, run the following commands:

- To aggregate the number of provisioned vCPUs:

  ```
  # gnocchi --insecure aggregates --resource-type instance --needed-overlap 0 \
  "(aggregate sum (metric vcpus mean))" \
  "project_id=75521ab61d1f4e9090aac5836c219492" \
  --start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
  ```

- To aggregate the amount of provisioned RAM:

  ```
  # gnocchi --insecure aggregates --resource-type instance --needed-overlap 0 \
  "(aggregate sum (metric memory mean))" \
  ```

```
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

- To aggregate the total size of provisioned storage space:

```
# gnocchi --insecure aggregates --resource-type volume --needed-overlap 0 \
"(aggregate sum (metric volume.size mean))" \
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

- To aggregate the size of provisioned storage space with the storage policy with the ID `10056d2e-6fc9-4f2e-92c2-dbebb1714778`:

```
# gnocchi --insecure aggregates --resource-type volume --needed-overlap 0 \
"(aggregate sum (metric volume.size.10056d2e-6fc9-4f2e-92c2-dbebb1714778 mean))" \
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

- To aggregate the number of used floating IP addresses:

```
# gnocchi --insecure aggregates --resource-type network --needed-overlap 0 \
"(aggregate sum (metric ip.floating mean))" \
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

- To aggregate the size of outgoing network traffic:

```
# gnocchi --insecure aggregates --resource-type network --needed-overlap 0 \
"(aggregate sum (metric bandwidth mean))" \
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

- To aggregate the number of load balancers:

```
# gnocchi --insecure aggregates --resource-type loadbalancer --needed-overlap 0 \
"(aggregate sum (metric network.services.lb.loadbalancer mean))" \
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

- To aggregate the number of Kubernetes clusters:

Important
Kubernetes clusters may be created with an empty project ID. In this case, specify `None` for the `project_id` attribute.

```
# gnocchi --insecure aggregates --resource-type coe_cluster --needed-overlap 0 \
"(aggregate sum (metric magnum.cluster mean))" \
"project_id=75521ab61d1f4e9090aac5836c219492" \
--start 2021-07-18T12:00:00 --stop 2021-07-19T12:00:00
```

Configuring memory policy for the storage services

You can configure memory limits and guarantees for the storage services at runtime by using the `vinfra memory-policy vstorage-services` commands. You can do this for the entire cluster or a specific node.

The following memory parameters can be configured manually:

- Memory guarantee
- Swap size
- Page cache (which, in turn, is set using cache ratio, minimum, and maximum)

Page cache is calculated according to the following formula:

```
$PAGE_CACHE = minimum <= ratio * $TOTAL_MEMORY <= maximum
```

The `minimum` and `maximum` values are hard limits that are applied if the `ratio * $TOTAL_MEMORY` value is outside these limits.

To better understand how page cache size is calculated, consider the following examples:

Page cache examples

	Example 1 **(cache size is within limits)**	**Example 2** **(cache size equals minimum)**	**Example 3** **(cache size equals maximum)**
Total memory	4 GiB	4 GiB	4 GiB
Cache ratio	0.5	0.1	0.9
Cache minimum	1 GiB	2 GiB	1 GiB
Cache maximum	3 GiB	3 GiB	3 GiB
Cache size	2 GiB	2 GiB	3 GiB

If memory parameters are set both per node and per cluster, the per-node ones are applied. If no memory parameters are configured manually, the memory management is performed automatically by the `vcmmd` daemon as follows:

- Each CS (for example, storage disk) requires 512 MiB of RAM for page cache.
- The page cache minimum is 1 GiB.
- If the total memory is less than 48 GiB, the page cache maximum is calculated as two-thirds of it.
- If the total memory is greater than 48 GiB, the page cache maximum is 32 GiB.

To check the current memory parameters for the storage services set by `vcmmd`, run:

```
# vcmmdctl list
name                                type active guarantee    limit swap   cache
<...>
vstorage.slice/vstorage-services.sl… SRVC    yes   1310720 24522132    0 1048576
```

vinfra memory-policy vstorage-services per-cluster change

Change per-cluster memory parameters:

```
usage: vinfra memory-policy vstorage-services per-cluster change
       [--guarantee <guarantee>] [--swap <swap>] [--cache-ratio <cache-ratio>
       --cache-minimum <cache-minimum> --cache-maximum <cache-maximum>]
```

`--guarantee <guarantee>`

Guarantee, in bytes

`--swap <swap>`

Swap size, in bytes, or -1 if unlimited

`--cache-ratio <cache-ratio>`

Cache ratio from 0 to 1 inclusive

`--cache-minimum <cache-minimum>`

Minimum cache, in bytes

`--cache-maximum <cache-maximum>`

Maximum cache, in bytes

Example:

```
# vinfra memory-policy vstorage-services per-cluster change \
--guarantee 8796093022208 --swap 1099511627776 --cache-ratio 0.5 \
--cache-minimum 1099511627776 --cache-maximum 3298534883328
+-----------+------------------------+
| Field     | Value                  |
+-----------+------------------------+
| cache     | maximum: 3298534883328 |
|           | minimum: 1099511627776 |
|           | ratio: 0.5             |
| guarantee | 8796093022208          |
| swap      | 1099511627776          |
+-----------+------------------------+
```

This command sets the storage services memory parameters for all nodes in the storage cluster as follows:

- The memory guarantee to 8 GB
- The swap size to 1 GB
- The page cache limits: the minimum to 1 GB, the maximum to 3 GB, and the cache ratio to 0.5

vinfra memory-policy vstorage-services per-cluster show

Show per-cluster memory parameters:

```
usage: vinfra memory-policy vstorage-services per-cluster show
```

Example:

```
# vinfra memory-policy vstorage-services per-cluster show
+-----------+------------------------+
| Field     | Value                  |
+-----------+------------------------+
| cache     | maximum: 3298534883328 |
|           | minimum: 1099511627776 |
|           | ratio: 0.5             |
| guarantee | 8796093022208          |
| swap      | 1099511627776          |
+-----------+------------------------+
```

This command lists the storage services memory parameters for all nodes in the storage cluster.

vinfra memory-policy vstorage-services per-cluster reset

Reset per-cluster parameters to default:

```
usage: vinfra memory-policy vstorage-services per-cluster reset [--guarantee]
                                                                [--swap] [--cache]
```

`--guarantee`

Reset only the guarantee.

`--swap`

Reset only the swap size.

`--cache`

Reset only cache values.

Example:

```
# vinfra memory-policy vstorage-services per-cluster reset --cache
+-----------+---------------+
| Field     | Value         |
+-----------+---------------+
| cache     |               |
| guarantee | 8796093022208 |
| swap      | 1099511627776 |
+-----------+---------------+
```

This command resets the manually configured page cache limits to default for all nodes in the storage cluster.

vinfra memory-policy vstorage-services per-node change

Change per-node memory parameters:

```
usage: vinfra memory-policy vstorage-services per-node change
       [--guarantee <guarantee>] [--swap <swap>] [--cache-ratio <cache-ratio>
       --cache-minimum <cache-minimum> --cache-maximum <cache-maximum>]
       --node <node>
```

`--guarantee <guarantee>`

Guarantee, in bytes

`--swap <swap>`

Swap size, in bytes, or -1 if unlimited

`--cache-ratio <cache-ratio>`

Cache ratio from 0 to 1 inclusive

`--cache-minimum <cache-minimum>`

Minimum cache, in bytes

`--cache-maximum <cache-maximum>`

Maximum cache, in bytes

`--node <node>`

Node ID or hostname

Example:

```
# vinfra memory-policy vstorage-services per-node change \
--guarantee 8796093022208 --swap 1099511627776 --cache-ratio 0.5 \
--cache-minimum 1099511627776 --cache-maximum 3298534883328 \
--node 7ffa9540-5a20-41d1-b203-e3f349d62565
+-----------+------------------------+
| Field     | Value                  |
+-----------+------------------------+
| cache     | maximum: 3298534883328 |
|           | minimum: 1099511627776 |
|           | ratio: 0.5             |
| guarantee | 8796093022208          |
| swap      | 1099511627776          |
+-----------+------------------------+
```

This command sets the storage services memory parameters for the node with the ID `7ffa9540-5a20-41d1-b203-e3f349d62565` as follows:

- The memory guarantee to 8 GB
- The swap size to 1 GB
- The page cache limits: the minimum to 1 GB, the maximum to 3 GB, and the cache ratio to 0.5

vinfra memory-policy vstorage-services per-node show

Show per-node memory parameters:

```
usage: vinfra memory-policy vstorage-services per-node show --node <node>
```

`--node <node>`

Node ID or hostname

Example:

```
# vinfra memory-policy vstorage-services per-node show \
--node 7ffa9540-5a20-41d1-b203-e3f349d62565
+-----------+-------------------------+
| Field     | Value                   |
+-----------+-------------------------+
| cache     | maximum: 13194139533312 |
|           | minimum: 8796093022208  |
|           | ratio: 0.7              |
| guarantee | 8796093022208           |
| swap      | 1099511627776           |
+-----------+-------------------------+
```

This command lists the storage services memory parameters set for the node with the ID `7ffa9540-5a20-41d1-b203-e3f349d62565`.

vinfra memory-policy vstorage-services per-node reset

Reset per-node memory parameters to defaults:

```
usage: vinfra memory-policy vstorage-services per-node reset [--guarantee]
                                                             [--swap] [--cache]
                                                             --node <node>
```

`--guarantee`

Reset only the guarantee.

`--swap`

Reset only the swap size.

`--cache`

Reset only the cache values.

`--node <node>`

Node ID or hostname

Example:

```
# vinfra memory-policy vstorage-services per-node reset --cache \
--node 7ffa9540-5a20-41d1-b203-e3f349d62565
+-----------+---------------+
| Field     | Value         |
+-----------+---------------+
| cache     |               |
| guarantee | 8796093022208 |
| swap      | 1099511627776 |
+-----------+---------------+
```

This command resets the manually configured page cache limits to default for the node with the ID `7ffa9540-5a20-41d1-b203-e3f349d62565`.

Configuring memory for virtual machines

You can configure the amount of memory that can be provisioned for virtual machines by setting the RAM overcommitment ratio. This is the ratio of the amount of maximum reserved RAM to physical. The default ratio is 1, which means that you cannot provision more than the amount of physical RAM available on all of the compute nodes. By increasing the ratio, you can increase the number of virtual machines running in the compute cluster at the cost of reducing their performance. The maximum recommended RAM overcommitment ratio is 1.5.

Memory overcommitment for virtual machines is only available if all of the compute nodes have enough swap space. Before enabling RAM overcommitment, you need to calculate the required swap space, and then configure it on each compute node.

Adding swap space

To support the RAM overcommitment ratio, you need to add swap space. The swap size depends on the chosen RAM overcommitment ratio and can be calculated by using the following formula:

```
(total RAM - RAM used for system) * (RAM overcommitment ratio - 1)
```

To better understand how the minimum swap size is calculated, consider the following example:

- The total amount of physical RAM on a compute node is 24 GiB
- 8 GiB of RAM is reserved for the system
- The desired RAM overcommitment ratio is 1.5

According to the formula, the minimum required swap size will be 8 GiB.

