Virtuozzo 7

User's Guide

August 31, 2018

Virtuozzo International GmbH

Vordergasse 59

8200 Schaffhausen

Switzerland

Tel: + 41 52 632 0411

Fax: + 41 52 672 2010

`https://virtuozzo.com`

Contents

CHAPTER 1

Learning Virtuozzo Basics

This chapter provides a brief description of Virtuozzo, virtual machines and containers, their specifications, and underlying technologies.

1.1 Virtuozzo Overview

Virtuozzo is a bare-metal virtualization solution that includes container virtualization, KVM-based virtual machines, software-defined storage along with enterprise features and production support. It runs on top of Virtuozzo Linux, a RHEL-based Linux distribution.

Virtuozzo provides the best value for cost-conscious organizations enabling them to:

- standardize server hardware platforms,
- effectively consolidate server resources,
- consolidate and support legacy operating systems and applications,
- streamline server and application deployment, maintenance, and management,
- simplify software testing and development,
- optimize server and application availability.

1.2 Differences between Virtuozzo and OpenVZ

OpenVZ is a free, open-source virtualization solution available under GNU GPL. OpenVZ is the base for Virtuozzo, the commercial solution that builds on OpenVZ and offers additional benefits to customers.

Compared to OpenVZ, Virtuozzo has the following extra features and differences:

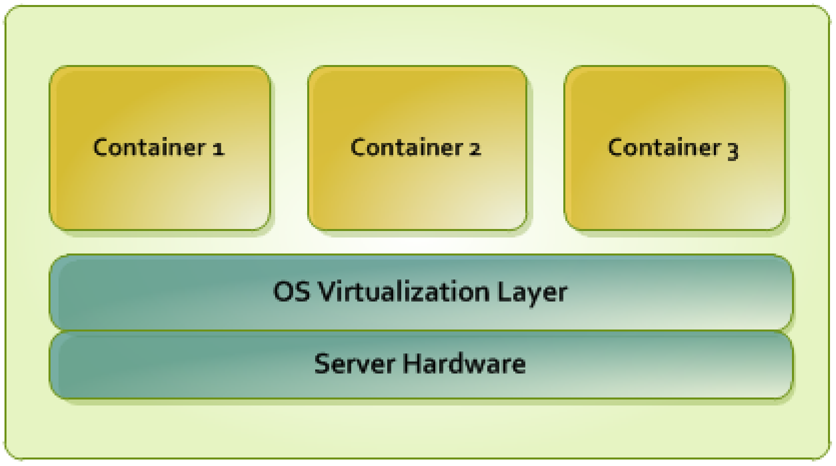
- container and virtual machine backups,
- software-defined storage,
- ReadyKernel,
- additional memory policies for vcmmd,
- technical support,
- different installer,
- different default package set.

1.3 OS Virtualization Layer

This section provides detailed information on the OS virtualization layer responsible for providing support for Virtuozzo containers.

1.3.1 Basics of OS Virtualization

The OS virtualization allows you to virtualize physical servers on the operating system (kernel) layer. The diagram below shows the basic architecture of OS virtualization.



The OS virtualization layer ensures isolation and security of resources between different containers. The virtualization layer makes each container appear as a standalone server. Finally, the container itself houses its own applications and workload. OS virtualization is streamlined for the best performance, management, and efficiency. Its main advantages are the following:

- Containers perform at levels consistent with native servers. Containers have no virtualized hardware and use native hardware and software drivers.
- Each container can seamlessly scale up to the resources of an entire physical server.
- OS virtualization technology provides the highest density available from a virtualization solution. You can create and run hundreds of containers on a standard production physical server.
- Containers use a single OS, making it extremely simple to maintain and update across containers. Applications may also be deployed as a single instance.

1.3.2 Virtuozzo Containers

From the point of view of applications and container users, each container is an independent system. This independence is provided by the Virtuozzo OS virtualization layer. Note that only a negligible part of the CPU resources is spent on virtualization. The main features of the virtualization layer implemented in Virtuozzo are the following:

- A container looks like a normal Linux system. It has standard startup scripts; software from vendors can run inside containers without any modifications or adjustment.
- A user can change any configuration file and install additional software inside containers.
- Containers are fully isolated from each other (file system, processes, sysctl variables).

- Containers share dynamic libraries, which greatly saves memory.
- Processes belonging to a container are scheduled for execution on all available CPUs. Consequently, containers are not bound to only one CPU and can use all available CPU power.

The two key parts of any container are the contents and configuration. By default, all container files are stored in the `/vz/private/<UUID>` directory on the hardware node, also called private area.

File Name	Description
`/vz/private/<UUID>`	Container private area.
`/vz/private/<UUID>/root.hdd/root.hdd`	Virtual hard disk with container contents. The maximum size of the virtual hard disk is 50 TB.
`/vz/root/<UUID>`	Container mount point.
`ve.conf`	Container configuration file: • Is symlinked to `/etc/vz/conf/<UUID>.conf` • Defines container parameters, such as allocated resource limits, IP address and hostname, and so on. • Overrides matching parameters in the global configuration file.

All container files are stored in a single image (`/vz/private/<UUID>/root.hdd/root.hdd`), similar to a virtual machine's hard disk. Such standalone nature:

- Enables easier migrations and backups due to a faster sequential I/O access to container images than to separate container files.
- Removes the need for OS and application templates once a container is created.
- Allows the use of native Linux disk quotas that are journaled and does not require quota recalculation after disasters like server crashes.

Note: Using containers that store all files in an image file (also known as containers with the container-in-an-image-file layout) is supported only for `/vz` partitions formatted as ext4.

1.3.2.1 Virtuozzo Container Hardware

A container may have the following virtual hardware:

Hardware	Theoretical	Certified
CPU	Up to the total number of threads on the host	Up to 64
RAM	Up to total amount of physical RAM on the host	Up to 1 TB
Disk drives	Up to 15: hard disk drives mapped to QCOW2 image files and DVD drives mapped to ISO image files, up to 50 TB each	
Network Interfaces	Up to 15	

1.3.3 Memory and IOPS Deduplication

Virtuozzo provides memory and IOPS deduplication that helps save memory and IOPS on the server and increases the maximum number of running containers per server.

Deduplication is provided by Virtuozzo File Cache which includes the `pfcached` daemon and a ploop image mounted to a directory on the server. The file cache ploop contains copies of eligible files located inside containers. To be eligible for caching, files in containers must meet certain configurable requirements, e.g., be read in a certain number of containers, be of certain size, be stored in certain directories in containers.

When the kernel gets a request to read a file located in a container ploop, it searches the file cache ploop for a copy of that file by the SHA1 hash stored as file's extended attribute. If successful, the copy in the file cache ploop is read instead of the original file in the container ploop. Otherwise, the original file in the container ploop is read.

To populate the file cache ploop with most requested files, `pfcached` periodically obtains container files read statistics from kernel, analyzes it, and copies files which are eligible to the file cache ploop. If the file cache ploop is running out of space, the least recently used files are removed from it.

Virtuozzo File Cache offers the following benefits:

- Memory deduplication. Only a single file from the file cache ploop needs to be loaded to memory instead of loading multiple identical files located in multiple containers.
- IOPS deduplication. Only a single file from the file cache ploop needs to be read instead of reading multiple identical files located in multiple containers.

If the physical server has storage drives of various performance, e.g., IDE and SSD, the file cache ploop performs better if located on the fastest storage drive on the node, e.g., SSD. In any case:

- If the server memory is not overcommitted, the file cache mostly helps speed up container start during which most files are read. In this case caches residing in memory are not cleaned often, so copies in the file cache ploop, once read during container start, do not need to be reread often during container operation.
- If the server memory is overcommitted, Virtuozzo File Cache helps speed up both container start and operation. In this case, caches residing in memory may be cleaned often, so files in the file cache ploop need to be reread as often.

Virtuozzo File Cache can be managed with the `pfcache` utility.

1.3.4 Templates

A template (or a package set) in Virtuozzo is a set of original application files repackaged for use by Virtuozzo. Usually, it is just a set of RPM packages for Red Hat like systems. Virtuozzo provides tools for creating templates, installing, upgrading, adding them to and removing them from a container.

Using templates lets you:

- Share RAM among similar applications running in different containers to save hundreds of megabytes of memory.
- Deploy applications simultaneously in many containers.
- Use different versions of an application in different containers (for example, perform upgrades only in certain containers).

There are two types of templates: OS and application.

- An OS template is an operating system and the standard set of applications to be found right after the installation. Virtuozzo uses OS templates to create new containers with a preinstalled operating system.
- An application template is a set of repackaged software packages optionally accompanied with configuration scripts. Application templates are used to add extra software to existing containers.

For example, you can create a container on the basis of the `redhat` OS template and add the MySQL application to it with the help of the `mysql` template.

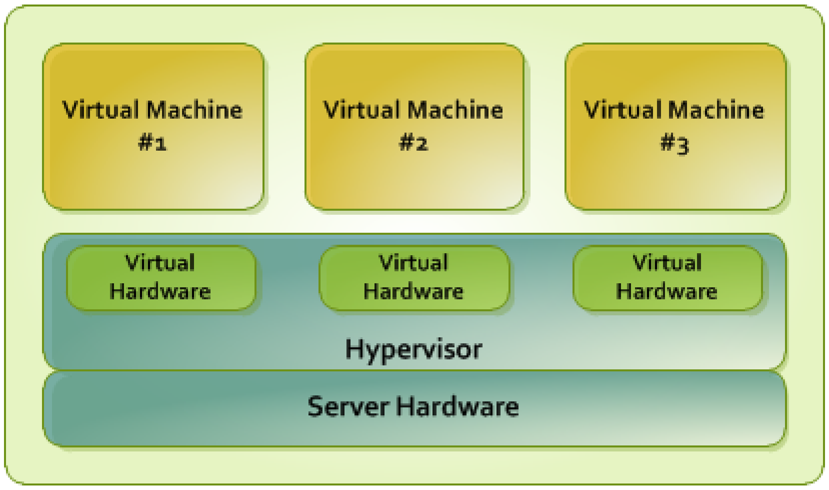
1.4 Hardware Virtualization Layer

This section familiarizes you with the second component of Virtuozzo—the hardware virtualization layer. This layer provides the necessary environment for creating and managing virtual machines.

1.4.1 Hardware Virtualization Basics

Virtuozzo is based on the concept of hardware virtualization. Hardware virtualization has a base layer—a hypervisor. This layer is loaded directly on the bare server and acts as an intermediary between the server hardware and virtual machines. To allocate hardware and resources to virtual machines, Virtuozzo virtualizes all hardware on the server. Once virtualized, hardware and resources can be easily assigned to virtual machines. With its virtual hardware, a virtual machine runs its own complete copies of an operating system and applications.

The following diagram shows the basic architecture of hardware virtualization.



Specifically, Virtuozzo uses the KVM/QEMU hypervisor and manages virtual machines via the libvirt API.

Hardware virtualization enables you to:

- Create multiple virtual machines with different operating systems on a single physical host.
- Run multiple guest operating systems and their applications simultaneously on a single physical host without rebooting.
- Consolidate and virtualize the computing environment, reduce hardware costs, lower operating expenses, and increase productivity.

- Use open APIs and SDK for integration with in-house and third-party applications.

1.4.2 Virtuozzo Virtual Machines

From the standpoint of applications and virtual machine users, each virtual machine (VM) is an independent system with an independent set of virtual hardware. This independence is provided by the Virtuozzo hardware virtualization layer. The main features of the virtualization layer are the following:

- A virtual machine resembles and works like a regular computer. It has its own virtual hardware. Software applications can run in virtual machines without any modifications or adjustment.
- Virtual machine configuration can be changed easily (e.g., adding new virtual disks or increasing RAM).
- Virtual machines are fully isolated from each other (file system, processes, sysctl variables) and the Virtuozzo host.
- A virtual machine can run any supported guest operating system. The guest OS and its applications are isolated inside a virtual machine and share physical hardware resources with other virtual machines.

1.4.2.1 Intel Virtualization Technology Support

Virtuozzo provides support for Intel virtualization technologies comprising a set of processor enhancements and improving the work of virtualization solutions. Utilizing these technologies, Virtuozzo can offload some workload to the system hardware, which results in the “near native” performance of guest operating systems.

1.4.3 Virtual Machine Hardware

A Virtuozzo virtual machine works like a usual standalone computer.

By default, virtual machines are created with the following virtual hardware:

- 1 VirtIO SCSI HDD, expanding,
- 1 CD-ROM (IDE for Windows and Debian guests, VirtIO SCSI for Linux guests except Debian),
- 1 VirtIO network adapter, bridged,
- 32MB video card.

Other hardware added to a default VM may depend on the chosen distribution (see *Creating Virtual Machines* on page 15).

The complete range of virtual hardware a virtual machine can have is provided in the table below.

CPU	Up to 64
RAM	Up to 1 TB
Video Adapter	VGA/SVGA video adapter with VBE 3.0
Video RAM	Up to 256 MB of video memory
Floppy Disk Drive	1.44 MB floppy disk drive mapped to an image file
IDE Devices	Up to 4 IDE devices: • hard disk drives mapped to QCOW2 image files (up to 16 TB each) • DVD drives mapped to ISO image files
SCSI Devices	Up to 15 SCSI devices: • hard disk drives mapped to QCOW2 image files (up to 16 TB each) • DVD drives mapped to ISO image files
VirtIO Devices	Up to 15 VirtIO hard disk drives mapped to QCOW2 image files (up to 16 TB each)
Network Interfaces	Up to 15 VirtIO (default), Intel 82545EM, or Realtek RTL8029 virtual network adapters.
Serial (COM) Ports	Up to 4 serial (COM) ports mapped to a socket, a real port, or an output file
Keyboard	Generic USB or PS/2 keyboard
Mouse	Generic USB or PS/2 wheel mouse

1.4.4 Virtual Machine Files

A virtual machine has at least two files: a configuration file (PVS file) and a hard disk image file (HDD file). It can also have additional files: a file for each additional virtual hard disk and output files for virtual ports. By default, the virtual machines files are stored in the `/vz/vmprivate/<UUID>` directory on the Virtuozzo server.

The list of files related to a virtual machine is given in the table below:

File Name	Description
`.pvs`	Virtual machine configuration file. It defines the hardware and resources configuration of the virtual machine. The configuration file is automatically generated during the virtual machine creation.
`.sav`	Dump file created when you suspend the virtual machine. This file contains the state of the virtual machine and its applications at the moment the suspend was invoked.
`.mem`	Memory dump file for the suspended virtual machine. For a running virtual machine, it is a temporary virtual memory file.
`.hdd`	Hard disk image in QCOW2 format. When you create a virtual machine, you can create it with a new virtual hard disk or use an existing one. A virtual machine can have multiple hard disks.
`.iso`	CD/DVD disc image. Virtual machines treat ISO images as real CD/DVD discs.
`.txt`	Output files for serial ports. The output `.txt` files are generated when a serial port connected to an output file is added to the virtual machine configuration.

1.4.5 Support of Virtual and Real Media

This section lists the types of disks that can be used by Virtuozzo virtual machines and provides the information about basic operations you can perform on these disks.

1.4.5.1 Supported Types of Hard Disks

Virtuozzo virtual machines can use only virtual hard disks image files as their hard disks.

1.4.5.2 Virtual Hard Disks

The capacity of a virtual hard disk can be set from 100 MB to 16 TB.

Virtuozzo uses expanding virtual hard disks. The image file of such a disk is initially small in size (smaller than the set virtual disk size) and grows as data is added to the disk in the guest OS.

1.4.5.3 CD/DVD Disc Images

Virtuozzo can use only CD/DVD disc images that are supported by the guest OS.

1.5 Virtuozzo Configuration

Virtuozzo allows you to configure settings for the physical server in general and for each container in particular. Among these settings are disk and user quotas, network parameters, default file locations, sample configuration files, and other.

Virtuozzo stores all OS virtualization-related configuration information in the global configuration file `/etc/vz/vz.conf`. It defines container parameters like the default OS templates, disk quotas, logging, and so on.

The configuration file is read when the Virtuozzo software and/or containers are started. However, many settings can also be changed on the fly by means of Virtuozzo standard utilities like `prlctl`, with or without modifying the corresponding configuration file to keep the changes for the future.

1.6 Resource Management

Virtuozzo resource management controls the amount of resources available to virtual machines and containers. The controlled resources include such parameters as CPU power, disk space, a set of memory-related parameters. Resource management allows you to:

- effectively share available physical server resources among virtual machines and containers
- guarantee Quality-of-Service in accordance with a service level agreement (SLA)
- provide performance and resource isolation and protect from denial-of-service attacks
- simultaneously assign and control resources for a number of virtual machines and containers
- collect usage information for system health monitoring

Resource management is much more important for Virtuozzo than for a standalone server since server resource utilization in such a system is considerably higher than that in a typical system.

1.7 Understanding Licensing

To start using Virtuozzo, you need a Virtuozzo license. You must install this license on your server after or during Virtuozzo installation. Every physical server hosting virtual machines and containers must have its own license. Licenses are issued by Virtuozzo and define a number of parameters in respect of your physical server. The main licensed parameters are listed below:

- The number of physical CPUs which can be installed on the physical server. That is, a dual core or hyperthreading processor is regarded as one CPU.
- The license expiration date. A license can be time-limited or permanent. Virtuozzo licenses have a start date, and if they are time-limited, can also have an expiration date specified in them. You must set up your system clock correctly. Otherwise, the license validation may fail.
- The number of virtual machines and containers that can simultaneously run on the physical server.
- The platform and architecture with which Virtuozzo is compatible.

For instructions on how to install, update, view, and transfer licenses, see *Managing Licenses* on page 133.

1.8 Physical Server Availability Considerations

The availability of a physical server running Virtuozzo is more critical than the availability of a typical PC server. Since it runs multiple virtual machines and containers providing a number of critical services, physical server outage might be very costly. It can be as disastrous as the simultaneous outage of a number of servers running critical services.

To increase physical server availability, we suggest that you follow the recommendations below:

- Use a RAID storage for critical virtual machines and containers. Do prefer hardware RAIDs, but software mirroring RAIDs might suit too as a last resort.
- Do not run any software on the server itself. Create special virtual machines and containers where you can host necessary services such as BIND, FTPD, HTTPD, and so on. On the server, you need only the SSH daemon. Preferably, it should accept connections from a pre-defined set of IP addresses only.
- Do not create users on the server itself. You can create as many users as you need in any virtual machine and container. Remember: compromising the server means compromising all virtual machines and containers as well.

CHAPTER 2

Managing Virtual Machines and Containers

This chapter describes how to perform day-to-day operations on virtual machines and containers.

2.1 Creating Virtual Machines and Containers

This section explains how to create new Virtuozzo virtual machines and containers using the `prlctl create` command. The options you should pass to this command differ depending on whether you want to create a virtual machine or container.

2.1.1 Choosing OS EZ Templates for Containers

Before creating a container, you need to choose an OS EZ template it will be based on.

2.1.1.1 Listing OS EZ Templates

To find out which OS EZ templates are already installed on the hardware node and cached (i.e. ready to be used), you can use the `vzpkg list` command. For example:

```
# vzpkg list -O
centos-6-x86_64                    2012-05-10 13:16:43
```

The timestamp next to an OS EZ template indicates when the template was cached.

Adding the `-O` option to the `vzpkg list` command, you can list only those OS EZ templates which are installed but not cached. You can also add the `--with-summary` option to display brief template descriptions:

```
# vzpkg list -O --with-summary
centos-6-x86_64                    :CentOS 6 (for Intel EM64T) EZ OS Template
```

2.1.1.2 Installing and Caching OS EZ Templates

Some of the supported OS EZ templates may not be preinstalled, so you may need to perform additional steps before you can create containers based on these templates. To list templates available for installation in the official remote repositories, run the following command:

```
# vzpkg list --available
fedora-23-x86_64                   factory
sles-11-x86_64                     factory
sles-12-x86_64                     factory
suse-42.1-x86_64                   factory
suse-42.2-x86_64                   factory
vzlinux-6-x86_64                   factory
```

To prepare a template for container creation, do the following:

1. Install the template package. For example:

   ```
   # vzpkg install template sles-11-x86_64
   ```

2. Configure additional template parameters if needed. Some of the EZ templates may require specific preparation steps that depend on the operating system. To prepare, for example, a SLES 11 template for container creation, you additionally need to set the `$RCE` parameter and SUSE repository credentials (obtained from the SUSE Customer Center) in the `/etc/vztt/url.map` file:

   ```
   $RCE            %24RCE
   $SLES11_PASS    <password>
   $SLES11_USER    <user>
   ```

3. Create the template cache:

   ```
   # vzpkg create cache sles-11-x86_64
   ```

Now you can create containers based on the prepared template.

2.1.2 Creating Containers

To create a container, use the `prlctl create` command as follows:

```
# prlctl create MyCT --vmtype ct
```

Virtuozzo will create a new container with the name `MyCT` using the default parameters from the global configuration file `/etc/vz/vz.conf`.

If you want to create a container with a guest OS different from the default specified in the global configuration file, add the `--ostemplate` option after the `prlctl create` command. For example:

```
# prlctl create MyCT --vmtype ct --ostemplate centos-6-x86_64
```

All container contents will be stored in this container's private area. To find out where the private area is located, use the `prlctl list` command as follows:

```
# prlctl list MyCT -i | grep "Home"
Home: /vz/private/26bc47f6-353f-444b-bc35-b634a88dbbcc
```

Note:

1. The first time you install an operating system in a container, its cache is created. To create a cache, you need to have an active Internet connection to access repositories that contain packages for the respective operating system. You can also set up a local package repository and use this repository to provide packages for your operating system. A local package repository is also required for some commercial distributions (e.g., for Red Hat Enterprise Linux).
2. For information on creating containers with preinstalled applications, see *Using Golden Image Functionality* on page 172.

2.1.3 Creating Virtual Machines

Creating a new virtual machine means creating a VM configuration based on a distribution you specified. To create VMs, use the `prlctl create` command. For example:

```
# prlctl create MyVM --distribution centos7 --vmtype vm
```

This command creates a configuration for a virtual machine `MyVM`, adjusts it for running the CentOS 7 guest OS, and places all the required files in the `/vz/vmprivate/<UUID>` directory.

Once the virtual machine configuration is ready, you will need to install a supported guest OS in it (e.g., via VNC as described in *Enabling VNC Access to Virtual Machines* on page 182).

Note: Before installing Windows Server 2012 (not R2) in a virtual machine, you need to manually mount the `floppy_win8.vfd` image file with the needed drivers by running the `prlctl set --device set ffd0 --image <path_to_file>` command. In Virtuozzo 7.0.4 (Update 4), the image is located in the `/usr/share/vz-guest-tools` directory, so the command is `prlctl set --device set ffd0 --image /usr/share/vz-guest-tools/floppy_win8.vfd`.

When choosing a distribution to install, have in mind that Virtuozzo supports VM guest initialization via cloud-init, so you can perform some of the initial configuration tasks on stopped virtual machines. To be able to use this feature, you can install a "cloud-enabled" distribution instead of a regular one. For more information, see *Using cloud-init for Virtual Machine Guest Initialization* on page 18.

2.1.4 Supported Guest Operating Systems

The following guest operating systems have been tested and are supported in virtual machines and containers.

2.1.4.1 Virtual Machines

- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2 with Service Pack 1
- CentOS 7.x (x64)
- CentOS 6.x (x64)
- Debian 9.x (x64)
- Debian 8.x (x64)
- Ubuntu 18.04 (x64)
- Ubuntu 17.10 (x64)
- Ubuntu 16.04 LTS (x64)
- Ubuntu 14.04 LTS (x64)

- Virtuozzo Linux 7.x (x64)
- CloudLinux 7.x (x64)
- CloudLinux 6.x (x64)

2.1.4.2 Containers

- CentOS 7.x (x64)
- CentOS 6.x (x64)
- Debian 9.x (x64)
- Debian 8.x (x64)
- Ubuntu 18.04 (x64)
- Ubuntu 17.10 (x64)
- Ubuntu 16.04 LTS (x64)
- Ubuntu 14.04 LTS (x64)
- Virtuozzo Linux 7.x (x64)
- Virtuozzo Linux 6.x (x64)
- openSUSE 42.x (x64)
- SUSE Linux Enterprise Server 12 Service Pack 1 (x64)
- SUSE Linux Enterprise Server 11 Service Pack 1, 2, 3, 4 (x64)

2.2 Performing Initial Configuration of Virtual Machines and Containers

Before you start using a newly created virtual machine or container, you will need to configure it. This section describes the main configuration steps.

2.2.1 Using cloud-init for Virtual Machine Guest Initialization

Virtuozzo supports VM guest initialization via cloud-init, so you can perform some of the initial configuration tasks described further in this section on stopped virtual machines. The supported tasks are: installing guest tools, setting user names and passwords, and configuring network settings.

The changes resulting from performing the above tasks are not applied to the VM immediately but rather saved as instructions to be carried out when the guest OS with cloud-init is loading. So when you run a corresponding command (e.g., `prlctl set --userpasswd`), the following happens: the bundled image with cloud-init instructions is copied to the VM home path, a CD-ROM device is added to the VM, and the image is mounted to said CD-ROM. However, the changes (e.g., to the user name and password) will only be applied after you install and start loading the guest OS.