After calculating the required swap size, proceed to configuring swap space by creating a swap file. On each node in the compute cluster, execute the `configure-swap.sh` script specifying the desired swap size:

Important
To be able to create a swap file, the root directory must have 100 GiB of free space after the swap file creation. For example, to create a swap file of 8 Gib, ensure that at least 108 GiB is available in the root directory.

```
# /usr/libexec/vstorage-ui-agent/bin/configure-swap.sh -s 8192
```

The script creates a swap file, prepares the swap space, and adds the swap mount point to `/etc/fstab`.

To check that the swap file is successfully created, run:

```
# swapon -s
Filename                Type         Size    Used    Priority
/dev/sda3               partition    8258556    0    -2
/swapfile0              file         8389628    0    -3
```

After the swap file is created, its size cannot be modified. To increase the swap size, you can add another swap file by running:

```
# /usr/libexec/vstorage-ui-agent/bin/configure-swap.sh -s 8192 --append
```

Enabling and disabling RAM overcommitment

RAM overcommitment for virtual machines is only available if all of the compute nodes have enough swap space.

You can enable RAM overcommitment when creating the compute cluster or afterwards by using either the `--custom-param` or `--nova-compute-ram-allocation-ratio` option (refer to "vinfra service compute create" (p. 82) or "Changing compute cluster parameters" (p. 89)). For example, to set the RAM overcommitment ratio to 1.5, run:

```
# vinfra service compute set --nova-compute-ram-allocation-ratio 1.5
```

To check that the ratio is successfully modified, execute the `vinfra service compute show` command:

```
# vinfra service compute show
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| <...>        | <...>                                   |
| options      | cpu_model: ''                           |
|              | custom_params:                          |
|              | - config_file: nova.conf                |
|              |   property: ram_allocation_ratio        |
|              |   section: DEFAULT                      |
```

```
|              |   service_name: nova-compute              |
|              |   value: 1.5                              |
| <...>        | <...>                                     |
+--------------+-------------------------------------------+
```

To disable RAM overcommitment for virtual machines, change the ratio to 1:

```
# vinfra service compute set --nova-compute-ram-allocation-ratio 1
```

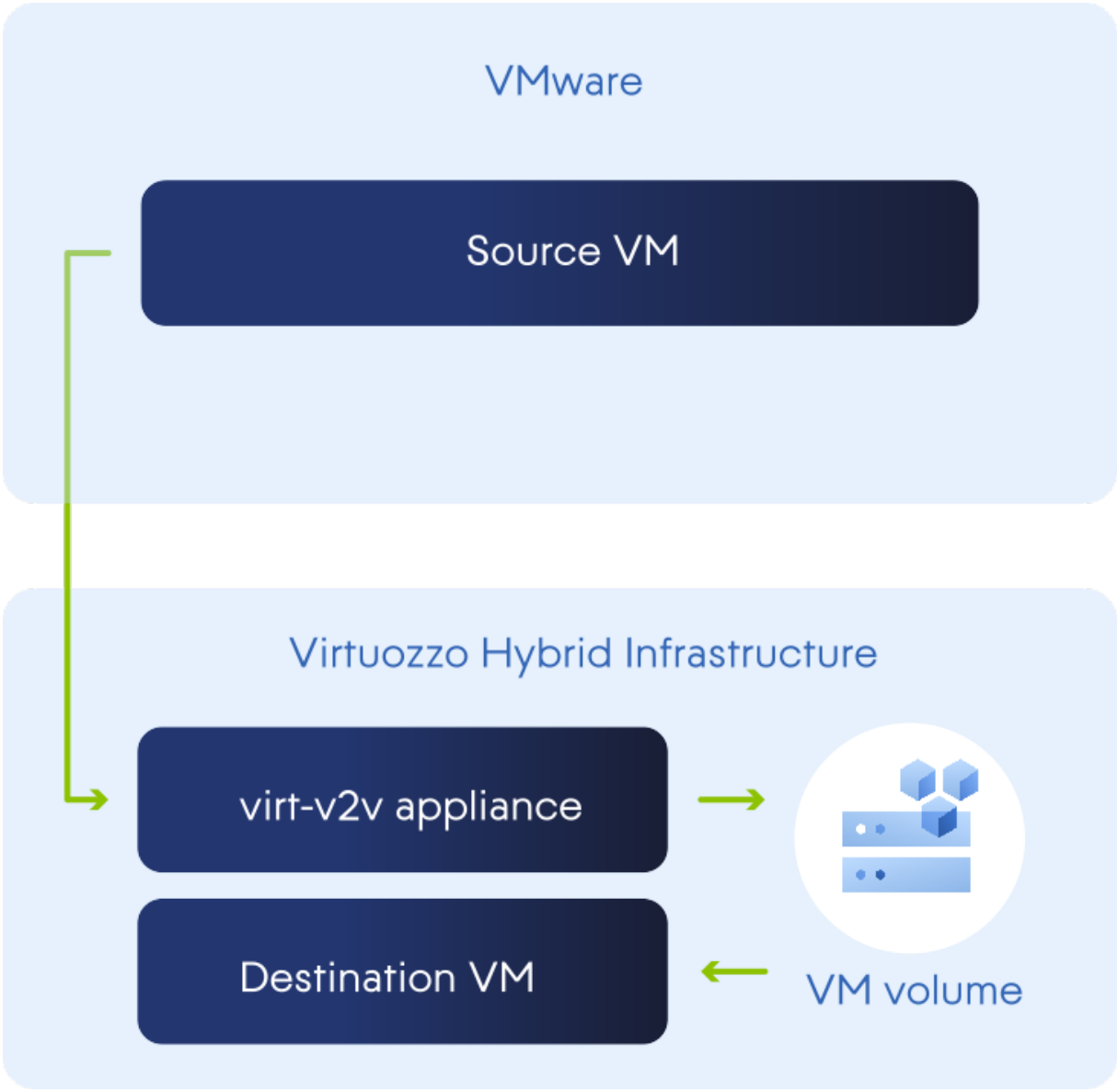
In this case, the swap space is not required anymore and the swap file can be removed. Execute the following:

```
# /usr/libexec/vstorage-ui-agent/bin/configure-swap.sh --remove-all
```

Run this script on each node in the compute cluster to remove the swap file from all of the compute nodes.

Migrating virtual machines from VMware vCenter

Starting from Virtuozzo Hybrid Infrastructure 3.5, you can migrate virtual machines from VMware vCenter 5.0 or newer to Virtuozzo Hybrid Infrastructure using the `virt-v2v` tool. You will need to create a `virt-v2v` appliance virtual machine to transfer and convert the data from.



Deploying the appliance virtual machine

To create a `virt-v2v` appliance VM, follow these steps:

1. Download the `virt-v2v` appliance image from the official repository.
2. Upload the image to Virtuozzo Hybrid Infrastructure. For example:

```
# vinfra service compute image create virt-v2v-img \
--file vmware_to_vhi.qcow2
```

3. Create an SSH key for the appliance if you do not have one. For example:

```
# vinfra service compute key create publickey \
--public-key virt-v2v-app-key.pub
```

4. Create a virtual machine and deploy the uploaded image in it. The VM needs at least two CPUs, 4 GiB RAM, and enough storage space to accomodate the largest VM to be migrated to Virtuozzo Hybrid Infrastructure. It must also be connected to the network that handles the **Compute API** traffic type and the network with access to VMware vCenter API. For example:

```
# vinfra service compute server create virt-v2v-appliance \
--flavor medium --key-name <key>
--network id=<compute_API> --network id=<vcenter_API> \
--volume source=image,id=virt-v2v-img,size=<size>
```

   Where:

   - `<key>` is the SSH key to authorize in the appliance VM.
   - `<compute_API>` is the network that handles the traffic type **Compute API**.
   - `<vcenter_API>` is the network that can access the VMware vCenter API.
   - `<size>` is the disk size. For online migration, it must be enough to accomodate the largest VM of all you are going to migrate. For offline migration, it must be enough to accomodate twice as much.

Setting up authentication in the appliance virtual machine

1. Log in to the appliance VM as the `admin` user with the SSH key.
2. Get root privileges, for example, with `sudo -i`.
3. Create a bash script that will export OpenStack credentials:

```
# cat > user-openrc.sh << EOF
export OS_PROJECT_DOMAIN_NAME=Domain_name
export OS_USER_DOMAIN_NAME=Domain_name
export OS_PROJECT_NAME=Project_name
export OS_USERNAME=user_name
export OS_PASSWORD=Password
export OS_AUTH_URL=https://<admin_panel_IP_addr>:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_INSECURE=true
export NOVACLIENT_INSECURE=true
export NEUTRONCLIENT_INSECURE=true
export CINDERCLIENT_INSECURE=true
export LIBGUESTFS_BACKEND=direct
EOF
```

   Note
   You will need the administrator credentials for the project that the appliance VM belongs to.

4. Copy the OpenStack root CA certificate and CA keys from the Virtuozzo Hybrid Infrastructure management node:

```
# scp root@<MN_IP>:/usr/libexec/vstorage-ui-backend/ca/ca.* \
/etc/pki/ca-trust/source/anchors/
# update-ca-trust extract
```

Where `<MN_IP>` is the management node IP address. For more information, refer to "Securing OpenStack API traffic with SSL" (p. 329).

5. Create a file with the VMware vCenter password to supply to `virt-v2v`. For example:

```
# echo $vCenterPass > password.txt
```

Alternatively, you can enter the password during migration or supply it to `virt-v2v` with the `--password-file` option.

Migrating virtual machines to Virtuozzo Hybrid Infrastructure online

Take note of the following before you proceed:

- You can migrate VMs created on vCenter 5.0 or newer.
- Remove VMware tools from Windows VMs before the migration to avoid issues on boot afterwards. VMware tools will be removed from Linux guests automatically.

To migrate a VM from VMware vCenter online, do the following:

1. Log in to the appliance VM as the `admin` user with the SSH key.
2. Get root privileges, for example, with `sudo -i`.
3. Set OpenStack credentials:

```
# source user-openrc.sh
```

4. Test the connection between libvirt and VMware vCenter. For example:

```
# virsh -c 'vpx://<domain>%5c<user>@<hostname>?no_verify=1' list --all
Enter root's password for vcenter.example.com: ***
+----+--------------+----------+
| Id | Name         | <...>    |
+----+--------------+----------+
| -  | Fedora 20    | <...>    |
| -  | Windows 2008 | <...>    |
+----+--------------+----------+
```

Where `<hostname>` is the name of the VMware ESXi host that runs virtual machines. Its full path looks like `<vCenter_hostname>/<datacenter_name>/<cluster_name>/<server_hostname>` and can be found in VMware vCenter.

If the VPX username contains a backslash (for example, `<domain>\<user>`), escape it with `%5c`: `<domain>%5c<user>`. Similarly, escape spaces with `%20`.