As mentioned above, you will need cloud-init installed in a guest OS for the feature to work. For Linux guests, the easiest way to get cloud-init is to install a "cloud-enabled" distribution that already comes with it. You can also install cloud-init manually (e.g., by running `yum install cloud-init` on CentOS 7). For Windows guests, you can create your own distributions with cloud-init or install it manually. The Windows version is available at https://cloudbase.it/cloudbase-init/.

2.2.2 Installing Virtuozzo Guest Tools

Virtuozzo guest tools enable you to configure running virtual machines from the physical host. With tools you can:

- Run commands in VMs with the `prlctl exec` command.
- Set passwords for users in VMs with the `prlctl set --userpasswd` command. If the user does not exist, it will be created.
- Obtain and change VM network settings.

In addition, installing Virtuozzo guest tools schedules weekly automatic "trimming" of filesystems in Linux guests by means of the `fstrim` service. It reclaims unused storage space by discarding data blocks unused by VM's filesystem. If you disable the service, it will not be re-enabled by future updates of Virtuozzo guest tools. In Windows guests, Storage Optimizer does the same periodic job by default.

2.2. Performing Initial Configuration of Virtual Machines and Containers

Note: If a node is in a Virtuozzo Storage cluster that provides data redundancy by replication, automatic compacting (trimming) of the filesystem(s) that store data replicas is disabled. In case data redundancy is provided by erasure coding, trimming is enabled.

It is recommended to have cloud-init installed in the guest OS (see the previous section). In this case, you only need to do the following:

1. Mount the guest tools image shipped with Virtuozzo to the virtual machine's optical drive. For example:

```
# prlctl installtools MyVM
```

2. Start the virtual machine:

```
# prlctl start MyVM
```

Cloud-init will install the guest tools automatically.

If a VM guest OS does not have cloud-init, you can install Virtuozzo guest tools either automatically or manually.

To install the guest tools automatically without cloud-init:

1. Make sure these requirements are met:

   - The `vz-guest-tools-updater` package is installed on the node.
   - In the `/etc/vz/tools-update.conf` file, the `InstallTools` parameter is set to `true` (default).
   - The virtual machine has the `--tools-autoupdate` parameter set to `on` (default).

2. Stop the virtual machine before installing the guest tools:

```
# prlctl stop MyVM
```

3. Start `vz-guest-tools-updater` for the VM:

```
# vz-guest-tools-updater MyVM
```

4. Start the virtual machine:

```
# prlctl start MyVM
```

Once the VM is launched, the `vz-guest-tools-updater` tool will start installing Virtuozzo guest tools, which can take several minutes.

Important: During installation, Virtuozzo guest tools image is forcibly mounted to VM's optical disk drive even if it is already in use.

To install the guest tools manually without cloud-init:

1. Mount the guest tools image shipped with Virtuozzo to the virtual machine's optical drive. For example:

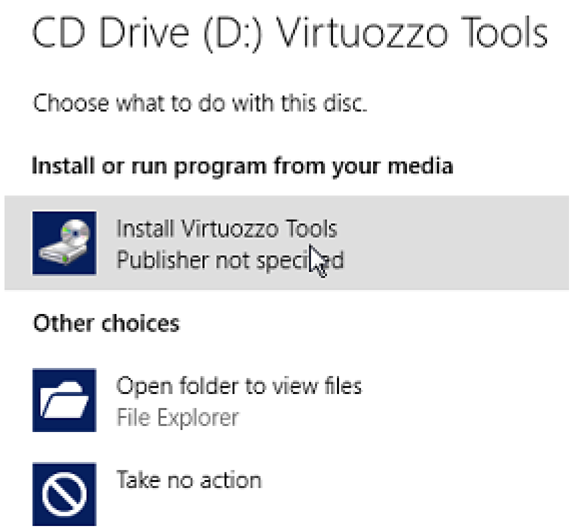
   ```
   # prlctl installtools MyVM
   ```

2. Log in to the virtual machine and do the following:

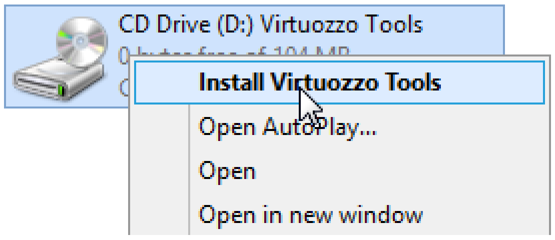
   - Inside a Linux VM, create a mount point for the optical drive with the guest tools image and run the installer:

     ```
     # mount /dev/cdrom /mnt/cdrom
     # bash /mnt/cdrom/install
     ```

   - Inside a Windows VM, if autorun is enabled, launch the installer in the AutoPlay window.


CD Drive (D:) Virtuozzo Tools
Choose what to do with this disc.
Install or run program from your media
Install Virtuozzo Tools
Publisher not specified
Other choices
Open folder to view files
File Explorer
Take no action

     Otherwise right-click the optical drive in Explorer and click **Install Virtuozzo Tools**.



Note:

1. Virtuozzo guest tools rely on QEMU guest agent which is installed alongside the tools. The agent daemon/service (`qemu-ga`) must be running for the tools to work.
2. If you find out that Virtuozzo guest tools are incompatible with some software inside a virtual machine, you can uninstall them from that VM (for details, refer to *Uninstalling Virtuozzo Guest Tools from Virtual Machines* on page 192).

2.2.3 Configuring Network Settings

To make virtual machines and containers accessible from the network, you need to assign valid IP addresses to them and configure DNS servers. The session below illustrates setting these parameters for the virtual machine `MyVM` and the container `MyCT`:

- Assigning IPv4 and IPv6 addresses:

  ```
  # prlctl set MyVM --device-set net0 --ipadd 10.0.186.100/24
  # prlctl set MyVM --device-set net0 --ipadd 1fe80::20c:29ff:fe01:fb07
  # prlctl set MyCT --ipadd 10.0.186.101/24
  # prlctl set MyCT --ipadd fe80::20c:29ff:fe01:fb08
  ```

  `net0` in the commands above denotes the network card in the virtual machine to assign the IP address to. You can view all network cards of a virtual machine using the `prlctl list <VM_name> -i` command.

- Setting DNS server addresses:

  ```
  # prlctl set MyVM --nameserver 192.168.1.165
  # prlctl set MyCT --nameserver 192.168.1.165
  ```

Note:

1. You can configure the network settings only for virtual machines that have Virtuozzo guest tools installed.
2. To assign network masks to containers operating in the `venet0` network mode, you must set the `USE_VENET_MASK` parameter in the `/etc/vz/vz.conf` configuration file to `yes`.

2.2.4 Setting Passwords for Virtual Machines and Containers

In Virtuozzo, you can use the `--userpasswd` option of the `prlctl set` command to create new accounts in your virtual machines and containers directly from the hardware node. The created account can then be used to log in to the virtual machine or container. The easiest way of doing it is to run this command:

```
# prlctl set MyCT --userpasswd user1:2wsx123qwe
```

This command creates the `user1` account in the container `MyCT` and sets the `2wsx123qwe` password for it. Now you can log in to the container as `user1` and administer it in the same way you would administer a standalone server: install additional software, add users, set up services, and so on.

The `prlctl set` command can also be used to change passwords for existing accounts in your virtual machines and containers. For example, to change the password for `user1` in the container `MyCT` to `0pi65jh9`, run this command:

```
# prlctl set MyCT --userpasswd user1:0pi65jh9
```

When setting passwords for virtual machines and containers, keep in mind the following:

- You can manage user accounts only inside virtual machines that have Virtuozzo guest tools installed.
- You should use passwords that meet the minimum length and complexity requirements of the respective operating system. For example, for Windows Server 2008, a password must be more than six characters in length and contain characters from three of the following categories: uppercase characters, lowercase characters, digits, and non-alphabetic characters.
- You should not create accounts with empty passwords for virtual machines and containers running Linux operating systems.

2.2.5 Setting Startup Parameters

The `prlctl set` command allows you to define the `autostart` startup parameter for virtual machines and containers. Setting this parameter to `on` makes your virtual machine or container automatically boot at the physical server startup. For example, to enable the container `MyCT` and the virtual machine `MyVM` to automatically start on your server boot, you can execute the following commands:

- For the container `MyCT`:

  ```
  # prlctl set MyCT --autostart on
  ```

- For the virtual machine `MyVM`:

```
# prlctl set MyVM --autostart on
```

Notice that the `autostart` parameter will have effect only on the next server startup.

2.3 Starting, Stopping, Restarting, and Querying Status of Virtual Machines and Containers

After a virtual machine or container has been created, it can be managed like a usual computer.

2.3.1 Starting Virtual Machines and Containers

You can start virtual machines and containers with the `prlctl start` command. For example:

- To start the container `MyCT`:

```
# prlctl start MyCT
```

- To start the virtual machine `MyVM`:

```
# prlctl start MyVM
```

2.3.2 Stopping Virtual Machines and Containers

You can stop virtual machines and containers with the `prlctl stop` command. For example:

- To stop the container `MyCT`:

```
# prlctl stop MyCT
```

- To stop the virtual machine `MyVM`:

```
# prlctl stop MyVM
```

2.3.3 Restarting Virtual Machines and Containers

You can restart virtual machines and containers with the `prlctl restart` command. For example:

- To restart the container `MyCT`:

```
# prlctl restart MyCT
```

- To restart the virtual machine `MyVM`:

```
# prlctl restart MyVM
```

Note: Restarting virtual machines requires a guest OS and Virtuozzo guest tools to be installed.

2.3.4 Checking Status of Virtual Machines and Containers

You can check the status of a virtual machine or container with the `prlctl status` command. For example:

- To check the status of the container `MyCT`:

```
# prlctl status MyCT
CT MyCT exists running
```

- To check the status of the virtual machine `MyVM`:

```
# prlctl status MyVM
Vm MyVM exists stopped
```

2.4 Listing Virtual Machines and Containers

To get an overview of the virtual machines and containers existing on the physical server and to get additional information about them—their IP addresses, hostnames, current resource consumption, and so on—use the `prlctl list` command. In the most general case, you can get a list of all virtual machines and containers by issuing the following command:

```
# prlctl list -a
UUID                                    STATUS    IP_ADDR       T   NAME
{600adc12-0e39-41b3-bf05-c59b7d26dd73}  running   10.10.1.101   CT  MyCT
{b2de86d9-6539-4ccc-9120-928b33ed31b9}  stopped   10.10.100.1   VM  MyVM
```

The `-a` option shows all—both running and stopped—VMs and containers (only running VMs and containers are shown by default). The default columns include VM and container UUIDs, status, type, IP addresses, and names. The list of columns can be customized with the `-o` option. For example:

```
# prlctl list -a -o name,ctid
NAME                                UUID
```

```
MyCT                            {26bc47f6-353f-444b-bc35-b634a88dbbcc}
MyVM                            {b8cb6d99-1af1-453d-a302-2fddd8f86769}
```

Note: To see a list of all columns, run `prlctl list -L`.

2.5 Cloning Virtual Machines and Containers

You can create a copy (clone) of a particular virtual machine or container that will have identical data and resource parameters. Cloning may save time as clones require little reconfiguration compared to setting up new virtual machines or containers.

You can clone stopped virtual machines and stopped and running containers. For example:

```
# prlctl clone MyCT --name MyCT_clone
# prlctl clone MyVM --name MyVM_clone
```

The `--name` option specifies a name for the clone.

When cloning Windows virtual machines, consider changing their security identifiers (SIDs) with the `--changesid` option.

Successfully cloned virtual machines and containers will be shown in the list of virtual environments on the host. For example:

```
# prlctl list -a
UUID                STATUS       IP_ADDR          T  NAME
{62951c2a-...}  stopped      10.30.10.101     CT MyCT
{49b66605-...}  stopped      10.30.10.101     CT MyCT_clone
{7f4904ad-...}  stopped      10.30.128.115    VM MyVM
{2afb2aa2-...}  stopped      10.30.128.134    VM MyVM_clone
```

The example above shows that the cloned container has the same IP address as the original container. Before starting to use the clones, make sure their IP addresses are unique (for instructions on how to assign IP addresses to VMs and containers, see *Configuring Network Settings* on page 21).

2.5.1 Creating Linked Clones

Starting from Virtuozzo 7.0.7 (Update 7), you can create linked clones of virtual machines. A linked clone is a copy of a virtual machine that shares virtual disks with the original virtual machine. Linked clones take less time and disk space to deploy as they store only changes to the original disks rather than copy them whole. A linked clone cannot run without access to its parent, so make sure the original virtual machine is available and its disks are not corrupted.

To create a linked clone, add the `--linked` option to the `prlctl clone` command. For example:

```
# prlctl clone MyVM --name MyVM_linked_clone --linked
```

On the host, linked clones are shown as usual virtual machines. For example:

```
# prlctl list -a
UUID            STATUS       IP_ADDR          T  NAME
{7f4904ad-...}  stopped      10.30.128.115    VM MyVM
{2e9862f6-...}  stopped      10.30.128.135    VM MyVM_linked_clone
```

Note: Migration, backup, restore, and unlink operations are not supported for linked clones.

2.5.2 Configuring Default Directories

When cloning a virtual machine or container, you can also override the following default directories:

- default directory `/vz/vmprivate/<dest_UUID>` storing the files of a cloned virtual machine (where `<dest_UUID>` denotes the UUID of the resulting virtual machine). To store the files of the `MyVM_clone` virtual machine in a custom directory, you can run the following command:

  ```
  # prlctl clone MyVM --name MyVM_clone --dst vz/vmprivate/customVMs
  ```

  In this case all virtual machine files will be placed to the `/customVMs` directory. Note that the specified directory must exist on the server; otherwise, the command will fail.

- default directory `/vz/private/<dest_UUID>` defining the container private area (where `<dest_UUID>` denotes the UUID of the resulting container). To define a custom private area path for the container `MyCT_clone`, you can execute the following command:

  ```
  # prlctl clone MyCT1 --name MyCT_clone --dst /vz/private/customCTs
  ```

Note: The default `/vz/vmprivate` and `/vz/private` are valid for servers that do not participate in Virtuozzo storage clusters.

2.6 Suspending Virtual Machines and Containers

Virtuozzo allows you to suspend a running virtual machine or container on the physical server by saving its current state to a special file. Later on, you can resume the virtual machine or container and get it in the same state the virtual machine or container was at the time of its suspending. Suspending your virtual machines and containers may prove useful, for example, if you need to restart the physical server, but do not want to:

- quit the applications currently running in the virtual machine or container
- spend much time on shutting down the guest operating system and then starting it again

You can use the `prlctl suspend` command to save the current state of a virtual machine or container . For example, you can issue the following command to suspend the container `MyCT`:

```
# prlctl suspend MyCT
```

At any time, you can resume the container `MyCT` by executing the following command:

```
# prlctl resume MyCT
```

Once the restoration process is complete, any applications that were running in the container `MyCT` at the time of its suspending will be running again and the information content will be the same as it was when the container was suspended.

2.7 Running Commands in Virtual Machines and Containers

Virtuozzo allows you to execute arbitrary commands inside virtual machines and containers by running them on the physical server, i.e. without the need to log in to the respective virtual machine or container. For example, this can be useful in these cases:

- If you do not know the virtual machine or container login information, but need to run some diagnosis commands to verify that it is operational.
- If the virtual machine or container has no network access.

In both cases, you can use the `prlctl exec` command to run a command inside the respective virtual machine or container. By default, running `prlctl exec <command>` is equivalent to executing `bash -c <command>` in a Linux VM or container or `cmd /c <command>` in a Windows VM. Adding the `--without-shell` option allows running commands directly without the shell.

The session below illustrates the situation when you run the stopped SSH daemon inside a Linux virtual machine with the name of `My_Linux`:

```
# prlctl exec My_Linux /etc/init.d/sshd status
openssh-daemon is stopped
# prlctl exec My_Linux /etc/init.d/sshd start
Starting sshd: [  OK  ]
# prlctl exec My_Linux /etc/init.d/sshd status
openssh-daemon (pid 26187) is running...
```

Note:

1. You can use the `prlctl exec` command only inside virtual machines that have Virtuozzo guest tools installed.
2. The `prlctl exec` command is executed inside a virtual machine or container from the `/` directory rather than from `/root`.

2.8 Deleting Virtual Machines and Containers

You can delete a virtual machine or container that is not needed anymore using the `prlctl delete` command. Note that you cannot delete a running or mounted virtual machine or container. The example below illustrates deleting the running container `MyCT`:

```
# prlctl delete MyCT
Removing the CT...
Failed to remove the CT: Unable to complete the operation. This operation cannot \
be completed because the virtual machine "{4f27f27f-c056-4a65-abf6-27642b6edd21}"\
is in the "running" state.
# prlctl stop MyCT
Stopping the CT...
The CT has been successfully stopped.
# prlctl delete MyCT
Removing the CT...
The CT has been successfully removed.
```

2.9 Viewing Detailed Information About Virtual Machines and Containers

To view detailed information about a virtual machine or container, you can use the `prlctl list -i` command. For example, the following command lists all information about the virtual machine `MyVM`:

```
# prlctl list -i MyVM
```

The following table describes the main options displayed by `prlctl list -i`.

Option	Description
ID	Virtual machine identifier. Usually, you use this ID, along with the virtual machine name, when performing an operation on the virtual machine.
EnvID	Kernel virtual machine identifier. This is the ID the kernel on the physical server uses to refer to a virtual machine when displaying some information on this virtual machine.
Name	Virtual machine name.
Description	Virtual machine description.
State	Virtual machine state.
OS	Guest operating system installed in a virtual machine.
Uptime	Time that shows for how long a virtual machine has been running since counter reset. **Note:** The uptime counter as well as count start date and time can be reset with the `prlctl reset-uptime` command.
Home	Directory storing virtual machine files.
Guest tools	Shows whether Virtuozzo guest tools are installed in a virtual machine.
Autostart	Shows whether a virtual machine is automatically started when you turn on the physical server.
Boot order	Order in which the virtual machine devices are checked for an operating system.
Hardware	Devices available in a virtual machine.
Offline management	Denotes whether the offline management feature is enabled for the virtual machine, and if yes, lists the available offline services.

Note: The options `prlctl list` displays for containers are similar to those for virtual machines.

2.10 Managing Virtual Machine and Container Backups

Backing up your virtual machines and containers on a regular basis is essential for system reliability. Virtuozzo allows you to back up and restore virtual machines and containers on the local server with the `prlctl` and `prlsrvctl` utilities.

2.10.1 Creating Virtual Machine and Container Backups

You can create backups of virtual machines and containers with the `prlctl backup` command. The command is executed on the local server where the virtual machines or containers are located. The resulting backups can be stored on either the same local server or a remote server (e.g., a dedicated backup server).

By default, an incremental backup is created that contains only the files changed since the previous full or incremental backup. If no previous backups exist, a full backup is created. (You can also create a full backup with the `-f` option.)

Note: For increased security during backup operations, Virtuozzo provides connection tunneling between the local server and remote backup server. Tunneling increases backup time, so if you want to speed up the process and do not need a secure tunnel between servers, you can disable connection tunneling with the `--no-tunnel` option. To use it, configure the firewall of the destination server to allow incoming connections on any port on the corresponding network interface.

To create a backup of the virtual machine `MyVM` and store it on the local server, run the following command:

```
# prlctl backup MyVM
...
The VM has been successfully backed up with \
backup id {746dba2a-3b10-4ced-9dd6-76a2b1c14a69}
```

The backup UUID, like the one shown above, will be required to manage the backup in the future.

If you want to create a backup of the virtual machine `MyVM` and store said backup on a remote server, specify the remote server's IP address or hostname with the `-s` option, for example:

```
# prlctl backup MyVM -s 192.168.0.10
```

The root account is used to log in to the remote server by default, so you will be asked for the root password. You can also provide different credentials (and port) in the format `[<user>[:<passwd>]@]<server>[:<port>]]`.

By default, backups are placed in the `/vz/vmprivate/backups` directory. To set a different default backup directory, use the `prlsrvctl set --backup-path <path>` command.

Note:

1. You can back up both running and stopped virtual machines and containers.
2. Creating a consistent backup of a running virtual machine requires the Virtuozzo guest tools to be installed in said virtual machine.
3. You cannot back up virtual machines with attached physical HDDs, mounted ISO or floppy disk images, etc.

2.10.2 Listing Virtual Machine and Container Backups

You can list backups on the server with the `prlctl backup-list` command. For example:

```
# prlctl backup-list
           ID  Backup_ID      Node      Date                 Type  Size
{c1dee22f...}  {209d54a0...}  test.com  2011-05-30 10:19:32  f     411566405
[The ID and Backup ID are reduced for better readability.]
```

The command output shows that currently only one backup with the ID of `209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca` exists on the server. The information on the backup is presented in the following table:

Column	Description
ID	Virtual machine or container UUID.
Backup_ID	Backup UUID. You need to specify this ID when performing any backup-related operations.
Node	The hostname of the physical server storing the backup archive.
Date	The date and time when the backup archive was created.

Column	Description
Type	The backup type. Currently, you can create two types of backups: • full, `f`, • incremental, `i`, with only the files changed since the previous full or incremental backup. This is the default backup type.
Size	The size of the backup image, in bytes.

If required, you can filter the backup list with the `--vmtype ct|cm|all` option that only shows backups of containers, virtual machines, or both. To list only backups created on the local server, use the `--localvms` option.

2.10.3 Restoring Virtual Machines and Containers from Backups

Local or remote backups of virtual machines and containers can be restored with the `prlctl restore` command.

Note: For increased security during restore operations, Virtuozzo provides connection tunneling between the backup and destination servers. Tunneling increases restore time, so if you want to speed up the process and do not need a secure tunnel between servers, you can disable connection tunneling with the `--no-tunnel` option. To use it, configure the firewall of the destination server to allow incoming connections on any port on the corresponding network interface.

The following rules and considerations apply:

- Restore commands are run on the destination server (to which the backups will be restored).
- Only stopped virtual machines and containers can be restored from backup.
- Virtuozzo 6 backups can be restored to Virtuozzo 7 servers (with conversion to Virtuozzo 7 format).
  - Backups of virtual machines and containers with guests unsupported in Virtuozzo 7 may not be restored correctly (see *Supported Guest Operating Systems* on page 16).
  - VZFS-based containers must be converted to ploop format and backed up again before they can be restored to Virtuozzo 7.

> **Note:** If conversion of the restored VM fails, the restored VM is deleted from the destination server and you can try again. If the second attempt also fails, you need to enable a legacy VM debug mode on the destination Virtuozzo 7 server (see *Enabling Legacy VM Debug Mode* on page 195), make another restore attempt, send the problem report, and contact the technical support team. With the debug mode enabled, the migrated VM will not be deleted from the destination Virtuozzo 7 server after conversion failure. It will remain stopped or running with disabled network to let the technical support team study the memory dump and find out the reason for failure.

To restore a backup of the virtual machine `MyVM` with the UUID `a53f1184-333e-41cf-b410-2ec8ffea67d4`, run

```
# prlctl restore MyVM
```

or

```
# prlctl restore a53f1184-333e-41cf-b410-2ec8ffea67d4
```

If multiple backups of a virtual machine exist, the latest one is restored. To restore a particular backup, specify its ID with the `-t` option. For example:

```
# prlctl restore -t 24a3011c-8667-4870-9e11-278f1398eab0
```

If backups are stored on a remote server and the VM or container to be restored does not exist on the destination server, you can restore it by specifying the VM or container's UUID or the backup UUID with the `-t` option and that server's IP address or hostname with the `-s` option. For example:

```
# prlctl restore -t 24a3011c-8667-4870-9e11-278f1398eab0 -s 192.168.0.10
```

If the VM or container already exists on the destination server (e.g., has been restored from a remote backup once), you can also restore its latest backup by specifying the VM or container's name.

If necessary, you can transfer remotely stored backups to the local server and restore them locally. To do this:

1. Find out the default backup directories on the source and destination servers by running `prlsrvctl info | grep "Backup path"` on each.
2. Copy backups to the default backup directory on the destination server. Or, if you keep backups on a network storage, attach said network storage to the default backup directory on the destination server.
3. Restore the backup with the `prlctl restore -t` command as shown in the example above.

2.10.4 Deleting Virtual Machine and Container Backups

You can delete backups with the `prlctl backup-delete` command.

To remove a specific VM or container backup, provide the VM or container name or UUID as well as the backup ID (you can find out these IDs with the `prlctl backup-list` command). For example:

```
# prlctl backup-list
           ID  Backup_ID      Node      Date                 Type  Size
{c1dee22f...}  {209d54a0...}  test.com  2011-05-30 10:19:32  f     411566405
[The ID and Backup ID are reduced for better readability.]
# prlctl backup-delete MyVM -t 209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca --keep-chain
```

With the `--keep-chain` option specified, the remaining backup chain will be preserved after specific backups are deleted from it.

To delete all backups of a VM or container, specify only the VM or container name or UUID:

```
# prlctl backup-delete MyVM
```

2.10.5 Backing Up Entire Servers

In addition to backing up single virtual machines and containers, you can create backups of all virtual environments on the server with the `prlsrvctl backup` command. For example:

```
# prlsrvctl backup -f
Backing up the CT MyCT
...
The CT has been successfully backed up with backup id {b14ec76d-c0e2-432f-859a- \
4727c0042065}
Backing up the VM MyVM
...
The VM has been successfully backed up with backup id {746dba2a-3b10-4ced-9dd6- \
76a2blcl4a69}.
```

Note:

1. You can back up both running and stopped virtual machines and containers.
2. Creating consistent backups of running virtual machines requires the Virtuozzo guest tools to be installed in said virtual machines.
3. You cannot back up virtual machines with attached physical HDDs, mounted ISO or floppy disk images, etc.