5. Check the OpenStack connection and find out the `virt-v2v` appliance ID. For example:


```
# openstack --insecure server list
+--------------------------------------+--------------------+--------+-------+
| ID                                   | Name               | Status | <...> |
+--------------------------------------+--------------------+--------+-------+
| 635ae4cc-4c01-461a-ae63-91ca4187a7b1 | virt-v2v-appliance | ACTIVE | <...> |
+--------------------------------------+--------------------+--------+-------+
```

6. Shut down the VM. Windows VMs must be shut down gracefully for the migration to be successful.
7. Migrate the VM to a volume in Virtuozzo Hybrid Infrastructure specifying a storage policy. To list available storage policies, run `vinfra service compute storage-policy list` in Virtuozzo Hybrid Infrastructure. For example:

   ```
   # virt-v2v -ip password.txt -ic 'vpx://<domain>%5c<user>@<hostname>\
   ?no_verify=1' 'Windows 2008' -o openstack -oo server-id=635ae4cc-\
   4c01-461a-ae63-91ca4187a7b1 -os <policy_name>
   ```

   Where `<policy_name>` is the storage policy for the converted volume.
8. Find out the new volume's ID or name. For example:

   ```
   # openstack --insecure volume list
   +----------------------+------+-----------+------+-------------+
   | ID                   | Name | Status    | Size | Attached to |
   +----------------------+------+-----------+------+-------------+
   | 024b6843-2de3-<...> | sda1 | available |   64 |             |
   +----------------------+------+-----------+------+-------------+
   ```

9. If the VM used UEFI firmware, manually set the correct OS distribution and UEFI firmware type for the converted volume. To list available distributions, run `vinfra service compute show` in Virtuozzo Hybrid Infrastructure. For example:

   ```
   # openstack --insecure volume set sda1 --image-property os_distro=win2k8
   # openstack --insecure volume set sda1 --image-property hw_firmware_type=uefi
   ```

10. In Virtuozzo Hybrid Infrastructure, create a virtual machine based on the new volume. For example:

    ```
    # vinfra service compute server create migratedvm --network id=private \
    --network id=public --volume source=volume,id=sda1,size=64 --flavor medium
    ```

After the migration, it is recommended to install the guest tools inside the VM as described in "Installing guest tools" in the Administrator Guide. The guest tools installation will prevent possible problems with guest OS interaction via the VNC console.

Migrating virtual machines to Virtuozzo Hybrid Infrastructure offline

If the network connection between the `virt-v2v` appliance VM and VMware vCenter is inferior, you can manually copy the VMs to a USB drive, connect it to the `virt-v2v` appliance VM, and convert

them to Virtuozzo Hybrid Infrastructure.

Take note of the following before you proceed:

- You can migrate VMs created on vCenter 5.0 or newer.
- Remove VMware tools from Windows VMs before the migration to avoid issues on boot afterwards. VMware tools will be removed from Linux guests automatically.

To migrate a VM from VMware vCenter offline, do the following:

1. Copy all of the VM files, including `vmdk` and `vmx`, to a USB drive.
2. Attach the USB drive to a host in the same local network as the appliance VM.
3. Log in to the appliance VM as the `admin` user with the SSH key.
4. Get root privileges, for example, with `sudo -i`.
5. Copy VM files to the appliance VM, for example, using `rsync` or `scp`.
6. Set OpenStack credentials:

   ```
   # source user-openrc.sh
   ```

7. Migrate the VM to a volume in Virtuozzo Hybrid Infrastructure specifying a storage policy. To list available storage policies, run `vinfra service compute storage-policy list` in Virtuozzo Hybrid Infrastructure. For example:

   ```
   # virt-v2v -i libvirtxml <VM_config> -o openstack \
   -oo server-id=635ae4cc-4c01-461a-ae63-91ca4187a7b1 -os <policy_name>
   ```

   Where `<VM_config>` is the VM configuration file in the `vmx` format and `<policy_name>` is the storage policy for the converted volume.
8. Find out the new volume's ID or name. For example:

   ```
   # openstack --insecure volume list
   +----------------------+------+-----------+------+-------------+
   | ID                   | Name | Status    | Size | Attached to |
   +----------------------+------+-----------+------+-------------+
   | 024b6843-2de3-<...> | sda1 | available |   64 |             |
   +----------------------+------+-----------+------+-------------+
   ```

9. If the VM used UEFI firmware, manually set the correct OS distribution and UEFI firmware type for the converted volume. To list available distributions, run `vinfra service compute show` in Virtuozzo Hybrid Infrastructure. For example:

   ```
   # openstack --insecure volume set sda1 --image-property os_distro=win2k8
   # openstack --insecure volume set sda1 --image-property hw_firmware_type=uefi
   ```

10. In Virtuozzo Hybrid Infrastructure, create a virtual machine based on the new volume. For example:

    ```
    # vinfra service compute server create migratedvm --network id=private \
    --network id=public --volume source=volume,id=sda1,size=64 --flavor medium
    ```

After the migration, it is recommended to install the guest tools inside the VM as described in "Installing guest tools" in the Administrator Guide. The guest tools installation will prevent possible problems with guest OS interaction via the VNC console.

Changing the default load balancer flavor

By default, a load balancer is created by using the private `amphora` flavor that cannot be managed via `vinfra`. You can, however, change it by using the OpenStack command-line tool. Do the following:

1. Connect to the OpenStack command-line interface as a system administrator to authorize further OpenStack commands (refer to "Connecting to OpenStack command-line interface" (p. 326)):

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Check if the default `amphora` flavor exists:

```
# openstack --insecure flavor list --all
+---------+---------+-------+------+-----------+-------+-----------+
| ID      | Name    |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+---------+---------+-------+------+-----------+-------+-----------+
| 100     | tiny    |   512 |    0 |         0 |     1 | True      |
| 101     | small   |  2048 |    0 |         0 |     1 | True      |
| 102     | medium  |  4096 |    0 |         0 |     2 | True      |
| 103     | large   |  8192 |    0 |         0 |     4 | True      |
| 104     | xlarge  | 16384 |    0 |         0 |     8 | True      |
| amphora | amphora |  4096 |   30 |         0 |     2 | False     |
+---------+---------+-------+------+-----------+-------+-----------+
```

3. Delete this flavor:

```
# openstack --insecure flavor delete amphora
```

4. Create a new `amphora` flavor with custom parameters. For example:

```
# openstack --insecure flavor create amphora --id amphora --ram 8192 \
--vcpus 4 --disk 60 --private
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| disk                       | 60      |
| id                         | amphora |
| name                       | amphora |
| os-flavor-access:is_public | False   |
| properties                 |         |
| ram                        | 8192    |
| rxtx_factor                | 1.0     |
```



```
| swap                        |         |
| vcpus                       | 4       |
+-----------------------------+---------+
```

5. Change the load balancer flavor by performing its failover. For example:

```
# openstack --insecure loadbalancer failover mylbaas
```

The load balancer `mylbaas` will be recreated with 4 vCPUs, 8 GB of RAM, and 30 GB of disk space.

Changing parameters in OpenStack configuration files

You can modify the following parameters in the OpenStack configuration files:

- `ram_weight_multiplier` in `/etc/kolla/nova-scheduler/nova.conf` defines how compute nodes with available RAM are weighed. With a positive value, virtual machines are placed on nodes with more available RAM, and thus, evenly spread across all compute nodes. In this case, however, you may end up unable to launch large VMs on particular nodes while having plenty of free RAM in the entire cluster. To optimize VM distribution and fill up nodes as much as possible, you can set this parameter to a negative value.

  Valid values are integer or float. The default value is 1.0.
- `scheduler_host_subset_size` in `/etc/kolla/nova-scheduler/nova.conf` defines a number of compute nodes best suited for a new VM, one of which is randomly chosen by the scheduler.

  Valid values are 1 or greater. Any value less than 1 is treated as 1. The higher the value, the less optimal the chosen node may be for a VM. The default value is 1.
- `vxlan_udp_port` in `/etc/kolla/neutron-openvswitch-agent/ml2_conf.ini` specifies the UDP port used for VXLAN tunnels. When changing the port, `iptables` rules are automatically configured for both the old and new ports.

  The default port is 4789.
- `cpu_allocation_ratio` in `/etc/kolla/nova-compute/nova.conf` defines the virtual CPU to physical CPU allocation ratio.

  Valid values are positive integer or float. The default value is 8.0.

  Note
  Changing CPU allocation ratio will not affect virtual CPUs already provisioned for virtual machines.

- `ram_allocation_ratio` in `/etc/kolla/nova-compute/nova.conf` defines the maximum reserved RAM to physical RAM allocation ratio.

  Valid values are positive integer or float. The default value is 1.0. The maximum recommended value is 1.5.

  Note
  Changing RAM allocation ratio will not affect RAM already provisioned for virtual machines.

The parameters can be set when creating the compute cluster or afterwards by using either the `--custom-param` option or the shortcuts for each parameter (refer to "vinfra service compute create" (p. 82) or "Changing compute cluster parameters" (p. 89)). For example, to change `ram_weight_multiplier` and `vxlan_udp_port`, run:

```
# vinfra service compute set --nova-scheduler-ram-weight-multiplier -1 \
--neutron-openvswitch-vxlan-port 4787
```

To check that the custom parameters are successfully modified, execute the `vinfra service compute show` command:

```
# vinfra service compute show
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| <...>        | <...>                                   |
| options      | cpu_model: ''                           |
|              | custom_params:                          |
|              | - config_file: nova.conf                |
|              |   property: ram_weight_multiplier       |
|              |   section: DEFAULT                      |
|              |   service_name: nova-scheduler          |
|              |   value: -1.0                           |
|              | - config_file: ml2_conf.ini             |
|              |   property: vxlan_udp_port              |
|              |   section: agent                        |
|              |   service_name: neutron-openvswitch-agent |
|              |   value: 4787                           |
|              | notification_forwarding: disabled       |
| status       | active                                  |
+--------------+-----------------------------------------+
```

The applied changes are consistent on all compute nodes and not overwritten after product updates and upgrades.

Configuring retention policy for Prometheus metrics

The Prometheus service used for monitoring the cluster runs and stores its data on the management node. By default, Prometheus metrics are stored for seven days. This retention period can be insufficient for troubleshooting purposes. You can increase this period in the Prometheus configuration file by doing the following:

1. On the management node, open the `/etc/sysconfig/prometheus` file to edit, set the needed retention period for the `STORAGE_RETENTION` option, and then save your changes. For example:

   ```
   STORAGE_RETENTION="--storage.tsdb.retention.time=30d"
   ```

2. Restart the Prometheus service:

```
systemctl restart prometheus.service
```

If high availability is enabled in the storage cluster, repeat these steps for the other two management nodes.

However, with a long retention period, the root partition where the data is stored may run out of free space. To prevent this, you can define the maximum size for the Prometheus metrics. The oldest data will be removed first. To change the time retention policy to the size retention policy, do the following:

1. On the management node, open the `/etc/sysconfig/prometheus` file to edit, change the flag for the `STORAGE_RETENTION` option, and then save your changes. For example:

```
STORAGE_RETENTION="--storage.tsdb.retention.size=10GB"
```

2. Restart the Prometheus service:

```
systemctl restart prometheus.service
```

If high availability is enabled in the storage cluster, repeat these steps for the other two management nodes.

Exiting the rescue mode for Windows virtual machines

There might be an issue of exiting the rescue mode for a Windows VM. If in the rescue mode you set the original system disk online, its ID becomes the same as that of the rescue disk. Then, when you try to exit the rescue mode, the boot loader cannot find the proper boot disk. To resolve the ID conflict, follow the steps:

1. With the VM in the rescue mode, open the **Disk Management** window and note the numbers of the original system disk (offline) and the rescue disk (online). Set the original system disk to **Online**.
2. To edit the boot configuration, enter the following command in the **Command Prompt** window:

```
> bcdedit /store <the original system disk name>:\boot\bcd
```

3. Review the output and check that the rescue disk is the target for objects in the output (`partition=<the rescue disk name>`).

   If the objects do not point to drive C, fix it with the following commands:

```
> bcdedit /store <the original system disk name>:\boot\bcd \
/set {default} osdevice partition=<the rescue disk name>:
> bcdedit /store <the original system disk name>:\boot\bcd \
/set {default} device partition=<the rescue disk name>:
> bcdedit /store <the original system disk name>:\boot\bcd \
/set {bootmgr} device partition=<the rescue disk name>:
```

```
> bcdedit /store <the original system disk name>:\boot\bcd \
/set {memdiag} device partition=<the rescue disk name>:
```

4. To view the available disks, enter the following commands in the command line:

   ```
   > DISKPART
   > LIST DISK
   ```

   Match the disk number and name to those displayed in the **Disk Management** window.
5. To get the ID of the rescue disk, run the following commands:

   ```
   > SELECT DISK <the rescue disk number>
   > UNIQUEID DISK
   ```

   Record the disk ID, you will need it later.
6. Change this ID by using the following command:

   ```
   > UNIQUEID DISK id=<any hex value of 8 characters>
   ```

   Make sure that the value has changed with the `UNIQUEID DISK` command.
7. Assign the ID that you recorded previusly to the original system disk:

   ```
   > SELECT DISK <the original system disk number>
   > UNIQUEID DISK id=<the recorded disk ID>
   ```

   Make sure that the value has changed with the `UNIQUEID DISK` command.

You should now be able to exit the rescue mode.

Assigning users to multiple domains

Using the `vinfra` tool, system administrators are able to create special service users that can be used by third-party applications to access the compute API with administrator privileges. These users cannot log in to the admin or self-service panels. Service users are similar to system administrators with the **Compute** permission: they exist only within the **Default** domain and can view and manage all objects in the compute cluster, including compute nodes. You can assign service users to domains, thus giving them ability to create compute objects in projects of these assigned domains (for example, to create a VM from a backup).

Service users can view virtual machines in all existing projects by specifying the `all_tenants` query parameter for the `GET /servers` request (refer to the OpenStack API documentation).

To assign a service user to a domain, use the `--assign-domain <domain> <roles>` option for the "vinfra domain user create" (p. 251) or "vinfra domain user set" (p. 255) command. Specify `Default` for the `--domain` option and `compute` as a service account role. For example, to create the service user `my-service-user` and assign it to the `mydomain` and `mydomain2` domains, execute:

```
# vinfra domain user create my-service-user --domain default --assign-domain mydomain \
compute --assign-domain mydomain2 compute
Password:
+--------------------+-----------------------------------------------+
| Field              | Value                                         |
+--------------------+-----------------------------------------------+
| assigned_domains   | - domain_id: 7e0d54797152424a9331ae904e220b88 |
|                    |   roles:                                      |
|                    |   - compute                                   |
|                    | - domain_id: 2929ff42b1e64884a05dea3011862aed |
|                    |   roles:                                      |
|                    |   - compute                                   |
| assigned_projects  | []                                            |
| description        |                                               |
| domain_id          | default                                       |
| domain_permissions | []                                            |
| email              |                                               |
| enabled            | True                                          |
| id                 | 91b185b711fb4f2b81b09a661df0dd27              |
| name               | my-service-user                               |
| role               | service_account                               |
| system_permissions | []                                            |
+--------------------+-----------------------------------------------+
```

To check that the created service user is successfully assigned to the two domains, use the OpenStack client. For example, if the management node IP address is 10.136.16.227, run:

```
# openstack --insecure --os-username my-service-user --os-user-domain-name \
Default --os-auth-url=https://10.136.16.227:5000/v3 federation domain list
Password:
+----------------------------------+---------+-----------+-------------+
| ID                               | Enabled | Name      | Description |
+----------------------------------+---------+-----------+-------------+
| 2929ff42b1e64884a05dea3011862aed | True    | mydomain  |             |
| 7e0d54797152424a9331ae904e220b88 | True    | mydomain2 |             |
+----------------------------------+---------+-----------+-------------+
```

You can also view the list of all projects within the assigned domains by using the command `openstack --insecure --os-username <username> --os-user-domain-name Default --os-auth-url=https://<MN_IP_address>:5000/v3 federation project list`.

To unassign a service user from a domain, use the `--unassign-domain <domain>` option for the "vinfra domain user set" (p. 255) command. For example:

```
# vinfra domain user set my-service-user --domain default \
--unassign-domain mydomain
+--------------------+-----------------------------------------------+
| Field              | Value                                         |
+--------------------+-----------------------------------------------+
| assigned_domains   | - domain_id: 7e0d54797152424a9331ae904e220b88 |
```



```
|                     |   roles:                                        |
|                     |   - compute                                     |
| assigned_projects   | []                                              |
| description         |                                                 |
| domain_id           | default                                         |
| domain_permissions  | []                                              |
| email               |                                                 |
| enabled             | True                                            |
| id                  | 6c32d26d3674448c8b4f1bf9825a85cc                |
| name                | my-service-user                                 |
| role                | service_account                                 |
| system_permissions  | []                                              |
+---------------------+-------------------------------------------------+
```

Using network QoS policies

You can use quality of service (QoS) policies to guarantee or limit network bandwidth for egress and ingress VM traffic in different projects. QoS policies can be applied to separate network ports and floating IP addresses, as well as to entire networks. In addition, you can set a QoS policy as default for a project to automatically assign the policy to all new networks created within the project. The default QoS policy will be applied to a network if no other policy is explicitly assigned during the network creation process.

When you assign a policy to a network, all ports connected to this network inherit the policy unless the port has a specific policy assigned to it. The assigned policy is applied to both existing and new virtual machines. Internal network ports, like DHCP, and internal router ports are excluded from network policy application.

QoS policy rules

You can create two rule types to define a QoS policy: bandwidth limit and minimum bandwidth.

Bandwidth limit

Provides bandwidth limitations on networks, ports or floating IPs. Any VM traffic that exceeds the specified rate will be dropped.

To define a bandwidth limit, specify the following parameters:

- `max_kbps`: The maximum rate, in Kbps, that the VM can send.
- `max_burst_kbps`: The maximum amount of data, in kbits, that the port can send in a VM if the token buffer is full. The token buffer replenishes at a `max_kbps` rate.

Important

- If the burst value is set too low, bandwidth usage will be throttled even with a proper bandwidth limit setting, resulting in a lower than expected bandwidth.
- If the configured burst value is too high, too few packets could be limited, resulting in a higher than expected bandwidth limit.

If you omit this parameter, the recommended burst value for TCP traffic will be applied, which defaults to 80% of the bandwidth limit. For example, if the bandwidth limit is 1000 kbps, then a burst value of 800 kbps is enough.

- `ingress` or `egress`: The direction of traffic the rule is applied to. For a VM, ingress indicates download, and egress indicates upload.

Minimum bandwidth

Provides minimum bandwidth guarantees on networks, ports or floating IPs. VM traffic will use no less than the specified bandwidth.

Important
A QoS policy with this rule type cannot be applied to entire virtual networks.

To define minimum bandwidth, specify the following parameters:

- `min-kbps`: The minimum bandwidth, in Kbps, guaranteed to a VM.
- `ingress` or `egress`: The direction of traffic the rule is applied to. For a VM, ingress indicates download, and egress indicates upload.

Rules of different types can be combined in one QoS policy. For example, you can create a bandwidth limit rule and a minimum bandwidth rule. Additionally, you can add rules of one type to a policy if the traffic direction of each rule is different. For example, you can create two bandwidth limit rules, one for egress traffic and one for ingress traffic.

Creating QoS policies

To create a QoS policy with rules, do the following:

1. Connect to the OpenStack command-line interface, as described in "Connecting to OpenStack command-line interface" (p. 326).
2. Create a QoS policy:
   - If you used the environment file for a system administrator, create a policy within a project it will be applied to:

```
# openstack --insecure network qos policy create \
--project 3823a2d908ea4dd6909a8f93a6f66018 policy1
+-------------+--------------------------------------------------+
| Field       | Value                                            |
+-------------+--------------------------------------------------+
```



```
| description |                                                    |
| id          | 8e2511c9-7db5-456c-b8ee-939f7729d981               |
| is_default  | False                                              |
| location    | Munch({'project': Munch({'domain_name': None,  |
|             | 'domain_id': None, 'name': None,               |
|             | 'id': u'3823a2d908ea4dd6909a8f93a6f66018'}),   |
|             | 'cloud': '', 'region_name': '', 'zone': None}) |
| name        | policy1                                            |
| project_id  | 3823a2d908ea4dd6909a8f93a6f66018                   |
| rules       | []                                                 |
| shared      | True                                               |
| tags        | []                                                 |
+-------------+----------------------------------------------------+
```

- If you used the environment file for a domain administrator, create a policy, as outlined in "Creating QoS policies as a domain administrator" (p. 365).

Important
To be able to create and manage network QoS policies, a domain administrator needs to have the quota manager role assigned.

3. Create a rule for the QoS policy:
   - To create a bandwidth limit, specify `bandwidth-limit` for the `--type` option and specify rule parameters. For example, to limit the egress traffic to 3 Mbps, run:

```
# openstack --insecure network qos rule create --type bandwidth-limit \
--max-kbps 3000 --max-burst-kbits 2400 --egress policy1
+----------------+-----------------------------------------------------+
| Field          | Value                                               |
+----------------+-----------------------------------------------------+
| direction      | egress                                              |
| id             | 6f036f09-d952-420d-986b-27c7eb14b2da                |
| location       | Munch({'project': Munch({'domain_name': Default, |
|                | 'domain_id': None, 'name': admin,                   |
|                | 'id': u'e215189c0472482f93e71d10e1245253'}),        |
|                | 'cloud': '', 'region_name': '', 'zone': None})      |
| max_burst_kbps | 2400                                                |
| max_kbps       | 3000                                                |
| name           | None                                                |
| project_id     |                                                     |
+----------------+-----------------------------------------------------+
```

   - To create a minimum bandwidth guarantee, specify `minimum-bandwidth` for the `--type` option and specify rule parameters. For example, to guarantee the minimum of 100 Kbps to the ingress traffic, run:

```
# openstack --insecure network qos rule create --type minimum-bandwidth \
--min-kbps 1000 --ingress policy1
+------------+-----------------------------------------------------+
```

```
| Field      | Value                                              |
+------------+----------------------------------------------------+
| direction  | ingress                                            |
| id         | 4eb79c67-e2b7-4ee7-845c-4cbe39f095cd               |
| location   | Munch({'project': Munch({'domain_name': Default,  |
|            | 'domain_id': None, 'name': admin,                  |
|            | 'id': u'e215189c0472482f93e71d10e1245253'}),       |
|            | 'cloud': '', 'region_name': '', 'zone': None})     |
| min_kbps   | 1000                                               |
| name       | None                                               |
| project_id |                                                    |
+------------+----------------------------------------------------+
```

Setting the default QoS policy

You can set the default QoS policy to a project. It will be automatically assigned to all new networks that you create within the project. Existing networks within the project will not inherit the default QoS policy, you will need to explicitly assign the policy to them.