2.10.6 Attaching Backups to Virtual Machines and Containers

To read the contents of a virtual machine or container backup, you can attach it to a virtual machine or container as a virtual hard disk.

Note:

1. Only local backups can be attached.
2. The attached backup is writable so that the filesystem can process its journal on mount. However, all changes will be discarded when the backup is detached. The amount of data that can be written to the attached backup is limited to 256MB.
3. Attached backups are not preserved during clone, backup, and snapshot operations.

2.10.6.1 Attaching Backups to Linux Virtual Machines

1. Make sure that the `prl_backup` and `kpartx` utilities are installed in the virtual machine the backup will be attached to. The `prl_backup` utility is provided by Virtuozzo guest tools.
2. Obtain the ID and file name of the backup to attach. You can do this with the `prlctl backup-list` command. For example:

   ```
   # prlctl backup-list vm2 -f
   ...
   Backup_ID: {0fcd6696-c9dc-4827-9fbd-6ee3abe017fa}
   ...
   Name: harddisk.hdd.qcow2c
   ```

3. Attach the backup as an HDD to the Linux VM you will access the backup from. You can do this with the `prlctl set --backup-add` command. For example:

   ```
   # prlctl set vm1 --backup-add {0fcd6696-c9dc-4827-9fbd-6ee3abe017fa} \
   --disk harddisk.hdd.qcow2c
   Creating hdd1 (+) sata:2 real='backup:///{0fcd6696-c9dc-4827-9fbd-6ee3abe017fa}/ \
   harddisk.hdd.qcow2c' backup='{0fcd6696-c9dc-4827-9fbd-6ee3abe017fa}' \
   disk='harddisk.hdd.qcow2c'
   ```

   If the backup contains multiple disks and you need to connect them all, omit the `--disk` parameter.

4. Obtain the name of the newly attached device, which is disabled at the moment, using the `prl_backup list` command. For example:

```
# prlctl exec vm1 prl_backup list
...
List of disabled attached backups:
[1] /dev/sdc
```

5. Enable the backup with the `prl_backup enable` command. For example:

   ```
   # prlctl exec vm1 prl_backup enable /dev/sdc
   ```

6. Optionally, make sure the backup is now enabled, using the `prl_backup list -e` command. For example:

   ```
   # prlctl exec vm1 prl_backup list -e
   List of enabled attached backups:
   [1] /dev/sdc (/dev/mapper/backup1)
   NAME                TYPE  SIZE   FSTYPE   UUID
   MOUNTPOINT
   backup1 (dm-3)      dm    64G
   |-backup1p1 (dm-4)  part  500M   ext4     1ac82165-113d-40ee-8ae2-8a72f62d95bf
    -backup1p2 (dm-5)  part  63.5G  LVM2_mem Zw9QiY-BiU5-o8dn-ScTK-vOZx-KujW-wbgmS3
   ```

Now you can mount the required backup part as a filesystem.

Mounting the `ext4` part requires no additional steps. For example:

```
# prlctl exec vm1 mount /dev/mapper/backup1p1 /mnt/backup1p1
```

You can now access the backup part contents at `/mnt/backup1p1`.

Mounting the `LVM2_member` part requires the following preparations:

1. Assign the new volume group a new name so it can coexist with other volume groups. You can do this with the `vgimportclone` command. For example:

   ```
   # prlctl exec vm1 vgimportclone -n backup1p2 /dev/mapper/backup1p2
   ...
   Volume group "VolGroup" successfully renamed to "backup1p2"
   ...
   Found volume group "backup1p2" using metadata type lvm2
   ...
   ```

2. Obtain the list of mountable logical volumes with the `lvs` command. For example:

   ```
   # prlctl exec vm1 lvs | grep backup1p2
   lv_home backup1p2 -wi-------11.54g
   lv_root backup1p2 -wi------- 50.00g
   lv_swap backup1p2 -wi-------1.97g
   ```

3. Activate the required logical volume with the `lvchange -ay` command. For example:

```
# prlctl exec vm1 lvchange -ay /dev/backup1p2/lv_root
```

4. Mount the logical volume as a filesystem. For example:

```
# prlctl exec vm1 mount /dev/backup1p2/lv_root /mnt/backup1p2
```

You can now access the backup part contents at `/mnt/backup1p2`.

2.10.6.2 Attaching Backups to Windows Virtual Machines

1. Obtain the backup ID and file name. For example:

```
# prlctl backup-list vm2 -f
...
Backup_ID: {cff742a9-f942-41c5-9ac2-ace3b4eba783}
...
Name: harddisk.hdd.qcow2c
```

2. Attach the required backup as an HDD to the Windows virtual machine you will access the backup from. For example:

```
# prlctl set vm1 --backup-add {cff742a9-f942-41c5-9ac2-ace3b4eba783} \
--disk harddisk.hdd.qcow2c
Creating hdd1 (+) sata:2 real='backup:///{cff742a9-f942-41c5-9ac2-ace3b4eba783}/ \
harddisk.hdd.qcow2c' backup='{cff742a9-f942-41c5-9ac2-ace3b4eba783}' \
disk='harddisk.hdd.qcow2c'
```

The attached backup will appear as a ready-to-use disk in the Windows virtual machine.

2.10.6.3 Attaching Backups to Linux Containers

1. Obtain the backup ID and file name with the prlctl `backup-list -f` command. For example:

```
# prlctl backup-list 102 -f
...
Backup_ID: {d70441dd-f077-44a0-8191-27704d4d8fdb}
...
Name: root.hdd.qcow2c
...
```

2. Attach the backup as an HDD to the Linux container you will access the backup from. You can do this with the `prlctl set --backup-add` command. For example:

```
# prlctl set MyCT --backup-add {d70441dd-f077-44a0-8191-27704d4d8fdb} \
--disk root.hdd.qcow2c
Creating hdd1 (+) sata:0 real='backup:///{d70441dd-f077-44a0-8191-27704d4d8fdb}/ \
root.hdd.qcow2c' backup='{d70441dd-f077-44a0-8191-27704d4d8fdb}' \
disk='root.hdd.qcow2c'
```

3. Using the backup ID, identify the ploop device corresponding to the backup. For example:

```
# ploop list | grep {d70441dd-f077-44a0-8191-27704d4d8fdb}
ploop28261 /buse/{8417a267-0919-4c8f-a31d-68671358d6a8}_ \
{d70441dd-f077-44a0-8191-27704d4d8fdb}_root.hdd.qcow2c/content
```

4. Mount the logical volume as a filesystem. For example:

```
# prlctl exec MyCT mount /dev/ploop28261p1 /mnt/backup1
```

You can now access the backup contents at `/mnt/backup1`.

2.10.7 Detaching Backups from Virtual Machines and Containers

Note: Before detaching a backup from a running virtual machine, do the following:

1. (Linux VMs) Disable the backup device with the `prl_backup disable` command run in the guest OS.
2. (Linux and Windows VMs) Disconnect the corresponding virtual disk by running `prlctl set --device-set hdd<N> --disconnect` command on the server.

- To detach all virtual disks from all backups attached to a virtual machine or container, use the `prlctl set --backup-del all` command. For example:

```
# prlctl set vm1 --backup-del all
```

- To detach all virtual disks from a specific backup attached to a virtual machine or container, use the `prlctl set --backup-del <backup_ID>` command. For example:

```
# prlctl set vm1 --backup-del {e13561bb-5676-49bd-a935-ae0145eb0229}
```

- To detach a specific virtual disk from any of the backups attached to a virtual machine or container, delete said disk with the `prlctl set --device-del hdd<N>` command. For example:

```
# prlctl set vm1 --device-del hdd1
```

2.11 Managing Templates

A template in Virtuozzo is a pre-configured virtual machine or container that can be easily and quickly deployed into a fully functional virtual machine or container. Like any normal virtual machine or container, a template contains hardware (virtual disks, peripheral devices) and the operating system. It can also have additional software installed. In fact, the only main difference between a virtual machine or container and a template is that the latter cannot be started.

You can perform the following operations on templates:

- create a new template,
- list existing templates,
- create a virtual machine or container from a template,
- migrate templates between Virtuozzo servers (see *Migrating Virtual Machine and Container Templates* on page 50),
- store templates on Virtuozzo Storage.

These operations are described in the following subsections in detail.

Note: In addition, see *Using Customized EZ Templates* on page 173 for details on how to manually create and use custom container templates.

2.11.1 Creating Templates

In Virtuozzo, you can create a template using the `prlctl clone` utility. Making a template may prove useful if you need to create several virtual machines or containers with the same configuration. In this case, your steps can be as follows:

1. You create a virtual machine or container with the required configuration.
2. You make a template on the basis of the created virtual machine or container.
3. You use the template to create as many virtual machines or containers as necessary.

Let us assume that you want to create a template of the virtual machine `MyVM`. To do this, you can run the following command:

```
# prlctl clone MyVM --name template1 --template
```

This command clones the virtual machine and saves it as the `template1` template. After the template has been successfully created, you can use it for creating new virtual machines.

2.11.2 Listing Templates

Sometimes, you may need to get an overview of the templates available on your hardware node. For example, this may be necessary if you plan to create a virtual machine or container from a specific template, but do not remember its exact name. In this case, you can use the `prlctl list` command to list all templates on the hardware node and find the one you need:

```
# prlctl list -t
UUID                                    DIST            T  NAME
{017bfdf0-b546-4309-90d0-147ce55773f2}  centos          VM centos_tmpl
{92cd331e-0572-46ac-8586-f19b8d029c4d}  centos          CT ct201_tmp1
{fc40e38e-8da4-4b26-bb18-6098ec85f7b4}  debian          VM deb_tmpl
{0dea5912-b114-45a9-bd1a-dc065c1b8e9f}  ubuntu          VM ubuntu_tmp1
{479e66aa-332c-4e3e-975e-b8b6bfc9d2e0}  win-2012        VM w12en_tmpl
```

In this example, 5 templates exist on the server. The information on these templates is presented in the form of a table with the following columns (from left to right): the template ID, the operating system contained in the template, the template type (for a container or virtual machine) and the template name.

2.11.3 Deploying Templates

To deploy a virtual machine or container from a template, use the `--ostemplate` option of the `prlctl create` command. For example, to deploy the virtual machine `MyVMtemplate1` from the template `template1`, run the following:

```
# prlctl create MyVMtemplate1 --ostemplate template1
```

To check that the virtual machine has been successfully created, use the `prlctl list -a` command:

```
# prlctl list -a
STATUS       IP_ADDR         NAME
running      10.12.12.101    MyVM
stopped      10.12.12.34     MyVMtemplate1
```

The template itself is left intact and can be used for creating other virtual machines:

```
# prlctl list -t
{4ad11c28-9f0e-4086-84ea-9c0487644026} win-2008 template1
{64bd8fea-6047-45bb-a144-7d4bba49c849} rhel     template2
```

2.11.4 Storing Templates on Virtuozzo Storage

Starting from Virtuozzo 7.0.7 (Update 7), you can store container and virtual machine templates in shared directories of Virtuozzo Storage clusters. These templates will be available to any server participating in the cluster.

To place a template on Virtuozzo Storage, do as follows on the cluster node where the source container or VM is located:

1. Create a template. For example:

   ```
   # prlctl clone MyVM --name template1 --template
   ```

2. Move this template to the `vmtemplates` directory located on Virtuozzo Storage:

   - for Virtuozzo Storage with CLI management, the path is `/vstorage/<cluster_name>/vmtemplates`, e.g.:

     ```
     # prlctl move template1 --dst /vstorage/vstor1/vmtemplates
     ```

   - for Virtuozzo Storage with GUI management, the path is `/mnt/vstorage/vmtemplates`, e.g.:

     ```
     # prlctl move template1 --dst /mnt/vstorage/vmtemplates
     ```

Within five minutes, the template will be autodetected by the `prlctl` utility. You can check template availability by listing templates on another Virtuozzo Storage server. For example:

```
# prlctl list -t | grep template1
{4ad11c28-9f0e-4086-84ea-9c0487644026} win-2008 template1
```

Once the template is available throughout the cluster, you can start creating containers or VMs based on it on any cluster node as described in *Deploying Templates* on page 40.

2.12 Managing Snapshots

In Virtuozzo, you can save the current state of a virtual machine or container by creating a snapshot. You can then continue working in your virtual machine or container and return to the saved state any time you wish. Snapshots may be useful in the following cases:

- Configuring applications with a lot of settings. You may wish to check how settings work before applying them to the application. So, before you start experimenting, you create a snapshot.
- Participating in large-scale development projects. You may wish to mark development milestones by

creating a snapshot for each. If anything goes wrong, you can easily revert to the previous milestone and resume the development.

In Virtuozzo, you can create, list, revert to, and delete snapshots. These operations are described in the following subsections.

2.12.1 Creating Snapshots

To create a snapshot of a virtual machine or container, use the `prlctl snapshot` command.

2.12.1.1 Creating Virtual Machine Snapshots

To create a snapshot of the virtual machine `MyVM`, do the following:

```
# prlctl snapshot MyVM
...
The snapshot with ID {12w32198-3e30-936e-a0bbc104bd20} has been successfully created.
```

A newly created snapshot is saved to the `/vz/vmprivate/<UUID>/Snapshots/<snapshot_ID>.pvs` file, where `<UUID>` is the corresponding virtual machine ID and `<snapshot_ID>` is a unique snapshot ID. In the above example, the snapshot with ID `{12w32198-3e30-936e-a0bbc104bd20}` is saved to the file `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/Snapshots/{12w32198-3e30-936e-a0bbc104bd20}.pvs`.

```
# ls /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/Snapshots/
{063615fa-f2a0-4c14-92d4-4c935df15840}.pvc
```

Snapshot IDs are needed to switch to and delete snapshots.

When creating a snapshot, you can also set its name and description:

```
# prlctl snapshot MyVM -n Clean_System -d "This snapshot was created right after \
 installing Windows XP."
...
The snapshot with ID {0i8798uy-1eo0-786d-nn9ic106b9ik} has been successfully created.
```

You can then view the set name and description in the `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/Snapshots.xml` file.

2.12.1.2 Creating Container Snapshots

To create a snapshot of the container `MyCT`, do the following:

```
# prlctl snapshot MyCT
...
The snapshot with ID {08ddd014-7d57-4b19-9a82-15940f38e7f0} has been successfully \
created.
```

A newly created snapshot is saved to the `/vz/private/<UUID>/dump/<snapshot_ID>` file, where `<UUID>` is the container UUID and `<snapshot_ID>` is a snapshot ID. In the example above, the snapshot with ID `{08ddd014-7d57-4b19-9a82-15940f38e7f0}` is saved to the file `/vz/private/26bc47f6-353f-444b-bc35-b634a88dbbcc/dump/{08ddd014-7d57-4b19-9a82-15940f38e7f0}`.

```
# ls /vz/private/26bc47f6-353f-444b-bc35-b634a88dbbcc/dump
{08ddd014-7d57-4b19-9a82-15940f38e7f0}
```

Snapshot IDs are needed to switch to and delete snapshots.

When creating a snapshot, you can also set its name and description:

```
# prlctl snapshot MyCT --n Clean_System --d "This snapshot was created right after \
installing Windows XP."
...
The snapshot with ID {e78bb2b8-7a99-4c8b-ab9a-491a47648c44} has been successfully \
created.
```

The set name and description are stored in the `/vz/private/<UUID>/Snapshots.xml` file.

2.12.1.3 Snapshot Branching

Snapshot branches can be useful for working with, testing or comparing similar configurations. A snapshot branch is created when you do the following:

1. Create several snapshots.
2. Revert to one of the snapshots.
3. Make changes to the virtual machine or container.
4. Create a snapshot.

In this case, the newly created snapshot will start a new branch based on the snapshot from Step 2.

2.12.1.4 Restrictions and Recommendations

- Virtual machine and snapshot names and snapshot descriptions containing spaces must be enclosed in quotation marks (e.g., "Windows XP") when supplying them to the prlctl command.
- Before creating a snapshot, it is recommended that you finish any installations, downloads, and stop

writing to external devices. You should also complete or cancel any transactions performed via the virtual machine in external databases.

- You cannot create snapshots for containers with enabled NFS server feature.

2.12.2 Listing Snapshots

To list all snapshots of a particular virtual machine or container, use the `prlctl snapshot-list` command. For example, to check all current snapshots of the virtual machine `MyVM`, run this command:

```
# prlctl snapshot-list MyVM
PARENT_SNAPSHOT_ID                      SNAPSHOT_ID
                                        {989f3415-3e30-4494-936e-a0bbc104bd20}
{989f3415-3e30-4494-936e-a0bbc104bd20}  *{063615fa-f2a0-4c14-92d4-4c935df15840}
```

This command shows that the virtual machine `MyVM` has two snapshots. The snapshot with ID `{063615fa-f2a0-4c14-92d4-4c935df15840}` is based on the snapshot with ID `{989f3415-3e30-4494-936e-a0bbc104bd20}`, i.e. the former is a child of the latter. The asterisk marks the current snapshot.

To view the relationships between snapshots, use the `-t` option:

```
# prlctl snapshot-list MyVM -t
_{989f3415-3e30-4494-936e-a0bbc104bd20}_{063615fa-f2a0-4c14-92d4-4c935df15840}\
*{712305b0-3742-4ecc-9ef1-9f1e345d0ab8}
```

The command output shows you that currently two branches exist for the virtual machine `MyVM`. The snapshot with ID `{989f3415-3e30-4494-936e-a0bbc104bd20}` is the base for these branches.

To get detailed information on a particular snapshot, use the `-i` option with the snapshot ID:

```
# prlctl snapshot-list MyVM -i {063615fa-f2a0-4c14-92d4-4c935df15840}
ID: {063615fa-f2a0-4c14-92d4-4c935df15840}
Name: Clean_System
Date: 2012-07-22 22:39:06
Current: yes
State: poweroff
Description: <![CDATA[This snapshot was created right after installing Windows 7]]>
```

The `prlctl snapshot-list` command with the `-i` option displays the following information about snapshots:

Field	Description
ID	ID assigned to the snapshot.
Name	Name assigned to the snapshot.
Date	Date and time when the snapshot was created.

Field	Description
Current	Denotes that this is the current snapshot of the virtual machine.
State	State the virtual machine was in at the time you took the snapshot.
Description	The description set for the snapshot.

2.12.3 Reverting to Snapshots

To revert to a snapshot, use the `prlctl snapshot-switch` command. When you revert to a snapshot, the current state of the virtual machine or container is discarded, and all changes made to the system since the previous snapshot are lost. So, before reverting, you may want to save the current state by creating a new snapshot (see *Creating Snapshots* on page 42).

The `prlctl snapshot-switch` command requires the virtual machine or container name and the snapshot ID as arguments, for example:

```
# prlctl snapshot-switch MyVM --id {cedbc4eb-dee7-42e2-9674-89d1d7331a2d}
```

In this example, you revert to the snapshot `{cedbc4eb-dee7-42e2-9674-89d1d7331a2d}` for the virtual machine `MyVM`.

2.12.4 Deleting Snapshots

To delete unneeded snapshots of virtual machines or containers, use the `prlctl snapshot-delete` command. For example:

```
# prlctl snapshot-delete MyVM --id {903c12ea-f6e6-437a-a2f0-a1d02eed4f7e}
```

When you delete a parent snapshot, child snapshots are not deleted, and the information from the former is merged into the latter.

2.13 Migrating Virtual Machines and Containers

To facilitate hardware upgrades and load balancing between multiple hosts, Virtuozzo enables you to migrate virtual machines and containers between physical servers with the `prlctl migrate` command.

Important: For migration to work, a direct SSH connection on port 22 must be allowed between the source and destination servers.

Before migration, make sure that the destination server:

- has enough hard disk space to store the resulting virtual machine or container,
- has enough memory and CPU power to run the resulting virtual machine or container,
- has a stable network connection with the source server.

Note: Migration of virtual machines with snapshots is not supported (any existing snapshots must be deleted).

You can migrate VMs and containers both to and from a remote server. For example, to move a VM to a remote server, run this command on the local server:

```
# prlctl migrate MyVM root:passwd@remoteserver.com
```

To move a VM from a remote server, run this command the local server:

```
# prlctl migrate root:passwd@remoteserver.com/MyVM localhost
```

If you do not provide the destination server credentials in the command, you will be asked to do so during migration.

If you want to place the migrated virtual environment in a custom directory on the destination server, specify the full path to that directory in the `--dst=<custom_path>` option. The resulting path to the migrated virtual environment files will be `<custom_path>/<VE_UUID>`.

By default, once migration is complete:

- the original virtual machine is removed from the source server,
- the `.migrated` suffix is added to the names of the original container's private area and configuration file on the source server.

Adding the `--clone` option to the `prlctl migrate` command enables you to skip the two default actions above. That is, to keep the original VM and not to add the `.migrated` suffix to container's private area and configuration file. The clone will have a different UUID, MAC address, SID (for Windows-based VMs only; if the `--changesid` option is specified), and offline management disabled.

Migration implies transferring large amounts of data between servers which can take considerable time. To reduce the amount of data to be transferred, Virtuozzo has compression enabled by default. Compression consumes additional server resources and can be disabled if necessary with the `--no-compression` option.

2.13.1 Types of Migration

Virtuozzo allows you to perform two types of migration between Virtuozzo servers:

- Offline migration for stopped and suspended containers and virtual machines.
- Online (live) migration for running containers and running and paused virtual machines. Containers and virtual machines may be located on Virtuozzo Storage or local storage.

Both types are described in the following sections.

2.13.1.1 Offline Migration of Virtual Machines and Containers

Offline migration implies copying all files of a virtual machine or container from one server to another over the network.

2.13.1.2 Live Migration of Virtual Machines and Containers

The process of migrating virtual machines and containers live is as follows:

1. Once you start the migration, Virtuozzo checks whether the destination server meets all the migration requirements and the virtual machine or container can be migrated to it.
2. All virtual memory and disks of the virtual machine or container are migrated to the destination server.
3. The virtual machine or container on the source server is suspended.
4. The changed memory pages and virtual disk blocks, if any, are migrated to the destination server.
5. The virtual machine or container is resumed on the destination server.

The virtual machine or container continues running during steps 1 and 2 and is not available to the user during steps 3-5. But since the amount of memory pages and virtual disk blocks changed during step 2 is small, the downtime is almost imperceptible.

After migration, the relocated virtual machine or container may not be accessible over the network for several minutes due to network equipment reconfiguration (for example, as switches are updating their dynamic VLAN membership tables).

Note: For increased security during live migration, Virtuozzo provides connection tunneling between the source and destination servers. Tunneling increases migration time. If you do not need a secure tunnel between servers, you can speed up VM live migration by disabling connection tunneling with the `--no-tunnel` option. Tunnelless container migration is not supported. To use the option, configure the firewall of the destination server to allow incoming connections on any port on the corresponding network interface.

Live Migration Requirements and Restrictions

When performing live migration, take into account the following requirements and restrictions:

- Before starting live migration, it is recommended to synchronize the system time on the source and destination servers, for example, by means of NTP (http://www.ntp.org). The reason is that certain processes running in virtual machines and containers may rely on system time being steady and might behave unpredictably when resumed on a destination server where time is different.

  Note: In Virtuozzo 7, time synchronization via NTP is enabled by default using the `chronyd` service. If you want to use `ntpdate` or `ntpd`, stop and disable `chronyd` first.

- The network must support data transfer rates of at least 1 Gbps.
- The source and destination servers must belong to the same subnetwork.
- The CPUs on the source and destination servers must be manufactured by the same vendor, and the CPU capabilities of the destination server must be the same or exceed those on the source server.
- Virtual machine and container disks can be located on local disks, shared NFS and GFS2 storages, and iSCSI raw devices.
- Live migration is not supported for virtual machines and containers with open `prlctl enter` sessions and containers with IPSec connections.
- Containers with NFS clients inside can be migrated only if the following conditions are met:
  - Block and character device files shared over NFS are not opened.
  - Remote file locking and over-mounted NFS file systems are not used simultaneously.

> **Note:** Migration of local file locks is supported only for NFS version 3 as it has native support of such locks.

- Live migration is not supported for containers with enabled NFS server feature.

2.13.2 Migrating Virtual Machines and Containers Between Virtuozzo 7 Servers

You can migrate Virtuozzo 7 virtual machines and containers to other Virtuozzo 7 servers in any mode: online or offline (for details, see *Types of Migration* on page 47).

Virtual machines and containers stored in a Virtuozzo Storage cluster are migrated between cluster nodes without data copying, which significantly reduces migration time.

2.13.3 Migrating Virtual Machines and Containers from Virtuozzo 6 to Virtuozzo 7

You can migrate older Virtuozzo 6 virtual machines (online) and containers (offline) to Virtuozzo 7 servers. During migration, such VMs and containers will be converted in the Virtuozzo 7 format. In particular, VM devices will be replaced by those supported in Virtuozzo 7 (for a list of VM hardware supported in Virtuozzo 7, see *Virtual Machine Hardware* on page 8).

> **Note:** Migration from Virtuozzo 6 to Virtuozzo 7 implies VM and container downtime that depends on network bandwidth, virtual machine RAM size, and server load. To reduce downtime, it is recommended to at least perform migration when the server load is minimal.

Before migrating Virtuozzo 6 virtual machines, make sure these requirements are met in addition to those listed in *Live Migration Requirements and Restrictions* on page 48:

- The hardware node must be running the latest version of Virtuozzo 6 (or at least 6.0.11-3466).
- The VM must have a guest OS installed.
- The VM must be running.
- The VM must not have snapshots.

For example, to migrate the virtual machine `MyVM` from Virtuozzo 6 to Virtuozzo 7, run the following command on the Virtuozzo 6 server:

```
# prlctl migrate MyVM root:<passwd>@<VZ7_host_IP_address_or_hostname>
```

During migration, the virtual machine is copied to the destination server and converted to the Virtuozzo 7 format. After a successful migration, the original Virtuozzo 6 virtual machine is unregistered and the `.migrated` suffix is added to its directory name.

Note: To avoid restarting migrated legacy VMs during conversion, such VMs have their `--on-crash` parameter set to `paused` (while the default value is `restart`). After successful migration, check this parameter, e.g., with `prlctl list -i <VM_UUID|VM_name> | grep crash`, and reset it if required with `prlctl set <VM_UUID|VM_name> --on-crash restart`.

If conversion fails, the migrated VM is deleted from the destination server and you can try again. If the second attempt also fails, you need to enable a legacy VM debug mode on the destination Virtuozzo 7 server (see *Enabling Legacy VM Debug Mode* on page 195), make another migration attempt, send the problem report, and contact the technical support team.

With the debug mode enabled, the migrated VM will not be deleted from the destination Virtuozzo 7 server after conversion failure. It will remain stopped or running with disabled network to let the technical support team study the memory dump and find out the reason for failure.

2.13.4 Migrating Virtual Machine and Container Templates

Migrating virtual machine and container templates between Virtuozzo servers is similar to migrating virtual machines and containers offline.

- To migrate (move) the template `template1` to the remote server `remoteserver.com`, on the local server run:

  ```
  # prlctl migrate template1 root:passwd@remoteserver.com
  ```

- To migrate (move) the template `template1` from the remote server `remoteserver.com`, on the local server run:

  ```
  # prlctl migrate root:passwd@remoteserver.com/template1 localhost
  ```

If you do not provide the remote server credentials in the command, you will be asked to do so during migration.

Once migration is complete, the original template is removed from the source server (unless `--clone` is added).

2.13.5 Migrating EZ Templates

Unlike migrating VM and container templates which by default moves them between servers, migrating EZ templates always copies them to a remote server. If the source EZ template is not needed after migration, you can manually delete it with the `prlsrvctl cttemplate remove` command.

You can migrate OS and application EZ templates between Virtuozzo servers with the `prlsrvctl cttemplate copy` command. For example, to copy the OS EZ template `centos-7-x86-64` to the remote server `remoteserver.com`, on the local server run:

```
# prlsrvctl cttemplate copy root:passwd@remoteserver.com centos-7-x86-64
```

To copy an application EZ template, additionally specify the name of the corresponding OS EZ template.

Note: The specified OS template must be present on the destination server for migration to be successful.

For example, to copy the application template `mysql` of the OS template `centos-7-x86-64` to the remote server `remoteserver.com`, run:

```
# prlsrvctl cttemplate copy root:passwd@remoteserver.com mysql centos-7-x86-64
```

If you do not provide the destination server credentials in the command, you will be asked to do so during migration.

To skip all validation checks, indicate the `--force` option.

After migration, you can check that the migrated EZ templates are present on the destination server with the `prlsrvctl cttemplate list` command:

```
# prlsrvctl cttemplate list
centos-7-x86_64  os  x86_64 yes   Centos 7 (for Intel EM64T) Virtuozzo Template
...
mysql            app x86_64 -     mysql for Centos 7 (for Intel EM64T) Virtuozz
```

2.14 Performing Container-specific Operations

This section provides the description of operations specific to containers.

2.14.1 Reinstalling Containers

Reinstalling a container may help if any required container files have been inadvertently modified, replaced, or deleted, resulting in container malfunction. You can reinstall a container by using the `prlctl reinstall` command that creates a new container private area from scratch according to its configuration file and relevant OS and application templates. For example:

```
# prlctl reinstall MyCT
```

To keep the personal data from the old container, the utility also copies the old private area contents to the `/vz/root/<UUID>/old` directory of the new private area (unless the `--skipbackup` option is given). This directory may be deleted after you copy the personal data where you need.

The `prlctl reinstall` command retains user credentials base, unless the `--resetpwdb` option is specified.

2.14.1.1 Customizing Container Reinstallation

The default reinstallation, as performed by the `prlctl reinstall` command, creates a new private area for the broken container as if it were created by the `prlctl create` command and copies the private area of the broken container to the `/old` directory in the new private area so that no file is lost. There is also a possibility of deleting the old private area altogether without copying or mounting it inside the new private area, which is done by means of the `--skipbackup` option. This way of reinstalling corrupted containers might in certain cases not correspond exactly to your particular needs. It happens when you are accustomed to creating new containers in some other way than just using the `prlctl create` command. For example, you may install additional software licenses into new containers, or anything else. In this case you would naturally like to perform reinstallation in such a way so that the broken container is reverted to its original state as determined by you, and not by the default behavior of the `prlctl create` command.

To customize reinstallation, you should write your own scripts determining what should be done with the container when it is being reinstalled, and what should be configured inside the container after it has been reinstalled. These scripts should be named `vps.reinstall` and `vps.configure`, respectively, and should be located in the `/etc/vz/conf` directory on the hardware node. To facilitate your task of creating customized scripts, the containers software is shipped with sample scripts that you may use as the basis of your own scripts.

When the `prlctl reinstall <UUID>` command is called, it searches for the `vps.reinstall` and `vps.configure` scripts and launches them consecutively. When the `vps.reinstall` script is launched, the following parameters are passed to it:

Option	Description
`--veid`	Container UUID.
`--ve_private_tmp`	The path to the container temporary private area. This path designates where a new private area is temporarily created for the container. If the script runs successfully, this private area is mounted to the path of the original private area after the script has finished.
`--ve_private`	The path to the container original private area.

You may use these parameters within your `vps.reinstall` script.

If the `vps.reinstall` script finishes successfully, the container is started, and the `vps.configure` script is called. At this moment the old private area is mounted to the `/old` directory inside the new one irrespective of the `--skipbackup` option. This is done in order to let you use the necessary files from the old private area in your script, which is to be run inside the running container. For example, you might want to copy some files from there to regular container directories.

After the `vps.configure` script finishes, the old private area is either dismounted and deleted or remains mounted depending on whether the `--skipbackup` option was provided.

If you do not want to run these reinstallation scripts and want to stick to the default `prlctl reinstall` behavior, you may do either of the following:

- Remove the `vps.reinstall` and `vps.configure` scripts from the `/etc/vz/conf` directory, or at least rename them;
- Modify the last line of the `vps.reinstall` script so that it would read `exit 128` instead of `exit 0`.

The exit code `128` tells the utility not to run the scripts and to reinstall the container with the default behavior.

2.14.2 Enabling VPN for Containers

Virtual Private Network (VPN) is a technology which allows you to establish a secure network connection even over an insecure public network. Setting up a VPN for a separate container is possible via the TUN/TAP device. To allow a particular container to use this device, do the following:

1. Make sure the `tun.o` module is already loaded before Virtuozzo is started:

   ```
   # lsmod | grep 'tun'
   ```

2. Allow the container to use the TUN/TAP device:

```
# vzctl set MyCT --devnodes net/tun:rw --save
```

Configuring the VPN properly is a common Linux administration task, which is out of the scope of this guide. Some popular Linux software for setting up a VPN over the TUN/TAP driver includes Virtual TUNnel and OpenVPN.

2.14.3 Setting Up NFS Server in Containers

To set up an NFS server in a container, do the following:

1. Make sure the `rpcbind`, `nfsd`, and `nfslock` services are installed in the container.
2. Enable the NFS server feature for the container by running the `prlctl set --features nfsd:on` command on the hardware node. For example:

   ```
   # prlctl set MyCT --features nfsd:on
   ```

   If the container is running, stop it first. After enabling the feature, restart the container.

   Note: You cannot perform live migration or create snapshots of containers with enabled NFS server feature.

3. Start the `rpcbind` service in the container.

   ```
   # service rpcbind start
   Starting rpcbind:                                          [  OK  ]
   ```

4. Start the `nfs` and `nfslock` services in the container.

   ```
   # service nfs start
   Starting NFS services:                                     [  OK  ]
   Starting NFS quotas:                                       [  OK  ]
   Starting NFS mountd:                                       [  OK  ]
   Starting NFS daemon:                                       [  OK  ]
   # service nfslock start
   Starting NFS statd:                                        [  OK  ]
   ```

You can now set up NFS shares in the configured container.

2.14.4 Mounting NFS Shares on Container Start

If you configured an NFS share in the `/etc/fstab` file of a CentOS or RHEL-based container, and you need this NFS share to be mounted on container start, enable autostart for the `netfs` service with the `chkconfig netfs` `on` command.

2.14.5 Managing Container Virtual Disks

Note: You can manage virtual disks of both stopped and running containers.

Virtuozzo allows you to perform the following operations on container virtual disks:

- add new virtual disks to containers,
- configure the virtual disk properties,
- remove virtual disks from containers.

2.14.5.1 Adding Virtual Disks to Containers

New containers are created with a single virtual hard disk, but you can add more disks as follows:

- attach a new or existing image file that emulates a hard disk drive, or
- attach a physical hard disk of the host server.

Using Image Files

You can either attach an existing image to the container or create a new one and keep it at a custom location, e.g., on a regular disk or in a Virtuozzo Storage cluster. This allows creating more flexible containers, in which the operating system may be kept on a fast SSD and user data may be stored on redundant Virtuozzo Storage.

To create a new image file and add it to a container as a virtual hard disk, use the `prlctl set --device-add hdd` command. For example:

```
# prlctl set MyCT --device-add hdd --size 100G --mnt /userdisk
```

Note: If you omit the `--mnt` option, the disk will be added unmounted.

This command adds to the configuration of the container `MyCT` a virtual hard disk with the following parameters:

- name: `hdd<N>` where `<N>` is the next available disk index,
- default image location: `/vz/private/<CT_UUID>/harddisk<N>.hdd` where `<N>` is the next available disk index,
- size: 102400 MB,
- mount point inside the container `MyCT`: `/userdisk`. A corresponding entry is also added to container's `/etc/fstab` file.

To attach an existing image file to a container as a virtual hard disk, specify the path to the image file with the `--image` option. For example:

```
# prlctl set MyCT --device-add hdd --image /hdd/MyCT.hdd --size 100G --mnt /userdisk
```

Attaching Physical Hard Disks

You can attach to a container any physical block device available on the physical server, whether it is a local hard disk or an external device connected via Fibre Channel or iSCSI.

Note: A physical block device must be formatted and have only one filesystem before it can be attached to a container.

You will need to specify the path to the device, which you can find out the with the `prlsrvctl info` command. For example:

```
# prlsrvctl info
...
Hardware info:
     hdd  WDC WD1002FAEX-0 ATA (/dev/sda2)         '/dev/disk/by-id/lvm-pv-uuid-RDYrbU-YZsH-uS8w-\
                                                    aH0t-EH9W-6dir-ea9lDL'
     hdd  WDC WD1002FAEX-0 ATA (/dev/sda)          '/dev/disk/by-id/wwn-0x50014ee25a3df4dc'
   cdrom  PIONEER DVD-RW  DVR-220                  '/dev/sr0'
     net  eth0                                     'eth0'
  serial  /dev/ttyS0                               '/dev/ttyS0'
...
```

All physical devices available on the physical server are listed in the **Hardware info** section.

Once you know the path to the physical block device, you can attach it to a container with the `prlctl set --device-add hdd --device` command. For example:

```
# prlctl set MyCT --device-add hdd --device '/dev/disk/by-id/wwn-0x50014ee25a3df4dc' \
--mnt /userdisk
```

Note: If you omit the `--mnt` option, the disk will be added unmounted.

This command adds to the configuration of the container `MyCT` a virtual hard disk with the following parameters:

- name: `hdd<N>` where `<N>` is the next available disk index,
- path to the device: `/dev/disk/by-id/wwn-0x50014ee25a3df4dc` where `wwn-0x50014ee25a3df4dc` is a storage device unique identifier,
- mount point inside the container: `/userdisk`. A corresponding entry is also added to container's `/etc/fstab` file.

Note:

1. Before migrating containers with external hard drives, make sure that corresponding physical disks exist on the destination server and are available by the same name (for this purpose, use persistent naming, for example, via `/dev/disk/by-id/`),
2. During container backup operations, physical disks connected to the container are not backed up.
3. If you use multipath from a system to a device, it is recommended to use the `user_friendly_names no` feature so that multipath devices have names persistent across all nodes in a cluster.

2.14.5.2 Configuring Container Virtual Disks

To configure the parameters of a virtual disk attached to a container, use the `prlctl set --device-set` command.

You will need to specify the disk name, which you can find out the with the `prlctl list -i` command. For example:

```
# prlctl list -i MyCT | grep "hdd"
hdd0 (+) scsi:0 image='/vz/private/9fd3eee7-70fe-43e3-9295-1ab29fe6dba5/root.hdd' type='expanded' \
10240Mb mnt=/ subtype=virtio-scsi
hdd1 (+) scsi:1 real='/dev/disk/by-id/wwn-0x50014ee25a3df4dc' mnt=/userdisk subtype=virtio-scsi
```

Once you know the virtual device name, you can configure its properties. For example, to change the type of the virtual disk `hdd0` in the container `MyCT` from SCSI to IDE, execute:

```
# prlctl set MyCT --device-set hdd0 --iface ide
```

To check that the virtual disk type has been changed, use the `prlctl list -i` command. For example:

```
# prlctl list -i MyCT | grep "hdd"
hdd0 (+) ide:0 image='/vz/private/9fd3eee7-70fe-43e3-9295-1ab29fe6dba5/root.hdd' type='expanded' \
10240Mb mnt=/
```

2.14.5.3 Deleting Virtual Disks from Containers

You can delete a virtual hard disk from a container with the `prlctl set --device-del` command.

You will need to specify the disk name, which you can find out the with the `prlctl list -i` command. For example:

```
# prlctl list -i MyCT | grep "hdd"
hdd0 (+) scsi:0 image='/vz/private/9fd3eee7-70fe-43e3-9295-1ab29fe6dba5/root.hdd' type='expanded' \
10240Mb mnt=/ subtype=virtio-scsi
hdd1 (+) scsi:1 real='/dev/disk/by-id/wwn-0x50014ee25a3df4dc' mnt=/userdisk subtype=virtio-scsi
```

Once you know the virtual device name, you can remove it from your container. For example, to remove the virtual disk `hdd1` from the container `MyCT`, execute:

```
# prlctl set MyCT --device-del hdd1
```

2.14.6 Restarting Containers

You can restart containers from the inside using typical Linux commands, e.g., `reboot` or `shutdown -r`. Restarting is handled by the `vzeventd` daemon.

If necessary, you can forbid restarting containers from the inside as follows:

- To disable restarting for a specific container, add the `ALLOWREBOOT="no"` line to the container configuration file (`/etc/vz/conf/<UUID>.conf`).
- To disable restarting globally for all containers on the server, add the `ALLOWREBOOT="no"` line to the global configuration file (`/etc/vz/vz.conf`).
- To disable restarting globally except for specific containers, add the `ALLOWREBOOT="no"` line to the global configuration file (`/etc/vz/vz.conf`) and explicitly specify `ALLOWREBOOT="yes"` in the configuration files of the respective containers.

2.14.7 Creating SimFS-based Containers

In Virtuozzo 7, the simfs layout is based on bindmounts. When a simfs-based container is started, its private area is bindmounted to the root container area.

To create a simfs container:

1. Set `VEFSTYPE=simfs` in `/etc/vz/vz.conf`.
2. Run `prlctl create <CT_name>`.

The limitations of simfs in Virtuozzo 7 are:

1. No support for first- or second-level quotas.
2. No support for live migration of simfs-based containers.

2.15 Performing Virtual Machine-specific Operations

This section focuses on operations specific to virtual machines.

2.15.1 Pausing Virtual Machines

Pausing a running virtual machine releases the resources, such as RAM and CPU, currently used by this virtual machine. The released resources can then be used by the hardware node or other running virtual machines and containers.

To pause a virtual machine, you can use the `prlctl pause` command. For example, the following command pauses the virtual machine `MyVM`:

```
# prlctl pause MyVM
Pause the VM...
The VM has been successfully paused.
```

You can check that the virtual machine has been successfully paused by using the `prlctl list -a` command:

```
# prlctl list -a
STATUS  IP_ADDR         NAME
running 10.10.10.101    MyCT
```

```
paused  10.10.10.201    MyVM
```

The command output shows that the virtual machine `MyVM` is paused at the moment. To continue running this virtual machine, execute this command:

```
# prlctl start MyVM
Starting the VM...
The VM has been successfully started.
```

2.15.2 Managing Virtual Machine Devices

Virtuozzo allows you to manage the following virtual machine devices:

- hard disk drives
- CD/DVD-ROM drives
- floppy disk drives
- network adapters
- serial ports
- USB controllers

The main operations you can perform on these devices are:

- adding a new device to the virtual machine
- configuring the device properties
- removing a device from the virtual machine

2.15.2.1 Adding New Devices

This section provides information on adding new devices to your virtual machines. You can add new virtual devices to your virtual machine using the `prlctl set` command. The options responsible for adding particular devices are listed in the following table:

Option	Description
`hdd`	Adds a new hard disk drive to the virtual machine. You can either connect an existing image to the virtual machine or create a new one. **Note:** SCSI and VirtIO hard disks can be added to both running and stopped VMs, IDE disks can only be added to stopped VMs.
`cdrom`	Adds a new CD/DVD-ROM drive to the virtual machine.
`net`	Adds a new network adapter to the virtual machine.
`fdd`	Adds a new floppy disk drive to the virtual machine.
`serial`	Adds a new serial port to the virtual machine.
`usb`	Adds a new USB controller to the virtual machine.

For example, you can execute the following command to add a new virtual disk to the virtual machine `MyVM`:

```
# prlctl set MyVM --device-add hdd
Creating hdd1 () scsi:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' type='expanded' 65536Mb subtype=virtio-scsi
Created hdd1 () scsi:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' type='expanded' subtype=virtio-scsi
The VM has been successfully configured.
```

This command creates a new virtual disk with the following default parameters:

- name: `hdd1`
- disk type: SCSI
- disk subtype: VirtIO SCSI
- image file name and location: `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk1.hdd`
- disk format: expanded
- disk capacity: 65536 MB

You can redefine some of these parameters by specifying specific options during the command execution. For example, to create an IDE virtual disk that will have the capacity of 84 GB, you can run this command:

```
# prlctl set MyVM --device-add hdd --size 84000 --iface ide
Creating hdd2 () ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk2.hdd' type='expanded' 84000Mb
Created hdd2 () ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk2.hdd' type='expanded'
```

```
The VM has been successfully configured.
```

The virtual disk has been added to your virtual machine. However, before starting to use it, you must initialize the disk. Refer to the next subsection for information on how you can do it.

When managing devices, keep in mind the following:

- To a virtual machine, you can connect up to:
    - 4 IDE devices (virtual disks or CD/DVD-ROM drives),
    - 15 SCSI devices (virtual disks or CD/DVD-ROM drives),
    - 15 VirtIO virtual disks.
- A virtual machine can have up to 16 virtual network adapters.
- A virtual machine can have up to 4 serial ports.
- A virtual machine can have only 1 USB controller.
- A virtual machine can have only 1 floppy disk drive.

Adding Hyper-V SCSI Devices to Windows Virtual Machines

Virtuozzo supports Hyper-V paravirtualized device emulation of SCSI hard disks and CD/DVD-ROM drives. The emulation allows to use these devices with native Windows drivers. It is supported and recommended for the following Windows operating systems:

- Windows 10 (x64),
- Windows Server 2016,
- Windows Server 2012 R2,
- Windows Server 2012,
- Windows Server 2008 R2 with Service Pack 2.

To add, for example, a SCSI CD/DVD-ROM and HDD emulated via Hyper-V to a VM, run the following commands:

```
# prlctl set vm1 --device-add cdrom --image <path_to_image> --iface scsi --subtype hyperv
# prlctl set vm1 --device-add hdd --iface scsi --subtype hyperv
```

2.15.2.2 Initializing Newly Added Disks

After you added a new blank virtual hard disk to the virtual machine configuration, it will be invisible to the operating system installed inside the virtual machine until the moment you initialize it.

Initializing the New Virtual Hard Disk in Windows

To initialize a new virtual hard disk in a Windows guest OS, you will need the Disk Management utility available. For example, in Windows Server 2012 you can access this utility by clicking Start > Control Panel > System and Security > Administrative Tools > Computer Management > Storage > Disk Management.

When you open the Disk Management utility, it automatically detects that a new hard disk was added to the configuration and launches the Initialize Disk wizard:

1. In the Select disks section, select the newly added disk.
2. Choose the partition style for the selected disk: MBR (Master Boot Record) or GPD (GUID Partition Table).
3. Click OK.

The added disk will appear as a new disk in the Disk Management utility window, but its memory space will be unallocated. To allocate the disk memory, right-click this disk name in the Disk Management utility window and select New Volume. The New Volume Wizard window will appear. Follow the steps of the wizard and create a new volume in the newly added disk.

After that your disk will become visible in My Computer and you will be able to use it as a data disk inside your virtual machine.

Initializing the New Virtual Hard Disk in Linux

Initializing a new virtual hard disk in a Linux guest OS comprises two steps: (1) allocating the virtual hard disk space and (2) mounting this disk in the guest OS.

To allocate the space, you need to create a new partition on this virtual hard disk using the fdisk utility:

Note: To use the `fdisk` utility, you need the `root` privileges.

1. Launch a terminal window.

2. To list the IDE disk devices present in your virtual machine configuration, enter:

   ```
   fdisk /dev/hd*
   ```

   Note: If you added a SCSI disk to the virtual machine configuration, use the `fdisk /dev/sd*` command instead.

3. By default, the second virtual hard disk appears as `/dev/hdc` in your Linux virtual machine. To work with this device, enter:

   ```
   fdisk /dev/hdc
   ```

   Note: If this is a SCSI disk, use the `fdisk /dev/sdc` command instead.

4. To get detailed information about the disk, enter:

   ```
   p
   ```

5. To create a new partition, enter:

   ```
   n
   ```

6. To create the primary partition, enter:

   ```
   p
   ```

7. Specify the partition number. By default, it is 1.

8. Specify the first cylinder. If you want to create a single partition on this hard disk, use the default value.

9. Specify the last cylinder. If you want to create a single partition on this hard disk, use the default value.

10. To create a partition with the specified settings, enter:

    ```
    w
    ```

When you allocated the space on the newly added virtual hard disk, you should format it by entering the following command in the terminal:

```
# mkfs -t <FileSystem> /dev/hdc1
```

Note: `<FileSystem>` stands for the filesystem you want to use on this disk. It is recommended to use `ext4`.

When the added virtual hard disk is formatted, you can mount it in the guest OS.

1. To create a mount point for the new virtual hard disk, enter:

   ```
   # mkdir /mnt/hdc1
   ```

   Note: You can specify a different mount point.

2. To mount the new virtual hard disk to the specified mount point, enter:

   ```
   mount /dev/hdc1 /mnt/hdc1
   ```

When you mounted the virtual hard disk, you can use its space in your virtual machine.

2.15.2.3 Configuring Virtual Devices

In Virtuozzo, you can use the `--device-set` option of the `prlctl set` command to configure the parameters of an existing virtual device. As a rule, the process of configuring the device properties includes two steps:

1. Finding out the name of the device you want to configure.
2. Running the `prlctl set` command to configure the necessary device properties.

Finding Out Device Names

To configure a virtual device, you need to specify its name when running the `prlctl set` command. If you do not know the device name, you can use the `prlctl list` command to learn it. For example, to obtain the list of virtual devices in the virtual machine `MyVM`, run this command:

```
# prlctl list --info MyVM
...
Hardware:
  cpu cpus=2 VT-x accl=high mode=32 ioprio=4 iolimit='0'
  memory 1024Mb
  video 32Mb 3d acceleration=off vertical sync=yes
  hdd0 (+) scsi:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk.hdd' type='expanded' subtype=virtio-scsi
  hdd1 (+) scsi:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' type='expanded' subtype=virtio-scsi
  cdrom0 (+) scsi:1 image='' subtype=virtio-scsi
  usb (+)
  net0 (+) dev='vme426f6594' network='Bridged' mac=001C426F6594 card=virtio
...
```

All virtual devices currently available to the virtual machine are listed under `Hardware`. In our case the virtual machine `MyVM` has the following devices: 2 CPUs, main memory, video memory, a floppy disk drive, 2 hard disk drives, a CD/DVD-ROM drive, a USB controller, and a network card.