To set a QoS policy as default, use the `--default` option. For example:

```
# openstack --insecure network qos policy set --default policy1
```

Each project can have only one default QoS policy. If you want to change the default policy, unset the old default policy first. To do it, specify the `--no-default` option. For example:

```
# openstack --insecure network qos policy set --no-default policy1
```

Assigning QoS policies

Alongside the default QoS policy, you can assign QoS policies to specific network ports, floating IP addresses, and entire networks. To do it, find out the ID of the required resource, and then set the policy as follows:

- To assign a QoS policy to a network port, execute:

```
# openstack --insecure port list
+--------------------------------------+-----+---------------------------+
| ID                                   | <…> | Fixed IP Addresses        |
+--------------------------------------+-----+---------------------------+
| c0ea690f-4993-4467-afd5-5389016a0658 |     | ip_address='10.136.18.133' |
+--------------------------------------+-----+---------------------------+
# openstack --insecure port set --qos-policy policy1 \
c0ea690f-4993-4467-afd5-5389016a0658
```

- To assign a QoS policy to a floating IP address, execute:


```
# openstack --insecure floating ip list
+--------------------------------------+---------------------+-----+
| ID                                   | Floating IP Address | <…> |
+--------------------------------------+---------------------+-----+
| 866203a2-4e1c-459f-807f-14ed563409f1 | 10.136.18.135       |     |
+--------------------------------------+---------------------+-----+
# openstack --insecure floating ip set --qos-policy policy1 \
866203a2-4e1c-459f-807f-14ed563409f1
```

- To assign a QoS policy to all ports in a network, execute:

```
# openstack --insecure network list
+--------------------------------------+----------+-----+
| ID                                   | Name     | <…> |
+--------------------------------------+----------+-----+
| c6ee561e-9cf7-489b-bbab-7bca557ee7a5 | public   |     |
+--------------------------------------+----------+-----+
# openstack --insecure network set --qos-policy policy1 public
```

Modifying QoS policy rules

You can modify QoS policy rules at runtime. Changes will take effect on all ports to which the policy is applied.

To edit a policy rule, specify new parameter values, the policy name, and its rule ID:

```
# openstack --insecure network qos policy show policy1
+-------+-----------------------------------------------------------------+
| Field | Value                                                           |
+-------+-----------------------------------------------------------------+
| <…>   |                                                                 |
| name  | policy1                                                         |
| rules | [{u'max_kbps': 3000, u'direction': u'ingress',                  |
|       |   u'qos_policy_id': u'8e2511c9-7db5-456c-b8ee-939f7729d981',  |
|       |   u'type': u'bandwidth_limit', u'max_burst_kbps': 2400,       |
|       |   u'id': u'6f036f09-d952-420d-986b-27c7eb14b2da'}]            |
| <…>   |                                                                 |
+-------+-----------------------------------------------------------------+
# openstack --insecure network qos rule set --max-kbps 2000 \
--max-burst-kbits 1600 --ingress policy1 6f036f09-d952-420d-986b-27c7eb14b2da
```

Unassigning QoS policies

Before unassigning a QoS policy, make sure it is not used by networks, ports, or IP addresses.

To unassign a QoS policy, detach the required resource from the policy as follows:

- To unassign a policy from a network port, execute:

```
# openstack --insecure port unset --qos-policy \
c0ea690f-4993-4467-afd5-5389016a0658
```

- To unassign a policy from a floating IP address, execute:

```
# openstack --insecure floating ip unset --qos-policy \
866203a2-4e1c-459f-807f-14ed563409f1
```

- To unassign a policy from a network, execute:

```
# openstack --insecure network set --no-qos-policy public
```

Allowing domain administrators to manage projects

You can allow domain administrators to manage projects within the assigned domain. With this permission, domain administrators can perform the following additional tasks by using the OpenStack command-line tool:

- Create, update, and delete projects
- Set and update project quotas
- Create and manage network QoS policies

Creating and assigning the quota manager role

To create a domain administrator that can manage projects, you need to create the quota manager role and assign it to a domain administrator. Do the following:

1. Connect to the OpenStack command-line interface as a system administrator to authorize further OpenStack commands (refer to "Connecting to OpenStack command-line interface" (p. 326)).

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Create the `quota_manager` role:

```
# openstack --insecure role create 'quota_manager'
```

3. Create a domain and a domain administrator by using the `vinfra` tool. For example:

```
# vinfra domain create test
+----------------+----------------------------------+
| Field          | Value                            |
+----------------+----------------------------------+
| description    |                                  |
| enabled        | True                             |
```



```
| id             | b41c5bd8ca1e43f19f9720390c2869d5 |
| name           | test                             |
| projects_count | 0                                |
+----------------+----------------------------------+
# vinfra domain user create --domain test --domain-permissions domain_admin testuser
Password:
+--------------------+----------------------------------+
| Field              | Value                            |
+--------------------+----------------------------------+
| assigned_domains   | []                               |
| assigned_projects  | []                               |
| description        |                                  |
| domain_id          | b41c5bd8ca1e43f19f9720390c2869d5 |
| domain_permissions | - domain_admin                   |
| email              |                                  |
| enabled            | True                             |
| id                 | 73a8420bf2fc49998704701c6d36c255 |
| name               | testuser                         |
| role               | domain_admin                     |
| system_permissions | []                               |
| tags               | []                               |
+--------------------+----------------------------------+
```

4. Assign the `quota_manager` role to the new user:

```
# openstack --insecure role add --user-domain test --user testuser \
--domain test quota_manager
# openstack --insecure role add --user-domain test --user testuser \
--domain test quota_manager --inherited
```

5. Prepare an environment file for the new user. For example:

```
# vi domain-admin.sh
export OS_PROJECT_DOMAIN_NAME=test
export OS_USER_DOMAIN_NAME=test
export OS_DOMAIN_NAME=test
export OS_USERNAME=testuser
export OS_PASSWORD=1q2w3e
export OS_AUTH_URL=https://127.0.0.1:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_TYPE=password
export OS_INSECURE=true
export PYTHONWARNINGS="ignore:Unverified HTTPS request is being made"
export NOVACLIENT_INSECURE=true
export NEUTRONCLIENT_INSECURE=true
export CINDERCLIENT_INSECURE=true
export OS_PLACEMENT_API_VERSION=1.22
```

Managing projects as a domain administrator

To manage a project as a domain administrator, do the following:

1. Use the environment file for the domain administrator to authorize further OpenStack commands:

```
# unset OS_PROJECT_NAME; unset OS_DOMAIN_NAME; source domain-admin.sh
```

2. List all accessible domains to get the domain ID:

```
# openstack --insecure federation domain list
+----------------------------------+---------+------+-------------+
| ID                               | Enabled | Name | Description |
+----------------------------------+---------+------+-------------+
| b41c5bd8ca1e43f19f9720390c2869d5 | True    | test |             |
+----------------------------------+---------+------+-------------+
```

3. Create a project. For example:

```
# openstack --insecure project create \
--domain b41c5bd8ca1e43f19f9720390c2869d5 testproject
```

4. To set and update project quotas, authorize in the new project. For example:

```
# export OS_PROJECT_NAME=testproject
```

5. Set a quota for the project:
   - To limit the number of virtual CPUs, use the `--cores` option. For example, to limit the number of vCPUs to 128, execute:

     ```
     # openstack --insecure quota set --cores 128 testproject
     ```

   - To limit the amount of RAM, use the `--ram` option. For example, to limit the amount of RAM to 100 Gib, execute:

     ```
     # openstack --insecure quota set --ram 102400 testproject
     ```

   - To limit the storage space size, specify the policy with the `--volume-type` option and the required space with the `--gigabytes` option. For example, to limit the storage space size for the `default` storage policy to 1 Tib, execute:

     ```
     # openstack --insecure quota set --volume-type default \
     --gigabytes 1024 testproject
     ```

   - To limit the number of floating IP addresses, use the `--floating-ips` option. For example, to limit the number of floating IP addresses to 128, execute:

     ```
     # openstack --insecure quota set --floating-ips 128 testproject
     ```

   - To limit the number of Kubernetes clusters, get the project ID first, and then execute the `coe quotas create` command with the `--resource` and `--hard-limit` options. For example, to limit the number of Kubernetes clusters to 10, execute:

```
# openstack --insecure federation project list
+----------------------------------+-----------+---------+-------------+
| ID                               | Domain ID | Enabled | Name        |
+----------------------------------+-----------+---------+-------------+
| d746acd8b2e847c4925685b8ad95b828 | b41c<...> | True    | testproject |
+----------------------------------+-----------+---------+-------------+
# openstack --insecure coe quotas create \
--project-id d746acd8b2e847c4925685b8ad95b828 --resource Cluster \
--hard-limit 10
+------------+----------------------------------+
| Property   | Value                            |
+------------+----------------------------------+
| resource   | Cluster                          |
| in_use     | 0                                |
| created_at | 2021-01-13T16:14:28.003861+00:00 |
| updated_at | -                                |
| id         | 3                                |
| project_id | d746acd8b2e847c4925685b8ad95b828 |
| hard_limit | 10                               |
+------------+----------------------------------+
```

- To limit the number of load balancers, use the `loadbalancer quota set` command with the `--loadbalancer` option. For example, to limit the number of load balancers to 20, execute:

```
# openstack --insecure loadbalancer quota set  testproject --loadbalancer 20
+----------------+-------+
| Field          | Value |
+----------------+-------+
| load_balancer  | 20    |
| listener       | -1    |
| pool           | -1    |
| health_monitor | -1    |
| member         | -1    |
+----------------+-------+
```

You can check the applied project quotas by executing the following command:

```
# openstack --insecure quota show
```

Changes you make to project quotas are also reflected in the admin panel.

Creating QoS policies as a domain administrator

With the quota manager role, you can also create network QoS policies. Do the following:

1. Use the environment file for the domain administrator to authorize further OpenStack commands:

```
# source domain-admin.sh
```

2. Use the project name variable for the project where you want to create a QoS policy. For example:

```
# export OS_PROJECT_NAME=testproject
```

3. Create a QoS policy. For example:

```
# openstack --insecure network qos policy create policy1
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| description |                                                 |
| id          | 3c9a40a2-e5b1-4d63-a577-48d9489eaf77            |
| is_default  | False                                           |
| location    | Munch({'project': Munch({'domain_name': test,   |
|             | 'domain_id': None, 'name': testproject,         |
|             | 'id': u'd746acd8b2e847c4925685b8ad95b828'}),    |
|             | 'cloud': '', 'region_name': '', 'zone': None}) |
| name        | policy1                                         |
| project_id  | d746acd8b2e847c4925685b8ad95b828                |
| rules       | []                                              |
| shared      | True                                            |
| tags        | []                                              |
+-------------+-------------------------------------------------+
```

For more information about network QoS policies, refer to "Using network QoS policies" (p. 357).

Changing TLS configuration for backup and object storage

To filter connections to the backup and object storage services, an administrator can configure allowed TLS protocol versions and ciphers.

Configuring TLS parameters for backup storage

TLS protocol versions for backup storage

To restrict the use of TLS 1.0 and 1.1 protocols, specify the appropriate value in the `advanced.min_tls_version` parameter in the `/etc/vstorage/abgw.config` file. The following values are available:

- 0: Allows 1.0, 1.1, and 1.2 TLS protocol versions
- 1: Allows 1.1 and 1.2 TLS protocol versions
- 2: Allows only 1.2 TLS protocol version

For example, to allow using all TLS protocol versions, specify 0 as follows:

```
advanced.min_tls_version = 0
```


TLS ciphers for backup storage

To accept connections to backup storage only with particular TLS ciphers, specify them in the `advanced.tls_ciphers` parameter in the `/etc/vstorage/abgw.config` file. If a client has none of the specified ciphers, the connection will fail. For the cipher format and full set, refer to the cipher list section in the ciphers manual page.

By default, the following ciphers are used:

- ECDHE-ECDSA-CHACHA20-POLY1305
- ECDHE-RSA-CHACHA20-POLY1305
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- ECDHE-RSA-AES256-SHA
- ECDHE-ECDSA-AES128-SHA
- ECDHE-RSA-AES128-SHA
- DHE-RSA-AES128-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA

Note the following:

- If you specify one cipher (for example, RSA-AES128) and it is not supported, the connection will fail.
- If you specify two ciphers (for example, CAMELIA and RSA-AES128) and only one of them is supported (for example, CAMELIA), the connection will be established based on the supported cipher (in this case, CAMELIA).
- If you specify an empty value, all connections will fail.

For example, to limit the allowed TLS ciphers only to ECDHE-ECDSA-CHACHA20-POLY1305 and ECDHE-RSA-CHACHA20-POLY1305, specify them separated by colons as follows:

```
advanced.tls_ciphers = ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305
```

Configuring TLS parameters for object storage

By default, only TLS protocol version 1.2 is accepted for connections to the S3 cluster. Moreover, only the following ciphers are allowed:

- ECDHE-ECDSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-SHA256
- DHE-RSA-AES128-SHA256
- AES128-GCM-SHA256

These options are automatically applied to all S3 clusters running Virtuozzo Hybrid Infrastructure 4.7, even if a cluster was created in an earlier version.

TLS protocol versions for object storage

To accept connections to object storage with TLS 1.0 and 1.1 protocols, do the following:

1. Specify the required TLS protocols, space-separated, in the `OSTOR_S3_GW_CUSTOM_SSL_PROTOCOLS` parameter in the configuration file `/usr/libexec/vstorage-ui-backend/etc/backend.cfg` on each management node. For more details on this parameter, refer to the nginx documentation. For example, to enable TLS 1.1 used in earlier versions, specify:

   ```
   OSTOR_S3_GW_CUSTOM_SSL_PROTOCOLS = 'TLSv1.1 TLSv1.2'
   ```

2. Restart the backend service:

   ```
   # systemctl restart vstorage-ui-backend
   ```

3. In the admin panel, go to the **Storage services** > **S3** > **Nodes** screen, click **Protocol settings**, and then click **Done** to apply your changes.

TLS ciphers for object storage

To accept customer ciphers in object storage, do the following:

1. Specify the required ciphers, separated by colons, in the `OSTOR_S3_GW_CUSTOM_SSL_CIPHERS` parameter in the configuration file `/usr/libexec/vstorage-ui-backend/etc/backend.cfg` on each management node. For more details on this parameter, refer to the nginx documentation. For example, to enable ciphers used in earlier versions, specify the following:

   ```
   OSTOR_S3_GW_CUSTOM_SSL_CIPHERS = 'HIGH:!3DES:!RC4:!aNULL:!MD5:!kEDH'
   ```

2. Restart the backend service:

```
# systemctl restart vstorage-ui-backend
```

3. In the admin panel, go to the **Storage services** > **S3** > **Nodes** screen, click **Protocol settings**, and then click **Done** to apply your changes.

Restricting outbound traffic from cluster nodes

To control outbound traffic from your cluster nodes, you can configure outbound firewall rules for public networks by using the `vinfra` tool. By default, ports used by system services are opened, to ensure non-disruptive cluster operation. Additionally, outbound traffic is always allowed in the subnet dedicated to internal communication between cluster nodes. As a private network is not publicly exposed and does not communicate with any external endpoints, you do not need to restrict outbound traffic for it. A network is recognized as private if it is assigned any of these traffic types:

- OSTOR private
- Backup (ABGW) private
- Internal management
- Storage

A private network always has the rule `<private_subnet_cidr>:any:0`, which allows all outbound traffic in the current subnet. This rule is not visible via the `vinfra` commands and exists only in `iptables`.

To block all outbound traffic except that which is necessary for cluster operation, perform the following steps:

1. Create additional firewall rules, to allow outbound traffic for particular services.
2. Remove the rule that allows all outbound traffic.
3. Check your network settings.

Default outbound firewall rules

All networks in the cluster have the default outbound allow rules, which are specified in the format: `<address>:<protocol>:<port>:<description>`. These rules are the following:

`0.0.0.0:tcp:8888:Admin panel`
: Used by the cluster API

`0.0.0.0:tcp:80:HTTP`
: Connection to the update repository and the S3 backend when configured to serve HTTP requests

`0.0.0.0:tcp:443:HTTPS`
: Communication with Acronis Cyber Cloud and the S3 services

`0.0.0.0:udp:53:DNS`
: DNS name resolution

`0.0.0.0:tcp:53:DNS`

DNS name resolution

`0.0.0.0:udp:123:NTP`

Time syncronization

`0.0.0.0:tcp:8443:ABGW registration`

Data control for the Acronis Cyber Protect agents and Management server

`0.0.0.0:tcp:44445:ABGW Geo-replication`

Backup data replication between clusters

`0.0.0.0:tcp:9877:Acronis Cyber Protect`

Registration with Acronis Cyber Protect Management server in on-premise installations

`0.0.0.0:tcp:5900-6079:VM VNC Legacy`

Legacy ports for VNC console access to virtual machines

`0.0.0.0:udp:4789:VXLAN`

Network traffic between virtual machines in private virtual networks

`0.0.0.0:tcp:15900-16900:VM VNC`

VNC console access to virtual machines in the compute cluster

`0.0.0.0:tcp:7050:KA license`

Connection to the Key Authentication (KA) licensing server

`0.0.0.0:tcp:5224:KA report`

Sending reports to the KA server

`0.0.0.0:any:0:Allow all`

Allows all outbound traffic

Creating outbound firewall rules

To create custom outbound firewall rules, use the "vinfra cluster network set" (p. 20) command with the `--add-outbound-allow-list` option. A rule should be specified in the format: `<address>:<protocol>:<port>:<description>`, where:

- `<address>` is a single IP address (10.10.10.10), address range (10.10.10.0-10.10.10.10), or subnet CIDR (10.10.10.0/32)
- `<protocol>` can be `udp`, `tcp`, or `any`
- `<port>` is an integer value (22) or a range (20-22)
- `<description>` usually contains the name of the service that uses the specified port

The cases when you need to create an additional rule are the following:

- If you connect a remote iSCSI device to your cluster node, manually add a rule specifying the port number used for connecting this iSCSI device. For example:


```
# vinfra cluster network set Public --add-outbound-allow-list
"0.0.0.0:tcp:3260:Remote iSCSI"
```

- If you plan to change the network configuration and IP address assignment of your cluster nodes by using network migration, manually add a rule specifying TCP and UDP ports 60000–60100. For example:

```
# vinfra cluster network set Public --add-outbound-allow-list \
"0.0.0.0:tcp:60000-60100:Network migration","0.0.0.0:udp:60000-60100:Network
migration"
```

- If you plan to reassign an exclusive traffic type from one network to another, manually add rules specifying TCP and UDP ports 60000–60100 for both networks. For example:

```
# vinfra cluster network set Public --add-outbound-allow-list \
"0.0.0.0:tcp:60000-60100:Network migration","0.0.0.0:udp:60000-60100:Network
migration"
# vinfra cluster network set MyNet --add-outbound-allow-list \
"0.0.0.0:tcp:60000-60100:Network migration","0.0.0.0:udp:60000-60100:Network
migration"
```

- If you enable user authentication in an NFS share with Kerberos V5, manually add rules specifying TCP ports 88 and 749, UDP port 88, and the Kerberos server IP address. For example, if the IP address of the Kerberos server is 10.128.168.20, run:

```
# vinfra cluster network set Public --add-outbound-allow-list \
"10.128.168.20:tcp:88:Kerberos","10.128.168.20:tcp:749:Kerberos",\
"10.128.168.20:udp:88:Kerberos"
```

- If you configure a custom port for a particular service, manually add a rule specifying the used port number. For example:

```
 # vinfra cluster network set Public --add-outbound-allow-list
"0.0.0.0:udp:161:Zabbix"
```

Removing outbound firewall rules

To remove the rule `0.0.0.0:any:0:Allow all`, which allows all outbound traffic, use the "vinfra cluster network set" (p. 20) command with the `--del-outbound-allow-list` option. For example:

```
# vinfra cluster network set Public --del-outbound-allow-list "0.0.0.0:any:0:Allow all"
```

In this case, all attempts to establish connections from the cluster to external endpoints will be blocked.

Note
When restricting outbound traffic, it is recommend to modify the default outbound rules to use specific IP addresses or subnets, according to your network infrastructure and security policies.

Listing outbound firewall rules

To check that all the required outbound allow rules apply to your network, use the "vinfra cluster network show" (p. 20) command. For example:

```
# vinfra cluster network show Public
+--------------------+-------------------------------------------+
| Field              | Value                                     |
+--------------------+-------------------------------------------+
| id                 | c2e799f5-c41d-4865-bcce-06b471affed6      |
| inbound_allow_list | []                                        |
| inbound_deny_list  | []                                        |
| name               | Public                                    |
| outbound_allow_list | - 0.0.0.0:tcp:8888:Internal management    |
|                    | - 0.0.0.0:tcp:80:HTTP                     |
|                    | - 0.0.0.0:tcp:443:HTTPS                   |
|                    | - 0.0.0.0:udp:53:DNS                      |
|                    | - 0.0.0.0:tcp:53:DNS                      |
|                    | - 0.0.0.0:udp:123:NTP                     |
|                    | - 0.0.0.0:tcp:8443:ABGW registration      |
|                    | - 0.0.0.0:tcp:44445:ABGW Geo-replication  |
|                    | - 0.0.0.0:tcp:9877:Acronis Cyber Protect  |
|                    | - 0.0.0.0:tcp:5900-6079:VM VNC Legacy     |
|                    | - 0.0.0.0:udp:4789:VXLAN                  |
|                    | - 0.0.0.0:tcp:15900-16900:VM VNC          |
| traffic_types      | Backup (ABGW) public,Compute API,iSCSI,NFS, |
|                    | S3 public,Self-service ...<truncated>     |
| vlan               | 0                                         |
+--------------------+-------------------------------------------+
```

Restoring default outbound firewall rules

Important
When resetting to defaults, your custom outbound firewall rules will be discarded.

To restore the default outbound firewall rules for a network, use the "vinfra cluster network set" (p. 20) command with the `--restore-default-outbound-allow-list` option. For example:

```
vinfra cluster network set Public --restore-default-outbound-allow-list
```

Defining object storage classes

You can use up to four object storage classes for applications with different performance and redundancy requirements. The first storage class is set automatically during the S3 cluster creation. The other three classes you can define manually by using the `ostor-ctl set-storage-class` command.