Configuring Virtual Device Properties

Once you know the virtual device name, you can configure its properties. For example, you can execute the following command to configure the current type of the virtual disk `hdd1` in the virtual machine `MyVM` from SCSI to IDE:

```
# prlctl set MyVM --device-set hdd1 --iface ide
Configured hdd1 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' type='expanded'
The VM has been successfully configured.
```

To check that the virtual disk type has been successfully changed, use the `prlctl list --info` command:

```
# prlctl list --info MyVM
...
hdd1 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' type='expanded'
...
```

Connecting and Disconnecting Virtual Devices

In Virtuozzo, you can connect or disconnect certain devices when a virtual machine is running. These devices include:

- CD/DVD-ROM drives
- floppy disk drives
- network adapters
- printer ports
- serial ports

Usually, all virtual devices are automatically connected to a virtual machine when you create them. To disconnect a device from the virtual machine, you can use the `prlctl set` command. For example, the following command disconnects the CD/DVD-ROM drive `cdrom0` from the virtual machine `MyVM`:

```
# prlctl set MyVM --device-disconnect cdrom0
Disconnect device: cdrom0
The VM has been successfully configured.
```

To connect the CD/DVD-ROM drive back, you can run the following command:

```
# prlctl set MyVM --device-connect cdrom0
Connect device: cdrom0
```

```
The VM has been successfully configured.
```

2.15.2.4 Deleting Devices

You can delete a virtual device that you do not need any more in your virtual machine using the `--device-del` option of the `prlctl set` command. The options responsible for removing particular devices are listed in the following table:

Option	Description
`hdd`	Deletes the specified hard disk drive from the virtual machine. **Note:** Hard disks can be removed only from stopped virtual machines.
`cdrom`	Deletes the specified CD/DVD-ROM drive from the virtual machine.
`net`	Deletes the specified network adapter from the virtual machine.
`fdd`	Deletes the floppy disk drive from the virtual machine.
`serial`	Deletes the specified serial port from the virtual machine.
`usb`	Deletes the USB controller from the virtual machine.

As a rule deleting a virtual device involves performing two operations:

1. Finding out the name of the device to be deleted.
2. Deleting the device from the virtual machine.

Finding Out the Device Name

To remove a virtual device, you need to specify its name when running the `prlctl set` command. If you do not know the device name, you can use the `prlctl list` command to learn it (for details, see *Finding Out Device Names* on page 65).

Deleting a Virtual Device

Once you know the virtual device name, you can remove it from your virtual machine. For example, you can execute the following command to remove the virtual disk `hdd1` from the virtual machine `MyVM`:

```
# prlctl set MyVM --device-del hdd1
Remove the hdd1 device.
The VM has been successfully configured.
```

If you do not want to permanently delete a virtual device, you can temporarily disconnect it from the virtual machine using the `--disable` option.

2.15.3 Making Screenshots

When a virtual machine stops responding to requests, you can check its state by capturing an image (or screenshot) of its screen with the `prlctl capture` command. A screenshot is saved in PNG format.

Note: You can take screenshots of running virtual machines only.

To take a screenshot of the `MyVM` virtual machine screen and save it to the `/usr/screenshots/image1.png` file:

1. Make sure that the virtual machine is running:

```
# prlctl list
UUID                                    STATUS    IP_ADDR       T   NAME
{b2de86d9-6539-4ccc-9120-928b33ed31b9}  running   10.10.100.1   VM  MyVM
```

2. Take the virtual machine screenshot:

```
# prlctl capture MyVM --file /usr/screenshots/image1.png
```

If the `--file` option is not specified, a screenshot is dumped to the command output.

3. Check that the `image1.png` file has been successfully created:

```
# ls /usr/screenshots/
image1.png
```

2.15.4 Configuring IP Address Ranges for Host-Only Networks

All virtual machines connected to networks of the host-only type receive their IP addresses from the DHCP server. This DHCP server is set up during the Virtuozzo installation and includes by default IP addresses from 10.37.130.1 to 10.37.130.254. You can redefine the default IP address range for host-only networks and make virtual machines get their IP addresses from different IP address ranges. For example, you can run the following command to set the start and end IP addresses for the `Host-Only` network (this network is automatically created during the Virtuozzo installation) to 10.10.11.1 and 10.10.11.254, respectively:

```
# prlsrvctl net set Host-Only --ip-scope-start 10.10.11.1 --ip-scope-end 10.10.11.254
```

You can also specify a custom IP address range directly when creating a new network of the host-only type. Assuming that you want to create a network with the `Host-Only2` name and define for this network the IP addresses range from 10.10.10.1 to 10.10.10.254, you can execute the following command:

```
# prlsrvctl net add Host-Only2 -t host-only --ip-scope-start 10.10.10.1 --ip-scope-\
end 10.10.10.254
```

When working with IP address ranges, pay attention to the following:

- The start and end IP addresses of an IP address range must belong to the same subnetwork.
- IP address ranges can be defined for each network of the host-only type separately. For example, you can set the IP address range from 10.10.11.1 to 10.10.11.254 for the `Host-Only` network and from 10.10.10.1 to 10.10.10.254 for the `Host-Only2` network.

2.15.5 Configuring Virtual Machine Crash Mode

In Virtuozzo 7, you can configure a virtual machine behavior after the guest OS crash: restart or pause. By default, when a virtual machine fails, a crash dump is created and sent in the problem report to the Virtuozzo technical support team, and the virtual machine is restarted with the same configuration.

To address the problem yourself, you can switch the virtual machine crash mode to `pause` using the `prlctl set` command. For example:

```
# prlctl set MyVM --on-crash pause
```

The virtual machine resources will be preserved to allow analysis and its crash dump will be sent in the problem report.

As crash dumps can take up significant disk space and lead to the whole server malfunction, the crash mode is automatically switched to `pause`, if a virtual machine fails more than three times within twenty-four hours since the last crash.

If you want to skip creating the crash dump and sending the problem report, add `:no-report` to the command. For example:

```
# prlctl set MyVM --on-crash restart:no-report
```

CHAPTER 3

Managing Resources

The main goal of resource control in Virtuozzo is to provide Service Level Management or Quality of Service for virtual machines and containers. Correctly configured resource control settings prevent serious impacts resulting from the resource over-usage (accidental or malicious) of any virtual machine or container on the other virtual machines and containers. Using resource control parameters for resource management also allows you to enforce fairness of resource usage among virtual machines and containers and better service quality for preferred virtual machines and containers, if necessary. All these parameters can be set using command-line utilities.

3.1 Managing CPU Resources

You can manage the following CPU resource parameters for virtual machines and containers:

- CPU units for virtual machines and containers
- CPU affinity for virtual machines and containers
- CPU limits for virtual machines and containers
- NUMA nodes for virtual machines and containers
- CPU hotplug for virtual machines
- CPU topology for virtual machines

Detailed information on these parameters is given in the following sections.

3.1.1 Configuring CPU Units

CPU units define how much CPU time one virtual machine or container can receive in comparison with the other virtual machines and containers on the hardware node if all the CPUs of the hardware node are fully used. For example, if the container `MyCT` and the virtual machine `MyVM` are set to receive 1000 CPU units each and the container `MyCT2` is configured to get 2000 CPU units, the container `MyCT2` will get twice as much CPU time as the container `MyCT` or the virtual machine `MyVM` if all the CPUs of the Node are completely loaded.

By default, each virtual machine and container on the Node gets 1000 CPU units. You can configure the default setting using the `prlctl set` command. For example, you can run the following commands to allocate 2000 CPU units to the container `MyCT` and the virtual machine `MyVM`:

```
# prlctl set MyCT --cpuunits 2000
# prlctl set MyVM --cpuunits 2000
```

3.1.2 Configuring CPU Affinity for Virtual Machines and Containers

If your physical server has several CPUs installed, you can bind a virtual machine or container to specific CPUs so that only these CPUs are used to handle the processes running in the virtual machine or container. The feature of binding certain processes to certain CPUs is known as CPU affinity.

By default, any newly created virtual machine or container can consume the CPU time of all processors installed on the physical server. To bind a virtual machine or container to specific CPUs, you can use the `--cpumask` option of the `prlctl set` command. Assuming that your physical server has 8 CPUs, you can make the processes in the virtual machine `MyVM` and the container `MyCT` run on CPUs 0, 1, 3, 4, 5, and 6 by running the following commands:

```
# prlctl set MyVM --cpumask 0,1,3,4-6
# prlctl set MyCT --cpumask 0,1,3,4-6
```

You can specify the CPU affinity mask—that is, the processors to bind to virtual machines and containers—as separate CPU index numbers (0,1,3) or as CPU ranges (4-6). If you are setting the CPU affinity mask for a running virtual machine or container, the changes are applied on the fly.

To undo the changes made to the virtual machine `MyVM` and the container `MyCT` and set their processes to run on all available CPUs on the server, run these commands:

```
# prlctl set MyVM --cpumask all
# prlctl set MyCT --cpumask all
```

3.1.3 Configuring CPU Limits for Virtual Machines and Containers

A CPU limit indicates the maximum CPU power a virtual machine or container may get for its running processes. The container is not allowed to exceed the specified limit even if the server has enough free CPU power. By default, the CPU limit parameter is disabled for all newly created virtual machines and containers. This means that any application in any virtual machine or container can use all the free CPU power of the server.

Note: You can change which virtual machine threads—both service and activity or only activity—are limited by the parameters described below. To do this, enter the `prlsrvctl set --vm-cpulimit-type <full|guest>` command and restart running virtual machines for the changes to take effect.

To set a CPU limit for a virtual machine or container, you can use one of these options: `--cpulimit`, `--cpus`. Both options are described below in detail.

3.1.3.1 Using --cpulimit to Set CPU Limits

As a rule, you set a CPU limit for a virtual machine or container by using the `prlctl set --cpulimit` command. In the following example, the container `MyCT` is set to receive no more than 25% of the server CPU time even if the CPUs on the server are not fully loaded:

```
# prlctl set MyCT --cpulimit 25
```

This command sets the CPU limit for the container `MyCT` to 25% of the total CPU power of the server. The total CPU power of a server in per cent is calculated by multiplying the number of logical CPU cores installed on the server by 100%. So if a server has 2 logical CPU cores, 2 GHz each, the total CPU power will equal 200% and the limit for the container `MyCT` will be set to 500 MHz.

For example, on a hardware node with 2 logical CPU cores, 3 GHz each, the container `MyCT` will be able to get 25% of 6 GHz, that is, 750 MHz. To ensure that the container `MyCT` always has the same CPU limit on all servers, irrespective of their total CPU power, you can set the CPU limits in megahertz (MHz). For example, to make the container `MyCT` consume no more than 500 MHz on any hardware node, run the following command:

```
# prlctl set MyCT --cpulimit 500m
```

Note: For more information on setting CPU limits for virtual machines and containers, see also *CPU Limit Specifics* on page 73.

3.1.3.2 Using --cpus to Set CPU Limits

Another way of setting a CPU limit for a virtual machine or container is to use the `prlctl set --cpus` command. In this case, you can specify how many logical CPU cores per CPU socket the virtual machine or container may use. For example, as containers have only one CPU socket (for details, see *Configuring CPU Topology for Virtual Machines* on page 76), you can allow the container `MyCT` to use only 2 cores by running this command:

```
# prlctl set MyCT --cpus 2
```

To make sure that the CPU limit has been successfully set, you check `/proc/cpuinfo` in the container. For example:

```
# prlctl exec MyCT cat /proc/cpuinfo | grep "cpu cores"
cpu cores       : 2
```

3.1.3.3 Using --cpulimit and --cpus Simultaneously

If you use both `--cpulimit` and `--cpus` to set the CPU limit for a virtual machine or container, the smallest limit applies. For example, running the following commands on a server with 4 CPUs, 2 GHz each, will set the limit for the container `MyCT` to 2 GHz:

```
# prlctl set MyCT --cpus 2
# prlctl set MyCT --cpulimit 2000m
```

3.1.3.4 CPU Limit Specifics

Internally, Virtuozzo sets the CPU limit for virtual machines and containers in percent. On multi-core systems, each logical CPU core is considered to have the CPU power of 100%. So if a server has 4 CPU cores, the total CPU power of the server equals 400%.

You can also set a CPU limit in megahertz (MHz). If you specify the limit in MHz, Virtuozzo uses the following formula to convert the CPU power of the server from MHz into percent: `CPULIMIT_% = 100% * CPULIMIT_MHz / CPUFREQ`, where

- `CPULIMIT_%` is the total CPU power of the server in percent.

- `CPULIMIT_MHz` is the total CPU power of the server in megahertz.
- `CPUFREQ` is the CPU frequency of one core on the server.

When setting CPU limits, note the following:

- Make sure that the CPU limit you plan to set for a virtual machine or container does not exceed the total CPU power of the server. So if a server has 4 CPUs, 1000 MHz each, do not set the CPU limit to more than 4000 MHz.
- The processes running in a virtual machine or container are scheduled for execution on all server CPUs in equal shares. For example, if a server has 4 CPUs, 1000 MHz each, and you set the CPU limit for a virtual machine or container to 2000 MHz, the virtual machine or container will consume 500 MHz from each CPU.
- All running virtual machines and containers on a server cannot simultaneously consume more CPU power than is physically available on the node. In other words, if the total CPU power of the server is 4000 MHz, the running virtual machines and containers on this server will not be able to consume more than 4000 MHz, irrespective of their CPU limits. It is, however, perfectly normal that the overall CPU limit of all virtual machines and containers exceeds the Node total CPU power because most of the time virtual machines and containers consume only part of the CPU power assigned to them.

3.1.4 Binding CPUs to NUMA Nodes

On systems with a NUMA (Non-Uniform Memory Access) architecture, you can configure virtual machines and containers to use CPUs from specific NUMA nodes only. Consider the following example:

- Your physical server has 8 CPUs installed.
- The CPUs are divided into 2 NUMA nodes: NUMA node 0 and NUMA node 1. Each NUMA node has 4 CPUs.
- You want the processes in the container `MyCT` to be executed on the processors from NUMA node 1.

To set the container `MyCT` to use the processors from NUMA node 1, run the following command:

```
# prlctl set MyCT --nodemask 1
```

To check that the container `MyCT` is now bound to NUMA node 1, use this command:

```
# prlctl list -i MyCT | grep nodemask
  cpu cpus=unlimited VT-x hotplug accl=high mode=32 cpuunits=1000 ioprio=4 nodemask=1
```

To unbind the container `MyCT` from NUMA node 1, execute this command:

```
# prlctl set MyCT --nodemask all
```

Now the container `MyCT` should be able to use all CPUs on the server again.

Note: For more information on NUMA, visit http://lse.sourceforge.net/numa.

3.1.5 Enabling CPU Hotplug for Virtual Machines

If a guest operating system supports the CPU hotplug functionality, you can enable this functionality for the virtual machine. Once the CPU hotplug functionality is turned on, you can increase the number of CPUs available to your virtual machines even if they are running.

Currently, the following systems come with the CPU hotplug support:

- Linux operating systems based on the RHEL 5 kernel and higher (Red Hat Linux Enterprise 5, CentOS 5, and so on)
- x64 version of Windows Server 2008 R2 (Datacenter Edition)
- x64 version of Windows Server 2012 (Standard and Datacenter Edition)
- x64 version of Windows Server 2008 (Standard Edition)
- x64 version of Windows Server 2008 (Enterprise Edition)
- x64 version of Windows Server 2008 (Datacenter Edition)

By default, the CPU hotplug support is disabled for all newly created virtual machines. To enable this functionality, you can use the `--cpu-hotplug` option of the `prlctl set` command. For example, to enable the CPU hotplug support in the virtual machine `MyVM` that runs one of the supported operating systems, stop the virtual machine `MyVM` and run this command:

```
# prlctl set MyVM --cpu-hotplug on
set cpu hotplug: 1
The VM has been successfully configured.
```

Once the functionality is enabled, you can increase the number of CPUs in the virtual machine `MyVM` even it is running. Assuming that your physical server has 4 CPUs installed and the processes in the virtual machine `MyVM` are set to be executed on two CPUs, you can run the following command to assign 3 CPUs to the virtual machine:

```
# prlctl set MyVM --cpus 3
set cpus(4): 3
```

```
The VM has been successfully configured.
```

To disable the CPU hotplug support in the virtual machine `MyVM`, use this command:

```
# prlctl set MyVM --cpu-hotplug off
set cpu hotplug: 0
The VM has been successfully configured.
```

The changes will come into effect on the next virtual machine start.

3.1.6 Configuring CPU Topology for Virtual Machines

In the current version of Virtuozzo, you can specify CPU topology for virtual machines, i.e. the number of CPU sockets and CPU cores per socket. This can be efficient, for example, if you use a guest operating system that supports a specific number of CPU sockets (e.g., limited by a license). In such a case, you can set the VM to use as many CPU cores as allowed by the guest OS and achieve near-native guest OS performance. The overall number of CPU cores available to a virtual machine is calculated by multiplying the number of CPU sockets by the number of CPU cores per socket. It can be no greater than the number of CPU cores on the host physical server.

By default, a virtual machine is created with one CPU socket and two CPU cores.

Note: In turn, a container has only one CPU socket, and this parameter cannot be changed.

You can change the number of CPU sockets and CPU cores per socket of a virtual machine using the `--cpu-sockets` and `--cpus` options of the `prlctl set` command. For example, if your physical server has 4 CPU cores, you can allow the virtual machine `MyVM` to use 2 CPU sockets and 2 CPU cores per socket by running the following command:

```
# prlctl set MyVM --cpu-sockets 2 --cpus 2
```

If a virtual machine is running, the changes will take effect after it is restarted.

To check the current CPU topology of a VM, run the following command:

```
# prlctl list MyVM -i | grep "cpu"
cpu sockets=2 cpus=2 cores=2 <...>
```

3.2 Managing Disk Quotas

You can limit disk space that individual users and groups in a container can use with standard Linux tools from the `quota` package.

Before you can set disk quotas in a container, you will need to enable them for this container as follows:

1. Set `QUOTAUGIDLIMIT` to 1 in container configuration file (`/etc/vz/conf/<UUID>.conf`) or run the command `prlctl set <UUID> --quotaugidlimit 1`.
2. Restart the container.

3.3 Managing Virtual Disks

In Virtuozzo, you can manage virtual disks as follows:

- resize,
- compact (reduce their size on the physical hard drive),
- change interface.

All these operations are described in the following subsections in detail.

3.3.1 Resizing Virtual Disks

Warning: Only use `prl_disk_tool` on disks of stopped virtual machines.

You can resize virtual hard disks of your virtual machines or containers with the `prl_disk_tool resize --size` command. For example:

```
# prl_disk_tool resize --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk.hdd --size 30G
```

When resizing virtual disks, keep in mind the following:

- The virtual machine that uses the virtual disk to be resized must not have any snapshots.
- The virtual disk size shown inside the virtual machine or container may differ from the size the virtual

disk occupies on server's physical disk.

- In case disk size is increased, the added disk space is added as unallocated. You can use standard tools of the guest OS to allocate the added space.
- You cannot reduce XFS filesystems (the default choice for CentOS 7 and Red Hat Enterprise Linux 7).

3.3.1.1 Checking the Minimum Disk Capacity

If, before reducing disk capacity, you want to know the minimum size to which it can be reduced, use the `prl_disk_tool resize --info` command. For example, if the disk `hdd0` of the virtual machine `MyVM` is emulated by the image `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk.hdd`, run the following command:

```
# prl_disk_tool resize --info --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065- \
8fef6178bc5b/harddisk.hdd
Disk information:
...
Minimum: 2338M
...
```

3.3.2 Compacting Disks

Warning: Only use `prl_disk_tool` on disks of stopped virtual machines.

In Virtuozzo, you can reduce the space your virtual machines and containers occupy on the physical server's disk drive by compacting their virtual disks. Doing so frees up server disk space for hosting more virtual machines and containers.

To compact a virtual disk, you can use the `prl_disk_tool compact` command. For example, to compact the disk `/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/harddisk.hdd`, run this command:

```
# prl_disk_tool compact --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk.hdd/
```

To check the space that was freed by compacting the virtual disk, you can use standard Linux utilities (for example, `df`).

3.3.3 Managing Virtual Machine Disk Interfaces

By default, any virtual machine is created with a SCSI virtual hard disk. If necessary, you can change the interface type of a disk from SCSI to IDE or VirtIO. For example, to change the interface type of the default disk (`hdd0`) in the virtual machine `MyVM` from SCSI to IDE, you can run the following command:

```
# prlctl set MyVM --device-set hdd0 --iface ide
The VM has been successfully configured
```

To check that the interface type has been successfully changed, use this command:

```
# prlctl list -i MyVM | grep hdd0
Boot order: hdd0 cdrom0 net0
hdd0 (+) ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk.hdd'
```

The command output shows that now the interface type of the `hdd0` disk is IDE.

You can create additional disks for the virtual machine `MyVM`. For example, to add a new disk of the IDE type to the virtual machine, execute the following command:

```
# prlctl set MyVM --device-add hdd --iface ide
Creating hdd1 (+) ide:1 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' 65536Mb
Create the expanding image file, 65536Mb...
The VM has been successfully configured.
```

You can also create a VirtIO disk. To do this, specify `--iface virtio` instead of `--iface ide` in the command above. If you omit the `--iface` option, a SCSI disk is created by default.

The maximum number of devices you can add to a virtual machine is given below:

- 4 IDE devices (virtual hard disks or CD/DVD-ROM drives)
- 15 SCSI devices (virtual hard disks or CD/DVD-ROM drives)
- 15 VirtIO virtual hard disks

At any time, you can remove the `hdd1` disk from the virtual machine `MyVM`:

```
# prlctl set MyVM --device-del hdd1
Remove the hdd1 device.
The VM has been successfully configured.
```

Note:

1. Virtual IDE and SCSI disks can be added to or removed from stopped virtual machines only.
2. You need to initialize a newly added disk before you can start using it. To initialize the disk, use standard means provided by your guest operating system.

3.4 Managing Network Accounting and Bandwidth

This section explains how to perform the following tasks in Virtuozzo:

- configuring network classes
- viewing network traffic statistics
- turning on and off network bandwidth management
- configuring bandwidth limits

3.4.1 Network Traffic Parameters

The table below summarizes the network traffic parameters that you can control in Virtuozzo.

Parameter	Description
`traffic_shaping`	If set to `yes`, traffic limitations for outgoing traffic are set for virtual machines and containers. The default is `no`.
`bandwidth`	This parameter lists all network adapters installed on the hardware node and their bandwidth.
`totalrate`	This parameter defines the bandwidth to allocate for each network class. It is active if traffic shaping is turned on.
`rate`	If traffic shaping is turned on, this parameter specifies the bandwidth guarantee for virtual machines and containers.

Parameter	Description
ratebound	If this parameter is set to yes, the bandwidth guarantee (the global rate parameter) is also the limit for the virtual machine or container, and the virtual machine or container cannot borrow the bandwidth from the totalrate bandwidth pool.

3.4.2 Configuring Network Classes

Virtuozzo allows you to track the inbound and outbound network traffic as well as to shape the outgoing traffic for virtual machines and containers. To provide the ability to distinguish between types of traffic, e.g., domestic and international, a concept of network classes is introduced. A network class is a range of IP addresses for which Virtuozzo accounts and shapes the traffic.

Classes are specified in the `/etc/vz/conf/networks_classes` file. The file is in the ASCII format, and all empty lines and lines starting with the # sign are ignored. Other lines have the following format:

```
<class_id> <IP_address>/<prefix_length>
```

where `<class_id>` defines the network class ID, and the `<IP_address>/<prefix_length>` pair defines the range of IP addresses for this class. There may be several lines for each class.

Classes 0 and 1 have special meanings:

- Class 0 defines the IP address range for which no accounting is performed. Usually, it corresponds to the hardware node subnet (the node itself and its virtual machines and containers). Setting up class 0 is not required; however, its correct setup improves performance.
- Class 1 is defined by Virtuozzo to match any IP address. It must be always present in the network classes definition file. Therefore, it is suggested not to change the default line in the `networks_classes` file.

  ```
  1 0.0.0.0/0
  ```

  If your virtual machines and containers are using IPv6 addresses, you can also add the following line to this file:

  ```
  1 ::/0
  ```

Other classes should be defined after class 1. They represent exceptions from the “matching-everything” rule of class 1. The example below illustrates a possible configuration of the network classes definition file containing rules for both IPv4 and IPv6 addresses:

```
# Hardware node networks
0 192.168.0.0/16
0 fe80::/64
# any IP address (all traffic)
1 0.0.0.0/0
1 ::/0
# class 2 - addresses for the "foreign" traffic
2 10.0.0.0/8
2 2001:db88::/64
# inside "foreign" network there
# is a hole belonging to "local" traffic
1 10.10.16.0/24
1 2001:db88:3333::/64
```

In this example, IPv4 addresses in the range of `192.168.0.0` to `192.168.255.255` and IPv6 addresses in the range of `fe80::` to `fe80::ffff:ffff:ffff:ffff` are treated as class 0 addresses and no accounting is done for the traffic from virtual machines and containers destined to these addresses.

Class 2 matches the following IP addresses:

- IPv4 addresses from `10.0.0.0` to `10.255.255.255` with the exception of addresses in the sub-range of `10.10.16.0` to `10.10.16.255`, which are treated as class 1.
- IPv6 addresses from `2001:db88::` to `2001:db88::ffff:ffff:ffff:ffff` with the exception of addresses in the sub-range of `2001:db88:3333::` to `2001:db88:3333::ffff:ffff:ffff:ffff`, which are also treated as class 1.

All other IP addresses (both IPv4 and IPv6) belong to class 1.

To apply changes after editing the `/etc/vz/conf/networks_classes` file, restart either the virtual machine(s) or/and container(s) for which changes have been made or the hardware node itself if the changes are global.