To set a storage class, do the following:

1. Obtain the password for your storage cluster. For example:

```
# vinfra cluster password show
+----------+---------+
| Field    | Value   |
+----------+---------+
| id       | 1       |
| name     | cluster |
| password | W3HMNq  |
+----------+---------+
```

2. Find out the ID of the object storage volume. For example:

```
# ostor-ctl get-config
<...>
VOL_ID             TYPE     STATE
0100000000000002   OBJ     READY
<...>
```

3. Define a storage class specifying its name, the number of object servers it will include, and the required redundancy settings. When prompted, enter the password obtained in step 1. For example, to create the storage class 1 with 2 object servers and the redundancy scheme of 2 replicas for tier 1, run:

```
# ostor-ctl set-storage-class -s /mnt/vstorage/vols/ostor/ -V 0100000000000002 \
-C 1 -O 2 --vstorage-attr "replicas=2:1 tier=1"
Please enter password for 'ostor-private.svc.vstoragedomain.':
Storage 1 class is successfully assigned to services
```

   This command requires the following parameters:

   - `-s`: the path to the object storage directory
   - `-V`: the volume ID obtained in step 2
   - `-C`: the storage class name (can be 1–3)
   - `-O`: the number of object servers to create
   - `--vstorage-attr`: redundancy settings, where you can specify the desired tier, failure domain, and data redundancy scheme (refer to the `vstorage set-attr` help message)
4. Check that the new storage class is set. For example, for the storage class 1, run:

```
# vstorage get-attr /mnt/vstorage/vols/ostor/0100000000000002/services/sc1/
connected to MDS#1
Path: 'vstorage://hciHeat/vols/ostor/0100000000000002/services/sc1'
Attributes:
  directory
  client-ssd-cache=1
  replicas=2:1
  failure-domain=host
  failure-domain.int=1
  tier=1
  chunk-size=268435456
```

To change redundancy settings of a storage class, use the `ostor-ctl cfg-storage` command. For example:

```
# ostor-ctl cfg-storage -r /mnt/vstorage/vols/ostor/0100000000000002/ -C 1 \
--vstorage-attr "replicas=3:2 tier=0"
```

Creating virtIO disks for virtual machines

To improve I/O performance of virtual machines, you can use virtIO disks with them. By default, virtual machines are created with disks attached to the SCSI bus, which cannot be changed later. You can create a volume for a VM and attach this volume to the virtIO bus during the VM deployment. To do this, use the `--volume bus=virtio` option of the "vinfra service compute server create" (p. 95) command. This option can be applied for creating boot volumes from both ISO and QCOW2 images. You can also use it to attach non-boot volumes. Note that you need to specify this option each time you create a virtual machine and you cannot attach more volumes after the VM is created. For example:

- To create a VM from the `mytemplate` QCOW2 image, run:

  ```
  # vinfra service compute server create myvm1 --network id=private,fixed-
  ip=192.168.128.100 --flavor medium\
  --volume source=image,id=mytemplate,bus=virtio,size=100
  ```

- To create a VM from the `myiso` ISO image, run:

  ```
  # vinfra service compute server create myvm2 --network id=private,fixed-
  ip=192.168.128.100 --flavor medium\
  --volume source=blank,size=100,bus=virtio,boot-index=0,type=disk \
  --volume source=image,id=myiso,size=5,boot-index=1,type=cdrom
  ```

Alternatively, it is possible to apply the virtIO bus property to an image via the OpenStack command-line tool. This property allows you to create multiple VMs from configured images in the command-line interface, as well as in the admin and self-service panels. However, it only works for QCOW2 images. For example, to configure the `mytemplate` QCOW2 image, do the following:

1. Connect to the OpenStack command-line interface as a system administrator to authorize further OpenStack commands (refer to "Connecting to OpenStack command-line interface" (p. 326)):

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Run the command:

```
# openstack --insecure image set mytemplate --property hw_disk_bus=virtio
```

After the VM is created, all volumes that you add to it will be attached to the virtIO bus.

Attaching physical PCI devices to virtual machines

By attaching various PCI devices from compute nodes to virtual machines, you can reduce VM network latency or accelerate visualization inside a guest operating system. PCI device passthrough is available only on servers that support Input/Output Memory Management Unit (IOMMU). For a list of IOMMU-supporting hardware, refer to this article.

You can attach the following PCI devices:

- physical GPU cards
- network adapters with Single Root I/O Virtualization (SR-IOV) capabilities[1]
- host bus adapters

To attach a physical PCI device to a virtual machine, follow these steps:

Note
Steps for HBA passthrough are similar to those for GPU passthrough.

1. Prepare compute nodes for PCI passthrough.
2. Reconfigure the compute cluster to enable PCI passthrough.
3. Create virtual machines with attached PCI devices.

Preparing nodes for PCI passthrough

1. For SR-IOV virtualization, check that the network adapter that you want to pass through can be virtualized:
   a. List all network adapters on a node and obtain their VID and PID:

```
# lspci -nn | grep Ethernet
00:03.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family
```

[1]The SR-IOV technology enables splitting a single physical adapter (physical function) into several virtual adapters (virtual functions). Each virtual function appears as a separate PCI device that can be attached to multiple virtual machines.


```
[ConnectX-5] [15b3:1017]
00:04.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family
[ConnectX-5] [15b3:1017]
```

`[15b3:1017]` is the VID and PID of the network adapter.

b. Check that the chosen network adapter supports SR-IOV by using its VID and PID:

```
# lspci -vv -d 15b3:1017 | grep SR-IOV
Capabilities: [180 v1] Single Root I/O Virtualization (SR-IOV)
Capabilities: [180 v1] Single Root I/O Virtualization (SR-IOV)
```

2. For GPU passthrough, detach the GPU card that you want to attach to VMs from the host.
   a. List all graphics cards on a node and obtain their VID and PID:

```
# lspci -nn | grep VGA
03:00.0 VGA compatible controller [0302]: NVIDIA Corporation TU104GL [Tesla T4]
[10de:1eb8] (rev a1)
03:00.0 VGA compatible controller [0302]: NVIDIA Corporation TU104GL [Tesla T4]
[10de:1eb8] (rev a1)
```

   `[10de:1eb8]` is the VID and PID of the graphics card.

   b. Run the `pci-helper.py` script to detach the chosen GPU specifying its VID and PID. For an NVIDIA graphics card, additionally blacklist the Nouveau driver. For example:

```
# /usr/libexec/vstorage-ui-agent/bin/pci-helper.py detach 10de:1eb8 --blacklist-
nouveau
```

3. Enable IOMMU on the node by running the `pci-helper.py` script:

```
# /usr/libexec/vstorage-ui-agent/bin/pci-helper.py enable-iommu
```

   The script works for both Intel and AMD processors.

4. Reboot the node to apply the changes.

You can check that IOMMU is successfully enabled in the `dmesg` output:

```
# dmesg | grep -e DMAR -e IOMMU
[    0.000000] DMAR: IOMMU enabled
```

For Mellanox network adapters, you need to additionally enable SR-IOV in firmware. Do the following:

1. Download Mellanox Firmware Tools (MFT) from the official website and extract the archive on the node. For example:

```
# wget https://www.mellanox.com/downloads/MFT/mft-4.17.0-106-x86_64-rpm.tgz
# tar -xvzf mft-4.17.0-106-x86_64-rpm.tgz
```

2. Install the package, and then start Mellanox Software Tools (MST):

```
# yum install rpm-build
# . mft-4.17.0-106-x86_64-rpm/install.sh
# mst start
```

3. Determine the MST device path:

```
# mst status
```

4. Query the current configuration:

```
# mlxconfig -d /dev/mst/mt4119_pciconf0 q
...
Configurations:
...
         NUM_OF_VFS            4               # Number of activated VFs
         SRIOV_EN              True(1)         # SR-IOV is enabled
...
```

5. Set the desired values, if necessary. For example, to increase the number of virtual functions to 8, run:

```
# mlxconfig -d /dev/mst/mt4119_pciconf0 set SRIOV_EN=1 NUM_OF_VFS=8
```

6. Reboot the node to apply the changes.

Configuring the compute cluster for PCI passthrough

To enable PCI passthrough for the compute cluster, you need to create a configuration file in the YAML format and use it to reconfigure the compute cluster.

Below is an example of `pci-passthrough.yaml`:

```
- node_id: c3b2321a-7c12-8456-42ce-8005ff937e12
  devices:
    - device_type: sriov
      device: enp2s0
      physical_network: sriovnet
      num_vfs: 8
    - device_type: generic
      device: 1b36:0100
      alias: gpu
```

In this configuration file:

- `node_id` is the identifier of a compute node that hosts PCI devices
- `device_type` can be `sriov` for a network adapter or `generic` for a graphics card
- `device` is the device name for a network adapter or the VID and PID for a graphics card
- `physical_network` is an arbitrary name that will be used as an alias for a network adapter
- `alias` is an arbitrary name that will be used as an alias for a graphics card

- `num_vfs` is the number of virtual functions to create for a PCI device with the `sriov` device type
  The maximum number of virtual functions supported by a PCI device is specified in the `/sys/class/net/<device_name>/device/sriov_totalvfs` file. For example:

  ```
  # cat /sys/class/net/enp2s0/device/sriov_totalvfs
  63
  ```

Now, you can pass this configuration file to the `vinfra service compute set` command. For example:

```
# vinfra service compute set --pci-passthrough-config pci-passthrough.yaml
+---------+--------------------------------------+
| Field   | Value                                |
+---------+--------------------------------------+
| task_id | 624916c6-617b-4687-b52c-b348e1f4892a |
+---------+--------------------------------------+
```

If the reconfiguration fails with the following error in `/var/log/vstorage-ui-backend/ansible.log`:

```
2021-09-23 16:42:59,796 p=32130 u=vstoradmin | fatal: [32c8461b-92ec-48c3-ae02-
4d12194acd02]: FAILED! => {"changed": true, "cmd": "echo 4 > /sys/class/net/
enp103s0f1/device/sriov_numvfs", "delta": "0:00:00.127417", "end": "2021-09-23
19:42:59.784281", "msg": "non-zero return code", "rc": 1, "start": "2021-09-23
19:42:59.656864", "stderr": "/bin/sh: line 0: echo: write error: Cannot allocate
memory", "stderr_lines": ["/bin/sh: line 0: echo: write error: Cannot allocate memory"],
"stdout": "", "stdout_lines": []}
```

In this case, run the the `pci-helper.py` script, and reboot the node:

```
# /usr/libexec/vstorage-ui-agent/bin/pci-helper.py enable-iommu --pci-realloc
# reboot
```

When the node is up again, repeat the `vinfra service compute set` command.

After the compute cluster is successfully reconfigured, you can proceed to creating virtual machines.

Creating virtual machines with physical GPU

1. Connect to the OpenStack command-line interface as a system administrator to authorize further OpenStack commands (refer to "Connecting to OpenStack command-line interface" (p. 326)).

   ```
   # kolla-ansible post-deploy
   # source /etc/kolla/admin-openrc.sh
   ```

2. Create a flavor with the `pci_passthrough` property specifying the GPU alias from the `pci-passthrough.yaml` file and the number of GPUs to use. For example, to create the `gpu-flavor` flavor with 8 vCPUs and 16 GiB of RAM, run:


```
# openstack --insecure flavor create --ram 16 --vcpus 8 --property "pci_
passthrough:alias"="gpu:1" gpu-flavor
```

3. Some drivers may require to hide the hypervisor signature. To do this, add the `hide_hypervisor_id` property to the flavor:

```
# openstack --insecure flavor set gpu-flavor --property hide_hypervisor_id=true
```

4. Create a virtual machine specifying the `gpu-flavor` flavor. For example, to create the `gpu-vm` from the `vol2` volume, run:

```
# openstack --insecure server create --volume vol2 --flavor gpu-flavor gpu-vm
```

Creating virtual machines with SR-IOV network ports

1. Connect to the OpenStack command-line interface as a system administrator to authorize further OpenStack commands (refer to "Connecting to OpenStack command-line interface" (p. 326)).