3.4.3 Viewing Network Traffic Statistics

In Virtuozzo, you can view the current network traffic statistics for virtual machines and containers using the `vznetstat` utility. For example:

```
# vznetstat
UUID          Net.Class  Input(bytes)  Input(pkts)  Output(bytes)  Output(pkts)
0             0          566093064     2800575      3120481        41736
47406484...   0          67489         155          8033           110
fbb30afa-...  0          9369          78           12692          71
```

By default, `vznetstat` shows network statistics for both virtual machines and containers. Keep in mind that the `vznetstat` utility displays statistics only about virtual machines and containers that were started at least once.

The `vznetstat` utility displays the following information:

Column	Description
UUID	UUID assigned to virtual machine or container.
Net.Class	ID of the network class for which network statistics is calculated.
Input(bytes)	Amount of incoming traffic, in bytes.
Input(pkts)	Amount of incoming traffic, in packets.
Output(bytes)	Amount of outgoing traffic, in bytes.
Output(pkts)	Amount of outgoing traffic, in packets.

For example, from the command output above, you can see that around 9 MB of data were uploaded to the container `MyCT`, (2) about 12 MB were downloaded from it, and all the traffic was exchanged with servers from class 0 networks.

If necessary, you can view network traffic statistics separately for virtual machine or container by passing the `-t` option to `vznetstat`:

- For containers only:

  ```
  # vznetstat -t ct
  CTID          Net.Class  Input(bytes)  Input(pkts)  Output(bytes)  Output(pkts)
  0             0          566093064     2800575      3120481        41736
  fbb30afa-...  0          9369          78           12692          71
  ```

- For virtual machines only:

  ```
  # vznetstat -t vm
  UUID          Net.Class  Input(bytes)  Input(pkts)  Output(bytes)  Output(pkts)
  0             0          566093064     2800575      3120481        41736
  47406484...   0          67489         155          8033           110
  ```

You can also view network statistics for a particular virtual machine or container by specifying its ID after the `-v` option, for example:

```
# vznetstat -v fbb30afa-e770-4081-9d9e-6b9c262eb091
UUID          Net.Class  Input(bytes)  Input(pkts)  Output(bytes)  Output(pkts)
fbb30afa-...  0          9369          78           12692          71
```

This command displays statistics only for the container `MyCT`.

3.4.4 Configuring Traffic Shaping

Traffic shaping (also known as network bandwidth management) allows you to control what network bandwidth a virtual machine or container may use for outgoing traffic. This feature is disabled by default.

Note:

1. Traffic within a host cannot be shaped in the current version of Virtuozzo. This includes traffic between virtual machines and containers on the same host and between those and the host itself.
2. Incoming traffic cannot be shaped for virtual machines and containers in the current version of Virtuozzo.

The following parameters control traffic shaping in Virtuozzo:

- `TRAFFIC_SHAPING`, enables and disables traffic shaping.
- `BANDWIDTH`, sets bandwidth for specific network adapters.
- `TOTALRATE`, sets the size of a bandwidth pool divided between virtual machines and containers on the host.
- `RATEMPU`, limits packet rate in addition to byte rate.
- `RATE`, sets a bandwidth guarantee for virtual machines and containers.
- `RATEBOUND`, forces `RATE` as a limit.

Traffic shaping in Virtuozzo works as follows. The bandwidth pool for a given network class (set by `TOTALRATE`) is divided among the virtual machines and containers transmitting data proportionally to their `RATE` settings. If the sum of `RATE` values of all virtual machines and containers transmitting data does not exceed `TOTALRATE`, each virtual machine or container gets the bandwidth equal to or greater than its `RATE` value (unless `RATEBOUND` is enabled for said virtual machine or container). If the sum of `RATE` values of all virtual machines and containers transmitting data exceeds the `TOTALRATE` value, each virtual machine or container may get less than its `RATE` value.

To enable and configure traffic shaping, do the following:

1. Set the value of `TRAFFIC_SHAPING` to `yes` in the global configuration file `/etc/vz/vz.conf`.
2. Set the parameters `BANDWIDTH`, `TOTALRATE` in `/etc/vz/vz.conf`.
3. If required, set the optional parameters `RATEMPU`, `RATE`, `RATEBOUND` in `/etc/vz/vz.conf`.

4. If required, set `RATE` and `RATEBOUND` for specific virtual machines and containers with `prlctl set --rate` and `prlctl set --ratebound` commands.
5. To apply changes, restart either the virtual machines and containers for which changes have been made or the hardware node itself if the changes are global.

The following sections provide more details on and explain how to set traffic shaping parameters listed above.

3.4.4.1 Setting BANDWIDTH Parameter

The `BANDWIDTH` parameter is used for shaping traffic of specific network adapters. For example, for two Fast Ethernet cards, a typical setting may look like `enp0s5 enp0s6:100000` where `enp0s5` and `enp0s6` are network adapter names. By default, the parameter is set to `100000` which corresponds to a 100 Mbps Fast Ethernet card.

3.4.4.2 Setting TOTALRATE Parameter

The `TOTALRATE` parameter specifies the size of a bandwidth pool for specific network classes on the host. Virtual machines and containers can borrow bandwidth from the pool for communicating with hosts from the corresponding network class. The parameter thus limits the total available outgoing traffic for a network class that virtual machines and containers can consume.

The parameter is set as `<NIC>:<network_class>:<bandwidth_in_Kbps>`. For example, to set the pool size to 4 Mbps for network class 1 on the Ethernet adapter `enp0s5`, set `TOTALRATE` to `enp0s5:1:4000`. Multiple entries can be separated by spaces, e.g., `enp0s5:1:4000 enp0s6:2:8000`.

3.4.4.3 Setting RATEMPU Parameter

The optional `RATEMPU` parameter (where "MPU" stands for "minimum packet unit") limits the packet rate by making packets smaller than MPU in size consume HTB tokens. With it, small packets can be accounted as larger ones and limited by `TOTALRATE` and `RATE` parameters. Approximately, the maximum packets per second rate can be calculated as `TOTALRATE` / `RATEMPU`.

This parameter has the following syntax: `<NIC>:<network_class>[:<MPU_in_bytes_per_packet>]`. If the part `<MPU_in_bytes_per_packet>` is omitted, the default value of 1000 bytes is used. Multiple entries can be separated by spaces, e.g., `enp0s5:1:2000 enp0s6:2:4000`. To set the `RATEMPU` parameter for all known Ethernet devices set `<NIC>` to an asterisk (`*`). For example, to set the minimal packet size to 2 Kb for network class 1 on all the Ethernet adapters on the node, change the value to `*:1:2000`.

3.4.4.4 Setting RATE and RATEBOUND Parameters

The optional `RATE` parameter allows you to guarantee virtual machines and containers outgoing bandwidth to destinations in a specific network class on a specific Ethernet device. The guaranteed bandwidth is not a limit (unless the `RATEBOUND` parameter is also set to `on`, see below). A virtual machine or container can additionally obtain unused bandwidth from the bandwidth pool defined by `TOTALRATE`.

You can set the guaranteed bandwidth in two ways:

1. For all virtual machines and containers on the host by setting `RATE` in the global configuration file `/etc/vz/vz.conf`.

   The parameter is set as `<NIC>:<network_class>:<bandwidth_in_Kbps>`. For example, to guarantee all virtual machines and containers on the host the bandwidth of at least 8 Kbps for outgoing traffic in network class 1 on the Ethernet device `enp0s5`, set the `RATE` parameter to `enp0s5:1:8`.

2. For specific virtual machines or containers by means of the `prlctl set --rate` command.

   For example, to guarantee the container `MyCT` the bandwidth of at least 16 Kbps for outgoing traffic in network class 1, run

   ```
   # prlctl set MyCT --rate 1:16
   ```

   This command sets the bandwidth for the default network adapter only. If you need to set bandwidth for other network adapters, set `RATE` in `/etc/vz/vz.conf`.

   Note: It is recommended to increase `RATE` value in 8 Kbps increments and set it to at least 8 Kbps.

The optional `RATEBOUND` parameter specifies whether the network bandwidth guaranteed by `RATE` is also a limit. By default, this feature is disabled for all newly created virtual machines and containers so they may additionally obtain unused bandwidth from the pool set by `TOTALRATE`.

You can limit bandwidth of virtual machines and containers to the guaranteed value as follows:

1. For all virtual machines and containers on the host by setting `RATEBOUND` in the global configuration file `/etc/vz/vz.conf` (omitted by default).

2. For specific virtual machines or containers by means of the `prlctl set --ratebound` command. For example:

```
# prlctl set MyCT --ratebound yes
```

If set, values of RATE and RATEBOUND provided for specific virtual machines and containers are chosen over global values in /etc/vz/vz.conf.

3.4.4.5 Traffic Shaping Example

The example below illustrates a scenario when the containers MyCT1 and MyCT2 have RATEBOUND set to no, and the virtual machine MyVM has RATEBOUND set to yes. With the default TOTALRATE of 4096 Kbps and RATE of 8 Kbps, the bandwidth pool will be distributed as follows:

MyCT1	MyCT2	MyVM	Consumed Bandwidth
transmits	idle	idle	MyCT1: 4096 Kbps
idle	idle	transmits	MyVM: 8 Kbps
transmits	transmits	idle	MyCT1: 2048 Kbps MyCT2: 2048 Kbps
transmits	idle	transmits	MyCT1: 4032 Kbps MyVM: 8 Kbps
transmits	transmits	transmits	MyCT1: 2016 Kbps MyCT2: 2016 Kbps MyVM: 8 Kbps

3.5 Managing Disk I/O Parameters

This section explains how to manage disk input and output (I/O) parameters in Virtuozzo systems.

3.5.1 Configuring Priority Levels for Virtual Machines and Containers

In Virtuozzo, you can configure the disk I/O (input/output) priority level of virtual machines and containers. The higher the I/O priority level, the more time the virtual machine or container will get for its disk I/O activities as compared to the other virtual machines and containers on the hardware node. By default, any virtual machine or container on the hardware node has the I/O priority level set to 4. However, you can change the current I/O priority level in the range from 0 to 7 using the --ioprio option of the prlctl set command. For example, you can issue the following command to set the I/O priority of the container MyCT and the virtual machine MyVM to 6:

```
# prlctl set MyCT --ioprio 6
# prlctl set MyVM --ioprio 6
```

To check the I/O priority level currently applied to the container `MyCT` and the virtual machine `MyVM`, you can execute the following commands:

- For container `MyCT`:

  ```
  # grep IOPRIO /etc/vz/conf/fbb30afa-e770-4081-9d9e-6b9c262eb091.conf
  IOPRIO="6"
  ```

- For the virtual machine `MyVM`:

  ```
  # prlctl list MyVM --info | grep ioprio
    cpu cpus=2 VT-x accl=high mode=32 ioprio=6 iolimit='0'
  ```

3.5.2 Configuring Disk I/O Bandwidth

In Virtuozzo, you can configure the bandwidth virtual machines and containers are allowed to use for their disk input and output (I/O) operations. Limiting the disk I/O bandwidth can help you prevent the situations when high disk activities in one virtual machine or container (generated, for example, by transferring huge amounts of data to/from the virtual machine or container) can slow down the performance of other virtual machines and containers on the hardware node.

By default, the I/O bandwidth limit for all newly created virtual machines and containers is set to 0, which means that no limits are applied to any virtual machines and containers. To limit the disk I/O bandwidth for a virtual machine or container, you can use the `--iolimit` option of the `prlctl set` command. For example, the following command sets the I/O bandwidth limit for the container `MyCT` to 10 megabytes per second (MB/s):

```
# prlctl set MyCT --iolimit 10
```

By default, the limit is set in megabytes per second. However, you can use the following suffixes to use other measurement units:

- `G` sets the limit in gigabytes per second (1G).
- `K` sets the limit in kilobytes per second (10K).
- `B` sets the limit in bytes per second (10B).

Note: In the current version of Virtuozzo, the maximum I/O bandwidth limit you can set for a virtual machine or container is 2 GB per second.

To check that the I/O speed limit has been successfully applied to the container `MyCT`, use the `prlctl list` command:

```
# prlctl list MyCT -o iolimit
IOLIMIT
10485760
```

At any time, you can remove the I/O bandwidth limit set for container `MyCT` by running this command:

```
# prlctl set MyCT --iolimit 0
```

3.5.3 Configuring the Number of I/O Operations Per Second

In Virtuozzo, you can limit the maximum number of disk input and output operations per second virtual machines and containers are allowed to perform (known as the IOPS limit). You may consider setting the IOPS limit for virtual machines and containers with high disk activities to ensure that they do not affect the performance of other virtual machines and containers on the Node.

Note: By default all I/O inside containers is cached and the direct access flag (`O_DIRECT`) is ignored when opening files. This significantly reduces the number of IOPS required for container workload and helps avoid I/O bottlenecks on the Node. For instructions on how to configure honoring of the `O_DIRECT` flag inside containers, see *Setting the Direct Access Flag Inside Containers* on page 90 below.

By default, IOPS is not limited for newly created virtual machines and containers. To set the IOPS limit, you can use the `--iopslimit` option of the `prlctl set` command. For example, to allow the container `MyCT` and the virtual machine `MyVM` to perform no more than 100 disk I/O operations per second, you can run the following commands:

```
# prlctl set MyCT --iopslimit 100
# prlctl set MyVM --iopslimit 100
```

To ensure that the IOPS limit has been successfully applied, use the `prlctl list -i` command. For example:

```
# prlctl list MyCT -i | grep iopslimit
<...> iopslimit=100
```

At any time, you can remove the set IOPS limits by running this command:

```
# prlctl set MyCT --iopslimit 0
# prlctl set MyVM --iopslimit 0
```

3.5.3.1 Setting the Direct Access Flag Inside Containers

You can configure honoring of the `O_DIRECT` flag inside containers with the `sysctl` parameter `fs.odirect_enable`:

- To ignore the `O_DIRECT` flag inside a container, set `fs.odirect_enable` to `0` in that container.
- To honor the `O_DIRECT` flag inside the container, set `fs.odirect_enable` to `1` in that container.
- To have a container inherit the setting from the hardware node, set `fs.odirect_enable` to `2` in that container (default value). On the hardware node, `fs.odirect_enable` is `0` by default.

> **Note:** The `fs.odirect_enable` parameter on the Node only affects honoring of the `O_DIRECT` flag in containers and not on the Node itself where the `O_DIRECT` flag is always honored.

3.5.4 Viewing Disk I/O Statistics

In Virtuozzo, you can view disk input and output (I/O) statistics for all processes on the host. To do this:

1. Run the `vztop` utility.
2. Press F2 or S to switch to the Setup menu.
3. In the Setup column, choose Columns.
4. In Available Columns, choose from the following parameters to add to the output (Active Columns):

Parameter	Description	Column
`RBYTES`	Number of bytes read for the process.	IO_RBYTES
`WBYTES`	Number of bytes written for the process.	IO_WBYTES
`IO_READ_RATE`	Process read rate, in bytes per second.	DISK READ
`IO_WRITE_RATE`	Process write rate, in bytes per second.	DISK WRITE
`IO_RATE`	Process total I/O rate, in bytes per second.	DISK R/W
`IO_PRIORITY`	Process I/O priority.	IO

   To add a parameter, select it and press F5 or Enter. To remove a parameter from Active Columns, select it and press F9.

5. When you finish managing columns, press F10 to save the changes and view the output.

3.5.5 Setting I/O Limits for Backup and Migration Operations

Backup and migration operations with containers and virtual machines can generate a high I/O load on the server, thus reducing the performance of other virtual environments or the server itself. You can avoid such situations by setting I/O limits for these operations.

To set an I/O limit, do the following:

1. In the `/etc/vz/vz.conf` global configuration file, locate the following section:

   ```
   # VZ Tools limits
   <...>
   # Uncomment next line to specify required disk IO bandwidth in Bps (10485760 - 10MBps)
   # VZ_TOOLS_IOLIMIT=10485760
   ```

2. Uncomment the `VZ_TOOLS_IOLIMIT` parameter, and set the I/O limit for backup and migration operations in bytes per second.

3. Save the file.

When setting I/O limits, pay attention to the following:

- `VZ_TOOLS_IOLIMIT` is a global parameter that has effect on all virtual environments on the server.
- The `VZ_TOOLS_IOLIMIT` parameter controls the I/O load only for backup and migration operations. Restore operations are not limited.
- Simultaneous operations do not share the limit and are limited separately.
- For migration, only the limit set on the source server applies, while the limit set on the destination server is ignored.

3.6 Managing Containers Memory Parameters

This section describes the VSwap memory management system. You will learn to do the following:

- Configure the main VSwap parameters for containers.
- Set the memory allocation limit in containers.
- Configure OOM killer behavior.
- Enhance the VSwap functionality.

3.6.1 Configuring Main VSwap Parameters

Virtuozzo utilizes the VSwap scheme for managing memory-related parameters in containers. Like many other memory management schemes used on standalone Linux computers, this scheme is based on two main parameters:

- `RAM` determines the total size of RAM that can be used by the processes of a container.
- `swap` determines the total size of swap that can be used by a container for swapping out memory once the RAM is exceeded.

The memory management scheme works as follows:

1. You set for a container a certain amount of RAM and swap space that can be used by the processes running in the container.
2. When the container exceeds the RAM limit set for it, the swapping process starts. The swapping process for containers slightly differs from that on a standalone computer. The container swap file is virtual and, if possible, resides in the Node RAM. In other words, when the swap-out for a container starts and the Node has enough RAM to keep the swap file, the swap file is stored in the Node RAM rather than on the hard drive.
3. Once the container exceeds its swap limit, the system invokes the OOM Killer for this container.
4. The OOM Killer chooses one or more processes running in the affected container and forcibly kills them.

By default, any newly created container starts using the new memory management scheme. To find out the amount of RAM and swap space set for a container, you can check the values of the `PHYSPAGES` and `SWAPPAGES` parameters in the container configuration file, for example:

```
# grep PHYSPAGES /etc/vz/conf/26bc47f6-353f-444b-bc35-b634a88dbbcc.conf
PHYSPAGES="65536:65536"
# grep SWAPPAGES /etc/vz/conf/26bc47f6-353f-444b-bc35-b634a88dbbcc.conf
SWAPPAGES="65536"
```

In this example, the value of the `PHYSPAGES` parameter for the container `MyCT` is set to 65536. The `PHYSPAGES` parameter displays the amount of RAM in 4-KB pages, so the total amount of RAM set for the container `MyCT` equals to 256 MB. The value of the `SWAPPAGES` parameter is also set to 256 MB.

To configure the amounts of RAM and swap space for the container `MyCT`, use the `--memsize` and `--swappages` options of the `prlctl set` command. For example, you can execute the following command to set the amount of RAM and SWAP in the container `MyCT` to 1 GB and 512 MB, respectively:

```
# prlctl set MyCT --memsize 1G --swappages 512M
```

3.6.2 Configuring Container Memory Guarantees

A memory guarantee is a percentage of container's RAM that said container is guaranteed to have.

Important: The total memory guaranteed to all running virtual environments on the host must not exceed host's physical RAM size. If starting a virtual environment with a memory guarantee would increase the total memory guarantee on the host beyond host's physical RAM size, said virtual environment will not start. If setting a memory guarantee for a running virtual environment would increase the total memory guarantee on the host beyond host's physical RAM size, said memory guarantee will not be set.

For containers, the memory guarantee value is set to 0% by default. To change the default value, use the `prlctl set --memguarantee` command. For example:

```
# prlctl set MyCT --memguarantee 80
```

To revert to the default setting, run

```
# prlctl set MyCT --memguarantee auto
```

3.6.3 Configuring Container Memory Allocation Limit

When an application starts in a container, it allocates a certain amount of memory for its needs. Usually, the allocated memory is much more than the application actually requires for its execution. This may lead to a situation when you cannot run an application in the container even if it has enough free memory. To deal with such situations, the VSwap memory management scheme introduces the new `vm_overcommit` option. Using it, you can configure the amount of memory applications in a container may allocate, irrespective of the amount of RAM and swap space assigned to the container.

The amount of memory that can be allocated by applications of a container is the sum of RAM and swap space set for this container multiplied by a memory overcommit factor. In the default (basic) container configuration file, this factor is set to 1.5. For example, if a container is based on the default configuration file and assigned 1 GB of RAM and 512 MB of swap, the memory allocation limit for the container will be 2304 MB. You can configure this limit and set it, for example, to 3 GB by running this command:

```
# vzctl set MyCT --vm_overcommit 2 --save
```

This command uses the factor of 2 to increase the memory allocation limit to 3 GB:

(1 GB of RAM + 512 MB of swap) * 2 = 3 GB

Now applications in the container `MyCT` can allocate up to 3 GB of memory, if necessary.

3.6.4 Configuring Container OOM Killer Behavior

The OOM killer selects a container process (or processes) to end based on the badness reflected in `/proc/<pid>/oom_score`. The badness is calculated using process memory, total memory, and badness adjustment, and then clipped to the range from 0 to 1000. Each badness point stands for one thousandth of container memory. The process to be killed is the one with the highest resulting badness.

The OOM killer for container processes can be configured using the `/etc/vz/oom-groups.conf` file that lists patterns based on which badness adjustment is selected for each running process. Each pattern takes a single line and includes the following columns:

- `<command>`, mask for the task command name;
- `<parent>`, mask for the parent task name;
- `<oom_uid>`, task user identifier (UID) filter:
  - If `<oom_uid>` is -1, the pattern will be applicable to tasks with any UIDs,
  - If `<oom_uid>` is 0 or higher, the pattern will be applicable to tasks with UIDs equal to the `<oom_uid>` value,
  - If `<oom_uid>` is smaller than -1, the pattern will be applicable to tasks with UIDs smaller than the negative `<oom_uid>` value);
- `<oom_score_adj>` badness adjustment. As with badness itself, each adjustment point stands for one thousandth of total container memory. Negative adjustment values reduce process badness. In an out-of-memory situation, an adjustment will guarantee that the process will be allowed to occupy at least `<oom_score_adj>` thousandths of container memory while there are other processes with higher badness running in the container.

Note: The `<command>` and `<parent>` masks support wildcard suffixes: asterisk matches any suffix. E.g., “foo” matches only “foo”, “foo*” matches “foo” and “foobar”.

For example, the pattern

```
sshd     init     -500     -100
```

means that in an out-of-memory situation, `sshd`, a child of `init`, will be guaranteed at least 100 thousandths (i.e., 10%) of container memory, if its UID is smaller than -(-500) or just 500, e.g., 499. According to RHEL conventions, UIDs from 1 to 499 are usually reserved for system use, so such delimitation may be useful to prioritize and save system processes.

While calculating the badness of a process, the OOM killer searches `/proc/vz/oom_score_adj` for a suitable pattern based on masks and task UID filter. The search starts from the first line and ends when the first suitable pattern is found. The corresponding adjustment value is then used to obtain the resulting process badness.

The data from `/etc/vz/oom-groups.conf` is reset and committed to the kernel on boot. To reset and commit the config file manually, you can use the following command:

```
# cat /etc/vz/oom-groups.conf > /proc/vz/oom_score_adj
```

3.6.5 Tuning VSwap

The VSwap management scheme can be extended by using UBC parameters. For example, you can set the `numproc` parameter to configure the maximal number of processes and threads a container may create or the `numfile` parameter to specify the number of files that may be opened by all processes in the container.

3.7 Managing Virtual Machines Memory Parameters

This section describes how to configure memory parameters available for virtual machines:

- memory size,
- video memory size,
- memory hotplugging,
- memory guarantees.

3.7.1 Configuring Virtual Machine Memory Size

To increase or reduce the amount of memory that will be available to the virtual machine, use the `--memsize` option of the `prlctl set` command. The following example shows how to increase the RAM of the virtual machine `MyVM` from 1GB to 2GB and check that the new value has been successfully set:

```
# prlctl list -i MyVM | grep memory
memory 1024Mb
# prlctl set MyVM --memsize 2048
Set the memsize parameter to 2048Mb
The VM has been successfully configured.
# prlctl list -i MyVM | grep memory
memory 2048Mb
```

The changes are saved in the VM configuration file and applied to the VM on start. If the VM is running, it will need to be rebooted. To be able to increase or reduce virtual machine RAM size without reboot, enable memory hotplugging as described in *Enabling Virtual Machine Memory Hotplugging* on page 97.

Note: The value set with `prlctl --memsize` is not reported inside the VM as physical or other RAM size. A user logged in to the guest OS will see as much physical RAM as can be obtained by fully deflating the balloon (see MaxNumaSize in *Enabling Virtual Machine Memory Hotplugging* on page 97). The balloon size is not reported inside the VM as well. However, if the balloon is not fully deflated, a part of the reported physical RAM will appear to be occupied at all times (by what is in fact the balloon).

3.7.2 Configuring Virtual Machine Video Memory Size

To set the amount of video memory to be available to the virtual machine's video card, use the `--videosize` option of the `prlctl set` command. Assuming that the current video memory size of the virtual machine `MyVM` is set to 32 MB, you can increase it to 64 MB by running the following command:

```
# prlctl set MyVM --videosize 64
```

To check that the new value has been successfully set, use this command:

```
# prlctl list -i MyVM | grep video
video 64Mb
```

3.7.3 Enabling Virtual Machine Memory Hotplugging

Memory hotplugging allows increasing or reducing virtual machine RAM size on the fly, without the need to reboot the VM. Memory hotplugging is implemented as a combination of ballooning and addition/removal of virtual DIMM slots.