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Create a physical compute network specifying the network adapter alias from the `pci-passthrough.yaml` file. For example, to create the `sriov-net` network, run:

```
# openstack --insecure network create --provider-physical-network sriovnet --
provider-network-type flat sriov-net
```

3. Create a subnet for the `sriov-net` network disabling the built-in DHCP server and specifying the desired IP address range. For example, to create the `sriov-subnet` subnet with the 10.10.10.0/24 CIDR, run:

```
# openstack --insecure subnet create --no-dhcp --subnet-range 10.10.10.0/24 --network
sriov-net sriov-subnet
```

4. Create a network port in the `sriov-net` network with the direct PCI passthrough. For example, to create the `sriov-port` port with the IP address 10.10.10.10 from the `sriov-subnet` subnet, run:

```
# openstack --insecure port create --network sriov-net --vnic-type=direct --fixed-ip
subnet=sriov-subnet,ip-address=10.10.10.10 sriov-port
```

5. Create a virtual machine specifying the `sriov-port` port. Enable the `--config-drive` option, to automatically assign the IP address inside the guest operating system. For example, to create the `sriov-vm` from the `vol1` volume and with the `large` flavor, run:

```
# openstack --insecure server create --port sriov-port --volume vol1 --flavor large
sriov-vm --config-drive True
```

If the VM creation fails with the following error in `/var/log/hci/nova/nova-compute.log`:


```
2021-08-27 17:56:21.349 6 ERROR nova.compute.manager [instance: 9fb738bf-afe5-40ef-943c-
22e43696bfd9] libvirtError: internal error: qemu unexpectedly closed the monitor:
2021-08-27T14:56:20.294985Z qemu-kvm: -device vfio-pci,host=01:00.3,id=hostdev0,
bus=pci.0,addr=0x6: vfio error: 0000:01:00.3: group 1 is not viable
2021-08-27 17:56:21.349 6 ERROR nova.compute.manager [instance: 9fb738bf-afe5-40ef-943c-
22e43696bfd9] Please ensure all devices within the iommu_group are bound to their vfio
bus driver.
```

In this case, the physical and virtual functions of the network adapter might belong to the same IOMMU group. You can check this by using the `virsh nodedev-dumpxml` command and specifying the device names of physical and virtual functions. For example:

```
# virsh nodedev-dumpxml pci_0000_00_03_0 | grep iommuGroup
    <iommuGroup number='1'>
    </iommuGroup>
# virsh nodedev-dumpxml pci_0000_00_03_1 | grep iommuGroup
    <iommuGroup number='1'>
    </iommuGroup>
```

The device names have the format `pci_0000_<bus_number>_<device_number>_<function_number>`. These numbers can be obtained via the `lspci` command:

```
# lspci -nn | grep Ethernet
00:03.0 Ethernet controller [0200]: Mellanox Technologies MT27800 Family [ConnectX-5]
[15b3:1017]
...
```

In this output, `00` is the bus number, `03` is the device number, and `0` is the function number.

If the physical and virtual functions belong to the same IOMMU group, you need to detach the physical function from the node by running the `pci-helper.py` script and specifying its VID and PID. For example:

```
# /usr/libexec/vstorage-ui-agent/bin/pci-helper.py detach 15b3:1017
```

Configuring multiple subnets in compute networks

If you exhaust all public IP addresses in a physical compute network, you can add more subnets to this network by using the OpenStack command-line tool. The new subnets will be available in the admin and self-service panel for IP address allocation and management.

To create an additional subnet for a compute network, do the following:

1. Connect to the OpenStack command-line interface as a system administrator to authorize further OpenStack commands (refer to "Connecting to OpenStack command-line interface" (p. 326)).


```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Identify the required network by listing all of the existing networks:

```
# openstack --insecure network list
+----------------------+---------+--------------------------------------+
| ID                   | Name    | Subnets                              |
+----------------------+---------+--------------------------------------+
| a1d8d6ae-c89d-4307<…> | public  | d52aa9f4-6a4b-4268-a71d-1a50f9b60aa9 |
| e31eac69-9ab7-41ad<…> | private | 470526d3-ea5a-48fb-81ac-20273f005f61 |
+----------------------+---------+--------------------------------------+
```

3. Create a new subnet in the network by using the `openstack subnet create` command. For example:

```
# openstack --insecure subnet create --ip-version 4 --subnet-range 10.164.132.0/24 \
--gateway 10.164.132.1 --dhcp --dns-nameserver 8.8.8.8 --allocation-pool \
start=10.164.132.201,end=10.164.132.212 --network public newsubnet
```

Creating virtual machines with UEFI boot

You can create virtual machines with UEFI boot from either templates (QCOW2 images) or ISO images.

Creating a virtual machine with UEFI boot from a template

Important
To create a VM with UEFI boot from a template, your QCOW2 image must have UEFI boot already enabled.

1. Connect to the OpenStack command-line interface to authorize further `openstack` commands (refer to "Connecting to OpenStack command-line interface" (p. 326)):

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Find out the ID of the required template with UEFI. For example:

```
# openstack --insecure image list | grep centos.qcow2
| 007db63f-9b41-4918-b572-2c5eef4c8f4b | centos7.qcow2 | active |
```

3. Specify the UEFI firmware for this template by using the `hw_firmware_type` property. For example:

```
# openstack --insecure image set --property hw_firmware_type=uefi 007db63f-9b41-4918-
b572-2c5eef4c8f4b
```

4. Create a VM from this template in the command-line or user interface.

Creating a virtual machine with UEFI boot from an ISO image

1. Connect to the OpenStack command-line interface to authorize further `openstack` commands (refer to "Connecting to OpenStack command-line interface" (p. 326)):

```
# kolla-ansible post-deploy
# source /etc/kolla/admin-openrc.sh
```

2. Find out the ID of the required ISO image. For example:

```
# openstack --insecure image list | grep centos.iso
| c9d6f6e9-9c6d-4d1c-824f-c3542f70fdb0 | centos7.iso | active |
```

3. Specify the UEFI firmware for this ISO image by using the `hw_firmware_type` property. For example:

```
# openstack --insecure image set --property hw_firmware_type=uefi c9d6f6e9-9c6d-4d1c-
824f-c3542f70fdb0
```

4. Create a VM from this ISO image in the command-line or user interface.
5. When the VM is launched, shut it down, and then specify the UEFI firmware for this VM's boot volume by using the `hw_firmware_type` property. For example, if the boot volume ID is 12d360f4-afe8-48c9-af24-7f048dcec0c9, run:

```
# openstack --insecure volume set --image-property hw_firmware_type=uefi 12d360f4-
afe8-48c9-af24-7f048dcec0c9
```

6. Start the VM and continue the guest OS installation.

Viewing cluster logs

When you encounter a problem in Virtuozzo Hybrid Infrastructure, you can send a problem report, as described in "Getting technical support" in the Administrator Guide. The report will gather all the logs needed to troubleshoot the problem and send them to the technical support team.

Alternatively, you can investigate the root cause of your problem by using the logs listed in table below.

Cluster log location

Service	Log location	Description
Metadata	`/vstorage/mds/logs/mds.log.zst` on the storage node hosting an MDS service	Storage metadata service events

Service	Log location	Description
Storage	`/vstorage/<id>/cs/logs/cs.log.zst` on the storage node hosting a CS service **Note** To find out the log location of a particular CS on the node, run `vstorage -c <cluster_name> list-services -C`.	Chunk service events
Storage mountpoint	`/var/log/vstorage/<cluster_name>/vstorage-mount.*.blog` on any storage node	Software-defined storage mounting on each node
Management node	`/var/log/vstorage-ui-backend/messages.log` and `/var/log/vstorage-ui-backend/celery*.log` on the management node	Management node and admin panel events
	`/var/log/vstorage-ui-agent/*` on any storage node	Agent controller component events
Backup Gateway	`/var/log/vstorage/abgw.log*zst` on any node in the Backup Gateway cluster **Note** The latest log is `abgw.log.zst`, older ones are renamed to `abgw.log.0.zst`, `abgw.log.1.zst`, etc.	Backup Gateway cluster deployment and management
iSCSI	`/var/log/vstorage/iscsi/vstorage-target.log` on any node in an iSCSI target group	iSCSI target management
	`/var/log/vstorage/iscsi/vstorage-target-monitor.log` on any node in an iSCSI target group	iSCSI target monitoring
	`/var/log/vstorage/iscsi/scst.log.zst` on any node in an iSCSI target group	SCST service logs
S3	`/var/log/ostor/NS-*` on the S3 node with NS services	S3 name server events
	`/var/log/ostor/OS-*` on the S3 node with OS services	S3 object server events
	`/var/log/ostor/S3GW-*` on the S3 node with GW services	S3 gateway events
	`/var/log/nginx/*` on any node in the S3 cluster	`nginx` service logs
	`/var/log/ostor/GR-*` on the S3 node with GR services	S3 geo-replicator service events
NFS	`/var/log/ganesha/ganesha.log` and `/var/log/ostor/ostorfs.log.gz` on any node in the NFS cluster	NFS server events
	`/var/log/vstorage/vstorage-nfsd.log` on any node in the NFS cluster	NFS service events

Service	Log location	Description
	`/var/log/ostor/FS-*` on the node hosting an NFS share	FS service events
	`/var/log/ostor/OS-*` on the node hosting an NFS share	OS service events
Compute	`/var/log/vstorage-ui-backend/ansible.log` on the controller node	Compute cluster and add-on deployment
	`/var/log/hci/beholder/beholder.log` on the controller node	Notifications about all compute events, including VM placement
	`/var/log/hci/nova/*` on the compute node hosting a VM **Note** In case of problems during VM migration, view `/var/log/hci/nova/nova-compute.log` on the source and destination compute nodes.	Virtual machine management
	`/var/log/hci/neutron/neutron-l3-agent.log` on any compute node	Virtual routing events
	`/var/log/hci/neutron/neutron-openvswitch-agent.log` on the compute node hosting a VM	VM network interface management
	`/var/log/hci/cinder/*` on the controller node	Compute volume management
	`/var/log/hci/glance/glance-api.log` on the controller node	Image service API requests
	`/var/log/hci/octavia/octavia-worker.log` and `/var/log/hci/octavia/octavia-api.log` on the controller node	Load balancing service management
	`/var/log/hci/magnum/magnum-conductor.log`, `/var/log/hci/magnum/magnum-api.log`, and `/var/log/hci/heat/heat-engine.log` on the controller node	Kubernetes service and VM stack deployment and management
	`/var/log/hci/gnocchi/*` and `/var/log/hci/ceilometer/*` on any compute node	Billing metering service management
High availability	`/var/log/vstorage-ui-backend/ha.log` on all management nodes	High availability management
Updates	`/var/log/vstorage-ui-backend/software-updates.log` on the management node	Software update orchestration
	`/var/log/vstorage-ui-agent/software-updates.log` on any storage node	Software update downloading and installation on each node

To open log files, use the following commands:

- for LOG files:

```
# less <log_file>.log
```

- for BLOG files:

```
# blogcat <log_file>.blog | less
```

- for GZ files:

```
# zless <log_file>.gz
```

- for ZST files:

```
# zstdless <log_file>.zst
```