The algorithm is as follows. When a command to increase VM memory size to RAM_size is run (as described in *Configuring Virtual Machine Memory Size* on page 96), the memory is first expanded by deflating the VM's balloon. The balloon deflation limit, MaxNumaSize, is calculated automatically according to the formula

```
MaxNumaSize = (RAM_size + 4GB) rounded up to a multiple of 4GB
```

If fully deflating the balloon is not enough to obtain RAM_size (that is, RAM_size exceeds MaxNumaSize), then memory is further expanded by adding virtual DIMM slots (up to twice the MaxNumaSize) and MaxNumaSize is set equal to RAM_size (that is, the maximum balloon size grows as well). When a command to decrease VM memory size is run, the memory is shrunk by inflating the VM's balloon. The added virtual DIMM slots remain until VM restart. After restart, the VM has memory equal to RAM_size.

This feature is only supported for virtual machines with at least 1GB of RAM and is disabled by default. To enable it for a virtual machine (e.g., `MyVM`):

1. Make sure the VM is stopped.
2. Run

   ```
   # prlctl set MyVM --mem-hotplug on
   ```

3. Start the VM.

Now virtual machine RAM size can be increased and decreased with the `prlctl set --memsize` command without rebooting the VM.

3.7.4 Configuring Virtual Machine Memory Guarantees

A memory guarantee is a percentage of virtual machine's RAM that said VM is guaranteed to have.

Important: The total memory guaranteed to all running virtual environments on the host must not exceed host's physical RAM size. If starting a virtual environment with a memory guarantee would increase the total memory guarantee on the host beyond host's physical RAM size, said virtual environment will not start. If setting a memory guarantee for a running virtual environment would increase the total memory guarantee on the host beyond host's physical RAM size, said memory guarantee will not be set.

For virtual machines, the memory guarantee value is set to 80% by default. To change the default value, use the `prlctl set --memguarantee` command. For example:

```
# prlctl set MyVM --memguarantee 60
```

To revert to the default setting, run

```
# prlctl set MyVM --memguarantee auto
```

Note: Virtual machines with memory guarantees can only be started with `prlctl start`. Starting such VMs differently (e.g., using `virsh`) will result in memory guarantees not being applied.

3.8 Managing Container Resource Configuration

Any container is configured by means of its own configuration file. You can manage container configurations in a number of ways:

1. Using configuration sample files shipped with Virtuozzo. These files are used when a new container is being created (for details, see *Virtuozzo Containers* on page 3). Currently, the following configuration sample files are provided:

    - `basic` for creating standard containers.
    - `confixx` for creating containers that are to run the Confixx control panel.
    - `vswap.plesk` for creating containers with the Plesk control panel.
    - `vswap.256MB` for creating containers with 256 MB of main memory.
    - `vswap.512Mb` for creating containers with 512 MB of main memory.

- `vswap.1024Mb` for creating containers with 1024 MB of main memory.
- `vswap.2048Mb` for creating containers with 2048 MB of main memory.

Note: Configuration sample files cannot contain spaces in their names.

Any sample configuration file can also be applied to an existing container. You would do this if, for example, you want to upgrade or downgrade the overall resources configuration of a particular container:

```
# prlctl set MyCT --applyconfig basic
```

This command applies all the parameters from the `ve-basic.conf-sample` file to the container `MyCT`. When you install Virtuozzo on your hardware node, the default container samples are put to the `/etc/vz/conf` directory. They have the following format: `ve-<name>.conf-sample` (for example, `ve-basic.conf-sample`).

2. Using specific utilities for preparing configuration files in their entirety. The tasks these utilities perform are described in the following subsections of this section.
3. The direct creating and editing of the corresponding container configuration file (`/etc/vz/conf/<UUID>.conf`). This can be performed with the help of any text editor. The instructions on how to edit container configuration files directly are provided in the four preceding sections. In this case you have to edit all the configuration parameters separately, one by one.

3.8.1 Splitting Server Into Equal Pieces

Using the `vzsplit` command, you can create configurations for containers that would take a specific fraction of the hardware node resources. For example, to create a configuration `myconf` for up to 20 containers:

```
# vzsplit -n 20 -f myconf
Config /etc/vz/conf/ve-myconf.conf-sample was created
```

The configuration is calculated based on the hardware node resources. You can now use the `--config myconf` option of the `prlctl create` command to create containers based on this configuration.

3.8.2 Applying New Configuration Samples to Containers

Virtuozzo allows you to change the configuration sample file a container is based on and, thus, to modify all the resources the container may consume and/or allocate at once. For example, if the container `MyCT` is currently based on the `basic` configuration sample and you are planning to run the Plesk application inside the container, you may wish to apply the `vswap.plesk` sample to it instead of `basic`, which will automatically adjust the necessary container resource parameters for running the Plesk application inside the container `MyCT`. To do this, you can execute the following command on the hardware node:

```
# prlctl set MyCT --applyconfig vswap.plesk
```

This command reads the resource parameters from the `ve-vswap.plesk.conf-sample` file located in the `/etc/vz/conf` directory and applies them one by one to the container `MyCT`.

When applying new configuration samples to containers, keep in mind the following:

- All container sample files are located in the `/etc/vz/conf` directory on the hardware node and are named according to the following pattern: `ve-<name>.conf-sample`. You should specify only the `<name>` part of the corresponding sample name after the `--applyconfig` option (`vswap.plesk` in the example above).
- The `--applyconfig` option applies all the parameters from the specified sample file to the given container, except for the `OSTEMPLATE`, `TEMPLATES`, `VE_ROOT`, `VE_PRIVATE`, `HOSTNAME`, `IP_ADDRESS`, `TEMPLATE`, `NETIF` parameters (if they exist in the sample file).

You may need to restart your container depending on the fact whether the changes for the selected parameters can be set on the fly or not. If some parameters could not be configured on the fly, you will be presented with the corresponding message informing you of this fact.

3.9 Managing Virtual Machine Configuration Samples

The configuration of a virtual machine is defined by its `config.pvs` configuration file. This file in XML format is automatically created when you make a new virtual machine and contains all parameters of the virtual machine: memory, CPU, disk space, and so on.

Once a virtual machine is created, you can manually configure its parameters using the `prlctl` utility. However, if you need to configure multiple parameters for several virtual machines, this may become a tedious task. To facilitate your work, you can create virtual machine samples and use them to quickly and easily change the configuration of virtual machines. You can even further simplify the configuration process by creating a virtual machine template and several sample files. In this case, you can quickly make a new virtual machine on the basis of your template and apply the desired configuration file to it.

3.9.1 Creating a Configuration Sample

Before you can start using virtual machine configuration samples, you need to create at least one configuration sample. The easiest way of doing this is to follow the steps below:

1. Create a virtual machine configuration, for example:

   ```
   # prlctl create VmConfiguration
   ```

2. Set the parameters for the virtual machine configuration as you want them to be. For example, you can use the `prlctl set` command to set the required amount of memory and disk space. All your parameters are saved to the `config.pvs` file of the `VmConfiguration` virtual machine. For the list of parameters that can be applied from a configuration sample, see *Parameters Applied from Configuration Samples* on page 101 below.

3. Copy the `config.pvs` file to the `/etc/Parallels/samples` directory. If this directory does not exist, create it:

   ```
   # mkdir /etc/Parallels/samples
   # cp /vz/vmprivate/<VmConfiguration_UUID>/config.pvs /etc/Parallels/samples/configMySQL.pvs
   ```

The latter command copies the `config.pvs` file to the `configMySQL.pvs` file.

3.9.2 Applying Configuration Samples to Virtual Machines

Now that you have created the configuration sample, you can apply it to any of your virtual machines. You can do this using the `--applyconfig` option with the `prlctl set` command and specifying the sample name without the `.pvs` extension. For example, to apply the `configMySQL` sample to the `VM1` virtual machine, you can run this command:

```
# prlctl set VM1 --applyconfig configMySQL
```

You can apply configuration samples to stopped virtual machines only.

3.9.3 Parameters Applied from Configuration Samples

The following parameters are applied to a virtual machine from a new configuration sample:

- All memory-related parameters (both RAM and video). To view these parameters in a sample file, locate the `<Memory>` and `<Video>` elements.
- All CPU-related parameters. To view these parameters in a sample file, locate the `<Cpu>` element.
- IO and IOPS parameters. To view these parameters in a sample file, locate the `<IoLimit>` and `<IopsLimit>` elements, respectively.
- Disk space parameter. To view this parameter in a sample file, locate the `<Size>` element enclosed in the `<Hdd>` element:

```
<Hdd id=0" dyn_lists="Partition 0">
<Index>0</Index>
<Size>65536</Size>
</Hdd>
```

The virtual disk to which the value of the `<Size>` element is applied is defined by the index number in the `<Index>` element. For example, in the example above, the disk space parameter (65536 MB) is applied to the virtual disk with index number 0. If the virtual machine does not have a virtual disk with the specified index, the parameter is ignored.

3.10 Monitoring Resources

In Virtuozzo, you can use the `vztop` utility to monitor system resources in real time. When executed, the utility displays information about processor, swap and memory usage, number of tasks, load average, and uptime at the top of the screen. You can change the default meters by pressing F2 or S. For example, you can run the following command on the server to view your current system resources:

```
# vztop
1  [                                             0.0%]  Tasks: 77, 65 thr; 1 running
2  [|||                                          2.6%]  Load average: 0.02 0.03 0.05
3  [||||                                         4.6%]  Uptime: 06:46:48
4  [|                                            0.7%]
Mem[||||||||||||||||||||                   344M/3.68G]
Swp[                                         0K/3.87G]
```

The numbers on the left represent the number of CPUs/cores in the system. The progress bar shows their load and can be comprised of different colors. By default, the CPU progress bar is displayed in four colors:

- blue - low priority processes,
- green - normal priority (user) processes,
- red - kernel processes,

- cyan - virtualization time.

The memory progress bar is comprised of three colors:

- green - used memory pages,
- blue - buffer pages,
- yellow/orange - cache pages.

The swap progress bar include only one color, red, which denotes used swap space.

The command output is updated in intervals set with the `-d` option in tenths of a second. If the `-d` option is omitted, the default interval is 1 second (i.e. -d 10).

CHAPTER 4

Managing Services and Processes

This chapter provides information on what services and processes are, how they influence the operation and performance of your system, and what tasks they perform in the system.

You will learn how to use the command line utilities in order to manage services and processes in Virtuozzo. In particular, you will learn how to monitor active processes in your system, change the mode of the `xinetd`-dependent services, identify the container UUID where a process is running by the process ID, start, stop, or restart services and processes, and edit the service run levels.

4.1 What Are Services and Processes

Instances of any programs currently running in the system are referred to as processes. A process can be regarded as the virtual address space and the control information necessary for the execution of a program. A typical example of a process is the vi application running on your server or inside your Linux-based containers. Along with common processes, there are a great number of processes that provide an interface for other processes to call. They are called services. In many cases, services act as the brains behind many crucial system processes. They typically spend most of their time waiting for an event to occur or for a period when they are scheduled to perform some task. Many services provide the possibility for other servers on the network to connect to the given one via various network protocols. For example, the `nfs` service provides the NFS server functionality allowing file sharing in TCP/IP networks.

You may also come across the term “daemon” that is widely used in connection with processes and services. This term refers to a software program used for performing a specific function on the server system and is usually used as a synonym for “service”. It can be easily identified by `d` at the end of its name. For example, `httpd` (HTTP daemon) represents a program that runs in the background of your system and waits for incoming requests to a web server. The daemon answers the requests automatically and serves the hypertext and multimedia documents over the Internet using HTTP.

When working with services, you should keep in mind the following. During the lifetime of a service, it uses many system resources. It uses the CPUs in the system to run its instructions and the system’s physical memory to hold itself and its data. It opens and uses files within the file systems and may directly or indirectly use certain physical devices in the system. Therefore, in order not to decrease your system performance, you should run only those services on the hardware node that are really needed at the moment.

Besides, you should always remember that running services in the Host OS is much more dangerous than running them in virtual machines and containers. In case violators get access to one of the virtual machines and containers through any running service, they will be able to damage only the virtual machine or container where this service is running, but not the other virtual machines and containers on your server. The hardware node itself will also remain unhurt. And if the service were running on the hardware node, it would damage both the server and all virtual machines and containers residing on it. Thus, you should make sure that you run only those services on the server that are really necessary for its proper functioning. Launch all additional services you need at the moment inside separate virtual machines and containers. It can significantly improve your system safety.

4.2 Main Operations on Services and Processes

The ability to monitor and control processes and services in your system is essential because of the profound influence they have on the operation and performance of your whole system. The more you know about what each process or service is up to, the easier it will be to pinpoint and solve problems when they creep in.

The most common tasks associated with managing services running on the hardware node or inside a virtual machine or container are starting, stopping, enabling, and disabling a service. For example, you might need to start a service in order to use certain server-based applications, or you might need to stop or pause a service in order to perform testing or to troubleshoot a problem.

For `xinetd`-dependent services, you do not start and stop but enable and disable services. The services enabled in this way are started and stopped on the basis of the corresponding state of the `xinetd` daemon.

Disabled services are not started whatever the `xinetd` state.

In Virtuozzo, you can manage services on the hardware node and inside containers by means of special Linux command-line utilities. You can do it either locally or from any server connected on the network.

As for processes, such Virtuozzo utilities as `vzps`, `vztop`, `vzpid` enable you to see what a process is doing and to control it. Sometimes, your system may experience problems such as slowness or instability, and using these utilities can help you improve your ability to track down the causes. It goes without saying that in Virtuozzo you can perform all those operations on processes you can do in a normal system, for example, kill a process by sending a terminate signal to it.

4.3 Managing Processes and Services

In Virtuozzo, services and processes can be managed using the following command-line utilities:

- `vzps`
- `vzpid`
- `vztop`

With their help, you can perform the following tasks:

- print the information about active processes on your hardware node,
- view the processes activity in real time,
- change the mode of the services that can be either `xinetd`-dependent or standalone,
- identify the container UUID where a process is running by the process ID.

Note: The maximum number of processes per container is limited to 131,072.

4.3.1 Viewing Active Processes and Services

The `vzps` utility provides certain additional functionality related to monitoring separate containers running on the hardware node. For example, you can use the `-E` switch with the `vzps` utility to:

- display the container UUIDs where the processes are running

- view the processes running inside a particular container

vzps prints the information about active processes on your hardware node. When run without any options, vzps lists only those processes that are running on the current terminal. Below is an example output of vzps:

```
# vzps
  PID TTY          TIME CMD
 4684 pts/1    00:00:00 bash
27107 pts/1    00:00:00 vzps
```

Currently, the only processes assigned to the user/terminal are the bash shell and the vzps command itself. In the output, the PID (Process ID), TTY, TIME, and CMD fields are contained. TTY denotes which terminal the process is running on, TIME shows how much CPU time the process has used, and CMD is the name of the command that started the process.

Note: The IDs of the processes running inside containers and displayed by running the vzps command on the hardware node does not coincide with the IDs of the same processes shown by running the ps command inside these containers.

As you can see, the standard vzps command just lists the basics. To get more details about the processes running on your server, you will need to pass some command line arguments to vzps. For example, using the aux arguments with this command displays processes started by other users (a), processes with no terminal or one different from yours (x), the user who started the process and when it began (u).

```
# vzps aux
USER  PID %CPU %MEM  VSZ  RSS TTY   STAT START  TIME COMMAND
root    1  0.0  0.0 1516  128 ?     S    Jul14  0:37 init
root    5  0.0  0.0    0    0 ?     S    Jul14  0:03 [ubstatd]
root    6  0.0  0.0    0    0 ?     S    Jul14  3:20 [kswapd]
#27     7  0.0  0.0    0    0 ?     S    Jul14  0:00 [bdflush]
root    9  0.0  0.0    0    0 ?     S    Jul14  0:00 [kinoded]
root 1574  0.0  0.1  218  140 pts/4 S    09:30  0:00 -bash
```

There is a lot more information now. The fields USER, %CPU, %MEM, VSZ, RSS, STAT, and START have been added. Let us take a quick look at what they tell us.

The USER field shows you which user initiated the command. Many processes begin at system start time and often list root or some system account as the user. Other processes are, of course, run by actual users.

The %CPU, %MEM, VSZ, and RSS fields all deal with system resources. First, you can see what percentage of the CPU the process is currently utilizing. Along with CPU utilization, you can see the current memory utilization and its VSZ (virtual memory size) and RSS (resident set size). VSZ is the amount of memory the program would take up if it were all in memory. RSS is the actual amount currently in memory. Knowing how much a process is currently eating will help determine if it is acting normally or has spun out of control.

You will notice a question mark in most of the TTY fields in the `vzps aux` output. This is because most of these programs were started at boot time and/or by initialization scripts. The controlling terminal does not exist for these processes; thus, the question mark. On the other hand, the `bash` command has a TTY value of pts/4. This is a command being run from a remote connection and has a terminal associated with it. This information is helpful for you when you have more than one connection open to the machine and want to determine which window a command is running in.

STAT shows the current status of a process. In our example, many are sleeping, indicated by an S in the STAT field. This simply means that they are waiting for something. It could be user input or the availability of system resources. The other most common status is R, meaning that it is currently running.

You can also use the `vzps` command to view the processes inside any running container. The example below shows you how to display all active processes inside the container `MyCT` with UUID 26bc47f6-353f-444b-bc35-b634a88dbbcc:

```
# vzps -E 26bc47f6-353f-444b-bc35-b634a88dbbcc
                                CTID     PID TTY          TIME CMD
26bc47f6-353f-444b-bc35-b634a88dbbcc   14663 ?        00:00:00 init
26bc47f6-353f-444b-bc35-b634a88dbbcc   14675 ?        00:00:00 kthreadd/26bc47
26bc47f6-353f-444b-bc35-b634a88dbbcc   14676 ?        00:00:00 khelper
26bc47f6-353f-444b-bc35-b634a88dbbcc   14797 ?        00:00:00 udevd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15048 ?        00:00:00 rsyslogd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15080 ?        00:00:00 sshd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15088 ?        00:00:00 xinetd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15097 ?        00:00:00 saslauthd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15098 ?        00:00:00 saslauthd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15116 ?        00:00:00 sendmail
26bc47f6-353f-444b-bc35-b634a88dbbcc   15125 ?        00:00:00 sendmail
26bc47f6-353f-444b-bc35-b634a88dbbcc   15134 ?        00:00:00 httpd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15139 ?        00:00:00 httpd
26bc47f6-353f-444b-bc35-b634a88dbbcc   15144 ?        00:00:00 crond
26bc47f6-353f-444b-bc35-b634a88dbbcc   15151 ?        00:00:00 mingetty
26bc47f6-353f-444b-bc35-b634a88dbbcc   15152 ?        00:00:00 mingetty
```

4.3.2 Monitoring Processes in Real Time

The `vztop` utility is rather similar to `vzps` but is usually started full-screen and updates continuously with process information. This can help with programs that may infrequently cause problems and can be hard to see with `vzps`. Overall system information is also presented, which makes a nice place to start looking for problems.

The `vztop` utility can be used just as the standard Linux `htop` utility. It shows a dynamic list of all processes running on the system with their full command lines.

By default, it shows information about processor, swap and memory usage, number of tasks, load average, and uptime at the top of the screen. You can change the default meters, along with display options, color schemes, and columns at the setup screen (S or F2).

`vztop` can be used interactively for sending signals to processes. For example, you can kill processes—without knowing their PIDs—by selecting them and pressing F9. You can also change process priority by pressing F7 (increase; can only be done by the `root` user) and F8 (decrease).

The `vztop` utility usually has an output like the following:

```
# vztop
1  [                                            0.0%]  Tasks: 77, 65 thr; 1 running
2  [|||                                         2.6%]  Load average: 0.02 0.03 0.05
3  [||||                                        4.6%]  Uptime: 06:46:48
4  [|                                           0.7%]
Mem[|||||||||||||||||||                   344M/3.68G]
Swp[                                        0K/3.87G]

PID CTID USER  PRI NI VIRT   RES   SHR S CPU% MEM%  TIME+  Command
1      0 root  20  0 41620  4132  2368 S 0.0  0.1  0:05.91 /usr/lib/systemd/systemd
3164  0  root  20  0 19980  1380  1160 S 0.0  0.0  0:00.32 /usr/1ib/systemd/systemd-
3163  0  root  21  1 1402M 56992 10204 S 0.0  1.5  4:12.41 /usr/libexec/qemu-kvm -na
3186  0  root  20  0 1402M 56992 10204 S 0.0  1.5  0:00.09 /usr/libexec/qemu-kvm -na
3185  0  root  20  0 1402M 56992 10204 S 0.7  1.5  2:16.83 /usr/libexec/qemu-kvm -na
3180  0  root  20  0 1402M 56992 10204 S 0.0  1.5  0:00.00 /usr/libexec/qemu-kvm -na
3084  0  smmsp 20  0 85712  2036   516 S 0.0  0.1  0:00.19 sendmail: Queue runner@01
3064  0  root  20  0   98M  2380   572 S 0.0  0.1  0:01.43 sendmail: accepting conne
3036  0  root  20  0  291M  4788  3580 S 0.0  0.1  0:00.00 /usr/sbin/virt1ogd
3037  0  root  20  0  291M  4788  3580 S 0.0  0.1  0:00.00 /usr/sbin/virt1ogd
2787  0  nobody20  0 15548   896   704 S 0.0  0.0  0:00.14 /sbin/dnsmasq --conf-file
2788  0  root  20  0 15520   184     0 S 0.0  0.0  0:00.00 /sbin/dnsmasq --conf-file
2479  0  root  20  0 1962M 33344 24160 S 0.7  0.9  3:13.12 /usr/sbin/pr1_disp_servic
9022  0  root  20  0 1962M 33344 24160 S 0.0  0.9  0:10.74 /usr/sbin/pr1_disp_servic
```

The column CTID shows the container UUID inside which the process is running (the value `0` means that the process is running on the server), PRI (PRIORITY) displays the kernel's internal priority for the process, and NI (NICE) shows the nice value (the nicer the process, the more it lets other processes take priority).

To organize processes by parenthood, you can switch to the tree view by pressing F5.

4.3.3 Determining Container UUIDs by Process IDs

Each process is identified by a unique PID (process identifier), which is the entry of that process in the kernel's process table. For example, when you start Apache, it is assigned a process ID. This PID is then used to monitor and control this program. The PID is always a positive integer. In Virtuozzo, you can use the `vzpid` (retrieve process ID) utility to print the container UUID the process with the given id belongs to. Multiple process IDs can be specified as arguments. In this case the utility will print the container number for each of the processes.

The typical output of the `vzpid` utility is shown below:

```
# vzpid 14663
Pid                VEID    Name
14663     26bc47f6-...    init
```

Note: You can also display the container UUID where the corresponding process is running by using the `vzps` utility.

CHAPTER 5

Managing Network

This chapter familiarizes you with the Virtuozzo network structure, lists networking components, and explains how to manage these components in your working environments. In particular, it provides the following information:

- How you can manage network adapters on the hardware node.
- What virtual networks are and how you can manage them on the hardware node.
- How to create virtual network adapters inside your virtual machines and containers and configure their parameters.
- How to connect virtual machines and containers to different networks.

5.1 Managing Network Adapters on the Hardware Node

Network adapters installed on the hardware node are used to provide virtual machines and containers with access to each other and to external networks. During the installation, Virtuozzo registers all physical network adapters available on the server. Once Virtuozzo has been successfully installed, you can manage network adapters on the hardware node using native RHEL7 utilities. You can also create bond and VLAN interfaces (for example, with the `nmtui` tool) and use them instead of physical ones.

Important:

1. Each network adapter must have only one configuration file in the `/etc/sysconfig/network-scripts/` directory.
2. When you need to apply changes to network settings, use `systemctl restart NetworkManager.service` instead of `systemctl restart network`. The latter command may result in issues with both bridged and routed networks.

5.2 Networking Modes in Virtuozzo

This section describes networking modes available in Virtuozzo.

In Virtuozzo, any virtual machine or container can operate in one of the two networking modes: host-routed or bridged.

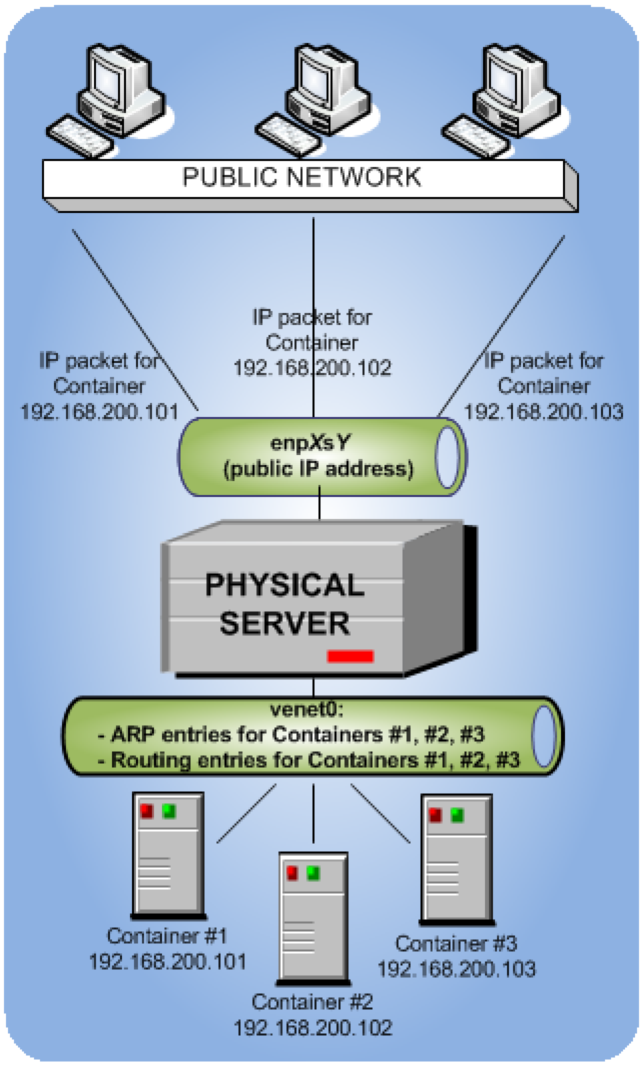
5.2.1 Container Network Modes

This section describes bridged and host-routed network modes for containers.

Note: IPSec connections inside containers are supported.

5.2.1.1 Host-Routed Mode for Containers

By default, a new container starts operating in the host-routed mode. In this mode, the container uses a special network adapter, `venet0`, to communicate with the server where it resides, with the other containers on the server, and with computers on external networks. The figure below demonstrates an example network configuration where all containers are set to work in the host-routed mode.


PUBLIC NETWORK
IP packet for Container 192.168.200.101
IP packet for Container 192.168.200.102
IP packet for Container 192.168.200.103
enpXsY (public IP address)
PHYSICAL SERVER
venet0:
- ARP entries for Containers #1, #2, #3
- Routing entries for Containers #1, #2, #3
Container #1 192.168.200.101
Container #2 192.168.200.102
Container #3 192.168.200.103

In this configuration:

- Containers #1, #2, and #3 use the `venet0` adapter as the default gateway to send and receive data to/from other networks. They also use this adapter to exchange the traffic between themselves.
- When containers #1, #2, and #3 start, the server creates ARP and routing entries for them in its ARP and routing tables. You can view the current ARP and routing entries on a server using the `arp -n` and `route -n` commands. For example:

```
# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.30.0.4                ether   00:1a:e2:c7:17:c1   C                     enp0s5
10.30.23.162             ether   70:71:bc:42:f6:a0   C                     enp0s5
192.168.200.101          *       *                   MP                    enp0s5
192.168.200.102          *       *                   MP                    enp0s5
192.168.200.103          *       *                   MP                    enp0s5
```

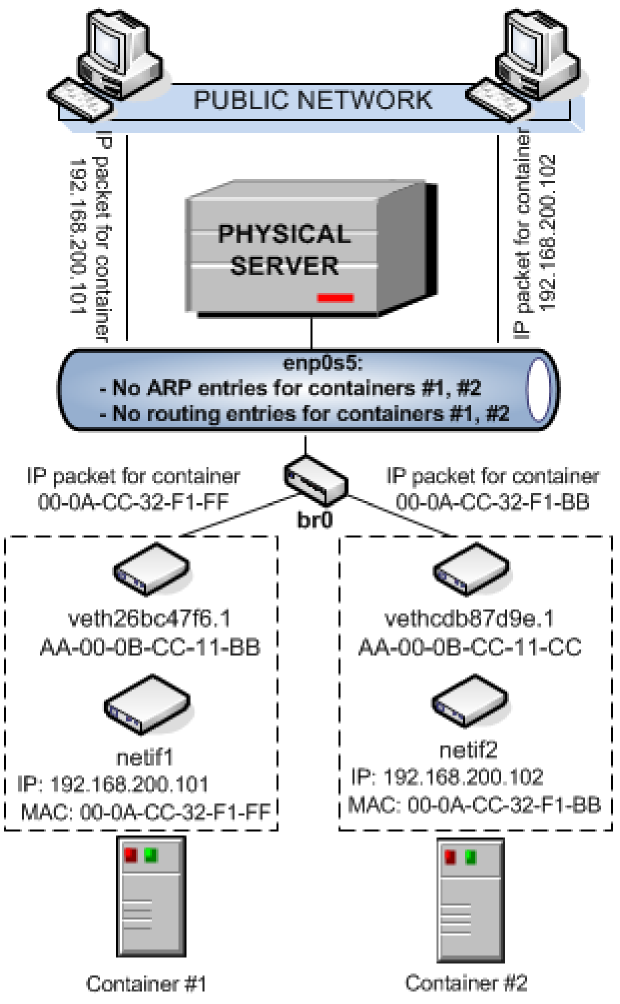
```
# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.200.101  *               255.255.255.255 UH    1000   0        0 venet0
192.168.200.102  *               255.255.255.255 UH    1000   0        0 venet0
192.168.200.103  *               255.255.255.255 UH    1000   0        0 venet0
10.30.0.0        *               255.255.0.0     U     0      0        0 enp0s5
default          virtuozzo.com   0.0.0.0         UG    0      0        0 enp0s5
```

As you can see, the ARP and routing tables contain entries about IP addresses 192.168.200.101, 192.168.200.102, and 192.168.200.103 that belong to containers 1, #2, and 3.

- All container outgoing network traffic goes to the `venet0` adapter and is forwarded via the `enp0s5` physical adapter to the destination, according to the routing table of the server.
- All container incoming network traffic is also processed by the `venet0` adapter. Consider the following situation:
    1. Computer X on the local network wants to send a data packet to container #1 with IP address 192.168.200.101, so it issues an ARP request which computer has this IP address.
    2. The server hosting container #1 replies with its MAC address.
    3. Computer X sends the data packet to the indicated MAC address.
    4. The server receives the packet and transmits it to `venet0` that forwards the packet to container #1.

5.2.1.2 Bridged Mode for Containers

The default network adapter of a container can operate in the host-routed mode only. You can, however, create additional virtual adapters in containers and make them operate in the bridged network mode. The following figure shows an example network configuration where containers #1 and #2 are set to work in the bridged mode.


PUBLIC NETWORK
IP packet for container 192.168.200.101
IP packet for container 192.168.200.102
PHYSICAL SERVER
enp0s5:
- No ARP entries for containers #1, #2
- No routing entries for containers #1, #2
IP packet for container 00-0A-CC-32-F1-FF
IP packet for container 00-0A-CC-32-F1-BB
br0
veth26bc47f6.1
AA-00-0B-CC-11-BB
vethcdb87d9e.1
AA-00-0B-CC-11-CC
netif1
IP: 192.168.200.101
MAC: 00-0A-CC-32-F1-FF
netif2
IP: 192.168.200.102
MAC: 00-0A-CC-32-F1-BB
Container #1
Container #2

In this configuration:

- Container #1 and container #2 have separate virtual adapters consisting of two network interfaces:
  - A `netif<X>` interface in the container (netif1 and netif2 in the figure). This interface represents a counterpart of a physical network adapter installed on a standalone server. Like any other physical adapter, it has a MAC address, can be assigned one or more IP addresses, included in different networks, and so on.
  - A `veth` interface on the hardware node (veth26bc47f6.1 and vethcdb87d9e.1 in the figure). This interface is mostly used to maintain the communication between the hardware node and Ethernet interfaces in containers.

  Note: To simplify things, virtual adapters operating in the bridged mode are called `veth` adapters, though it is not quite correct from the technical point of view.

Both interfaces are closely linked to each other, so a data packet entering one interface always comes out from the other one.

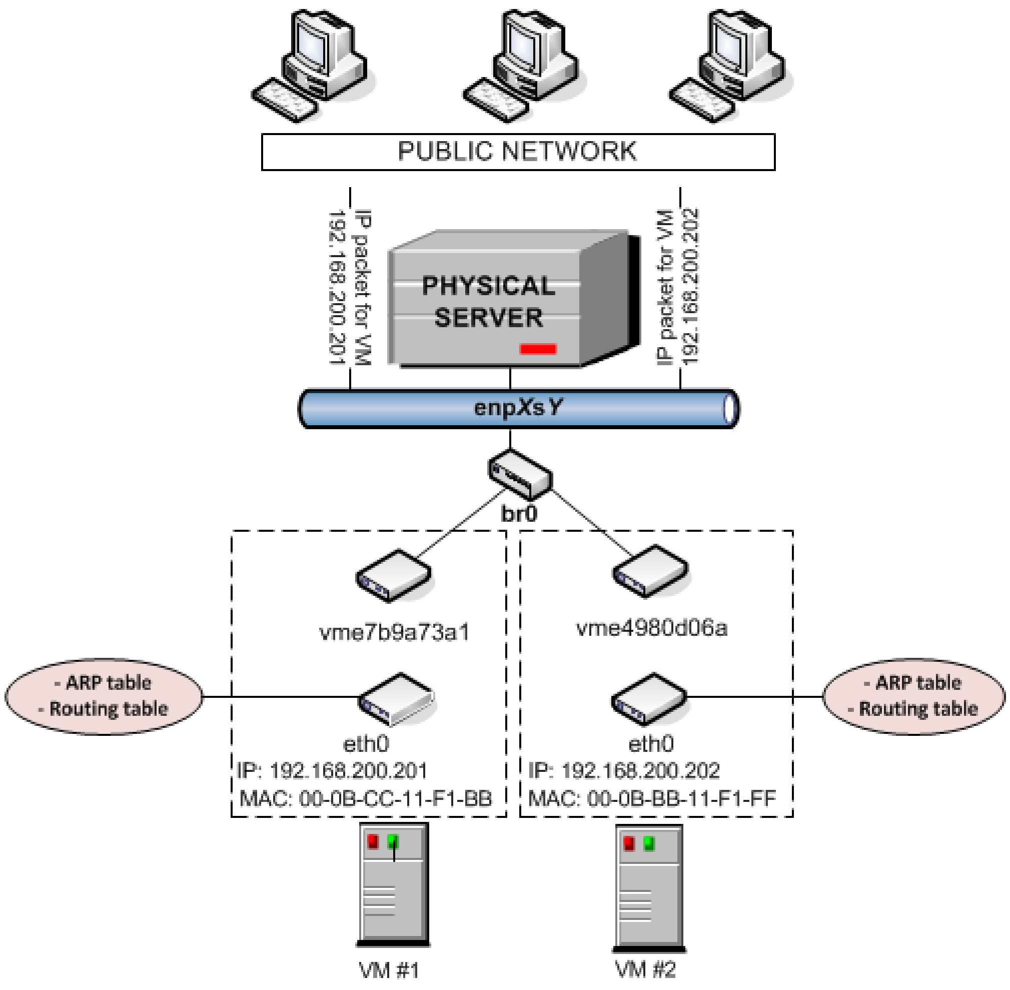
- Containers #1 and #2 keep their own ARP and routing tables that they consult when sending or receiving data.
- The `veth` adapters of both containers are bridged through the bridge `br0` to the physical network adapter `enp0s5`.
- All container outgoing traffic comes via the `veth` adapters to the bridge and are then transmitted through the `enp0s5` physical adapter to the destination, according to the routing tables stored in the containers.
- All incoming data packets for container #1 and #2 reach the `enp0s5` physical adapter first and are then sent through the bridge to the `veth` adapter of the destination container.

5.2.2 Virtual Machine Network Modes

This section describes bridged and host-routed network modes for virtual machines.

5.2.2.1 Bridged Mode for Virtual Machines

By default, a new virtual machine is created with a network adapter that operates in the bridged mode. The figure below demonstrates an example network configuration where two virtual machines, VM #1 and VM #2, are configured to work in the bridged mode.



In this configuration:

- Each virtual machine has a separate virtual adapter that exposes two interfaces: (1) an `ethX` interface in the virtual machine (`eth0` in the figure) and a `vme` interface on the server (`vme7b9a73a1` and `vme4980d06a` in the figure). Both interfaces are closely linked to each other, which means that an IP packet entering one interface always comes out of the other one. An eth adapter has a MAC address, can be assigned one or more IP addresses, belong to different network environments, and so on.

> **Note:** To simplify things, virtual adapters operating in the bridged mode are called `vme` adapters, though it is not quite correct from the technical point of view.

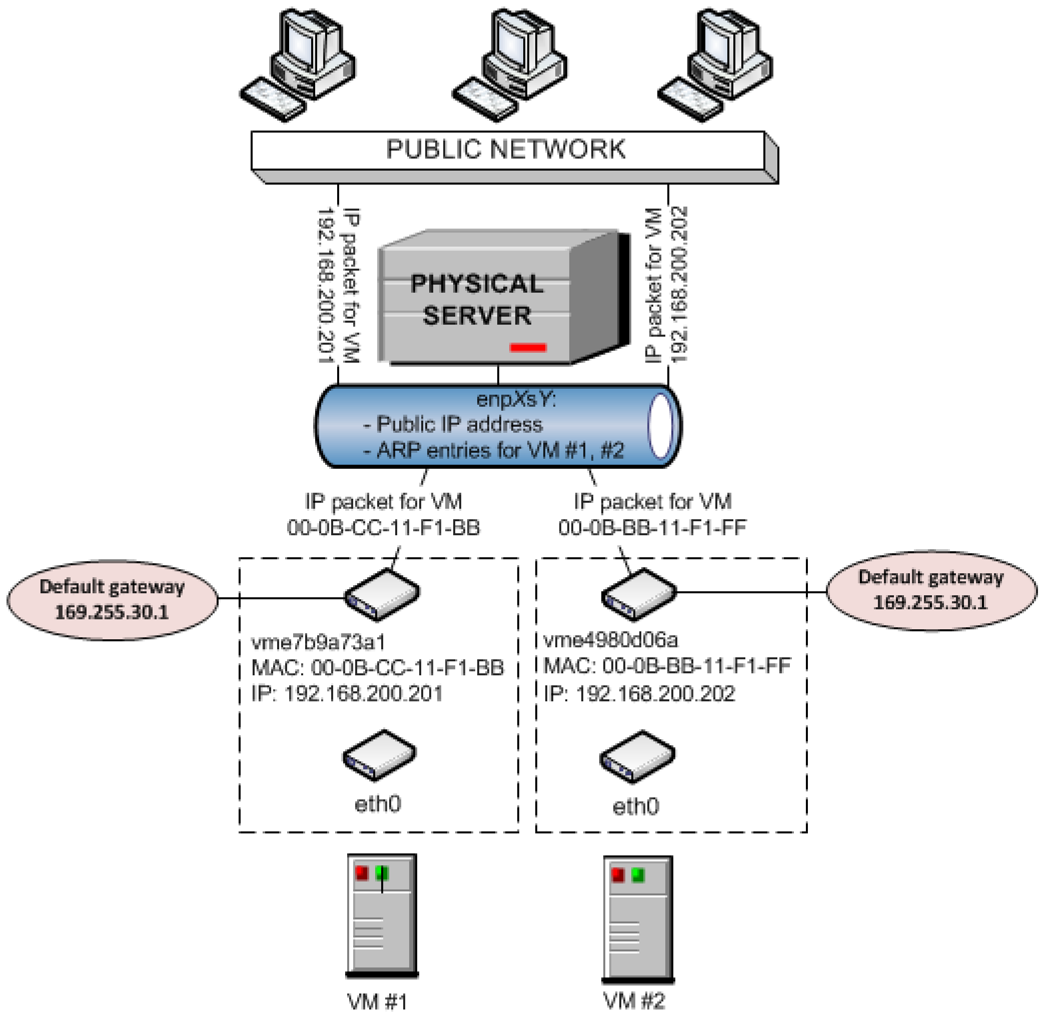
- VM #1 and VM #2 keep their own ARP and routing tables that they consult when sending or receiving data.

- The virtual adapters of both virtual machines are bridged through the bridge `br0` to the physical network adapter `enp0s5`.

- All outgoing data packets are sent from the virtual machines through the bridge and `enp0s5` physical adapter to the destination, according to their routing tables.
- All incoming data packets for VM #1 and VM #2 reach the `enp0s5` physical adapter first and are then transmitted through the bridge to the vme interface of the destination virtual machine.

5.2.2.2 Host-Routed Mode for Virtual Machines

The other network mode a virtual machine can work in is the host-routed mode. The figure below demonstrates an example network configuration where two virtual machines, VM #1 and VM #2, are set to operate in the host-routed mode.


PUBLIC NETWORK
IP packet for VM 192.168.200.201
PHYSICAL SERVER
IP packet for VM 192.168.200.202
enpXsY:
- Public IP address
- ARP entries for VM #1, #2
IP packet for VM 00-0B-CC-11-F1-BB
IP packet for VM 00-0B-BB-11-F1-FF
Default gateway 169.255.30.1
Default gateway 169.255.30.1
vme7b9a73a1
MAC: 00-0B-CC-11-F1-BB
IP: 192.168.200.201
vme4980d06a
MAC: 00-0B-BB-11-F1-FF
IP: 192.168.200.202
eth0
eth0
VM #1
VM #2

In this configuration:

- Each virtual machine also has a virtual adapter exposing two interfaces: an `eth` interface in the virtual machine and a `vme` interface on the server.
- Unlike the bridged mode, the ARP entries for VM #1 and VM #2 are stored on the server rather than in the virtual machines themselves. The server creates these ARP entries and saves them to its ARP table when VM #1 and VM #2 start. You can use the `arp -n` command to view the current ARP entries on a server, for example:

```
# arp -n
Address          HWtype  HWaddress                Flags Mask      Iface
10.30.0.4        ether   00:1a:e2:c7:17:c1        C               eth0
10.30.23.162     ether   70:71:bc:42:f6:a0        C               eth0
192.168.200.201 *        *                        MP              eth0
192.168.200.202 *        *                        MP              eth0
```

- Along with ARP entries, the server also creates routing entries for both virtual machines. So when the server receives a data packet destined for IP address 192.168.200.201, it knows that the packet must be forwarded to the `vme7b9a73a1` interface of VM #1.
- The server handles all incoming traffic for both virtual machines. Consider the following situation:
  1. Computer X on the network wants to send a data packet to VM #1 with IP address 192.168.200.201, so it issues an ARP request which computer has this IP address.
  2. The server replies with its own MAC address.
  3. Computer X sends the data packet to the indicated MAC address.
  4. The `enp0s5` physical adapter receives the packet and routes it to the `vme7b9a73a1` interface of VM #1.
- All outgoing network traffic sent from VM #1 and VM #2 are routed through the default gateway to the `enp0s5` adapter on the server. The default gateway for host-routed virtual machines is automatically assigned the IP address of 169.255.30.1. This special IP address is taken from the Automatic Private IP Addressing (APIPA) range and used exclusively to deliver data packets from virtual machines to the server.

5.2.3 Differences Between Host-Routed and Bridged Network Modes

The bridged network mode demonstrates a number of differences as compared to the host-routed one:

- Each `veth` or `vme` virtual adapter has a MAC address assigned to it while a host-routed adapter does not have any. Thanks to this fact:

    - Any virtual machine or container can see all broadcast and multicast packets received from or sent to the selected network adapter on the hardware node.
    - Using bridged virtual adapters, you can host DHCP or Samba servers in virtual machines and containers.
- There is no more need to assign all network settings (IP addresses, subnet mask, gateway, and so on) to virtual machines and containers from the server. All network parameters can be set from inside virtual machines and containers.
- `veth` and `vme` adapters can be bridged among themselves and with other devices. If several `veth` and `vme` adapters are united into a bridge, this bridge can be used to handle network traffic for the virtual machines and containers whose `veth` and `vme` adapters are included in the bridge.
- Due to the fact that `veth` and `vme` adapters act as full members on the network (rather than “hidden” beyond virtual networks adapters on the server), they are more prone to security vulnerabilities: traffic sniffing, IP address collisions, and so on. Therefore, `veth` and `vme` adapters are recommended for use in trusted network environments only.

5.3 Configuring Virtual Machines and Containers in Host-Routed Mode

You can configure the following parameters of network adapters that operate in the host-routed mode:

- IP addresses and network masks
- DNS servers
- DNS search domains

5.3.1 Setting IP Addresses

The session below shows how to set IP addresses for the virtual machine `MyVM` and the container `MyCT`

```
# prlctl set MyVM --device-set net0 --ipadd 10.0.186.100/24
# prlctl set MyVM --device-set net0 --ipadd 1fe80::20c:29ff:fe01:fb07
# prlctl set MyCT --ipadd 10.0.186.101/24
# prlctl set MyCT --ipadd fe80::20c:29ff:fe01:fb08
```

net0 in the commands above denotes the network card in the virtual machine MyVM to assign the IP address to. You can view all network cards of a virtual machine using the `prlctl list VM_name -i` command. For the container MyCT, you do not need to specify the network card name; `prlctl set` automatically performs the operation on the default adapter that always operates in the host-routed mode.

5.3.2 Setting DNS Server Addresses

To set a DNS server for the virtual machine MyVM and the container MyCT, you can use the following commands:

```
# prlctl set MyVM --device-set net0 --nameserver 192.168.1.165
# prlctl set MyCT --nameserver 192.168.1.165
```

5.3.3 Setting DNS Search Domains

To set a DNS search domain for the virtual machine MyVM and the container MyCT, run these commands:

```
# prlctl set MyVM --device-set net0 --searchdomain 192.168.10.10
# prlctl set MyCT --searchdomain 192.168.10.10
```

Note:

1. You can only configure network settings of virtual machines that have Virtuozzo guest tools installed.
2. Network adapters operating in the routed mode must have at least one static IP address assigned.
3. To be able to assign network masks to containers operating in the venet0 networking mode, set the USE_VENET_MASK parameter in the /etc/vz/vz.conf configuration file to yes.
4. Containers can only have one network adapter operating in the host-routed mode. This adapter is automatically created when you create a container.
5. You can set name servers and search domain in /etc/vz/vz.conf with the NAMESERVER and SEARCHDOMAIN parameters. If set to inherit, the values will be copied from /etc/resolv.conf on the host.

5.3.3.1 Switching Virtual Machine Adapters to Host-Routed Mode

By default, a virtual adapter in any newly created virtual machine starts operating in the bridged mode (see *Connecting Virtual Environments to Virtual Networks* on page 131 for details). To change the current network mode to host-routed, you can run the following command:

```
# prlctl set <VM_name> --device-set Net_ID --type routed
```

For example, to set the `net0` adapter in the virtual machine `MyVM` to operate in the host-routed mode, use this command:

```
# prlctl set MyVM --device-set net0 --type routed
```

5.4 Configuring Virtual Machines and Containers in Bridged Mode

Important: To create and manage virual networks and virtual network bridges, you must use either command-line interface or Virtuozzo Automator, but not both.

This section describes all operations related to configuring virtual machines and containers that operate in bridged mode.

5.4.1 Managing Virtual Networks

A virtual network acts as a binding interface between a virtual network adapter in a virtual machine or container and the corresponding network adapter on the hardware node. Using virtual networks, you can include virtual machines and containers in different networks. Virtuozzo enables you to manage virtual networks as follows:

- Create virtual networks.
- Configure virtual network parameters.
- List existing virtual networks.
- Delete virtual networks.

These operations are described in the following subsections in detail.

5.4.1.1 Creating Virtual Networks

By default, Virtuozzo creates the following virtual networks on the server:

- Bridged virtual network that is connected to one of the physical adapters on the hardware node (e.g., `enp0s5`) and provides virtual machines and containers included in this virtual network with access to the network behind this physical adapter.
- Host-only virtual network that is connected to a special virtual adapter on the server and allows the virtual machines and containers joined to this virtual network to access only the server and the other virtual machines and containers on this network.

You can create your own virtual networks using the `prlsrvctl` command. For example, to create a new virtual network `network1`, you can run:

```
# prlsrvctl net add network1
```

By default, the command creates a host-only virtual network, but you can change its type if needed (see *Configuring Virtual Network Parameters* on page 127).

5.4.1.2 Creating Network Bridges for Network Adapters

To connect a network adapter to a bridged virtual network, you need to create a network bridge first. A network adapter can be physical (`enp<X>s<Y>`) or logical: a VLAN (`enp<X>s<Y>.<N>`) or a bonding interface (`bond<N>`).

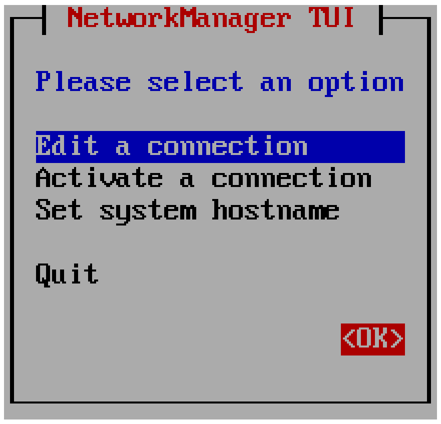
For example, to create a network bridge for the VLAN interface, you can use the NetworkManager text-based user interface tool, `nmtui`, as follows:

1. On the node, start `nmtui`:

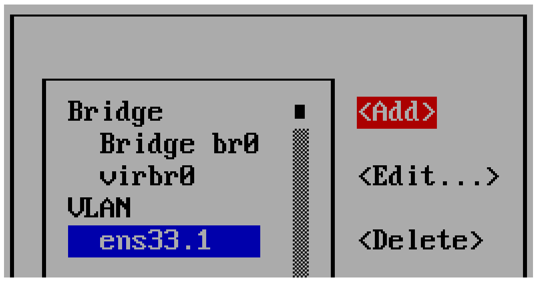
   ```
   # nmtui
   ```

   In the tool TUI, use the arrow keys and **Tab** to navigate through the options, **Enter** to select an option, and **Space** to set and clear check boxes.

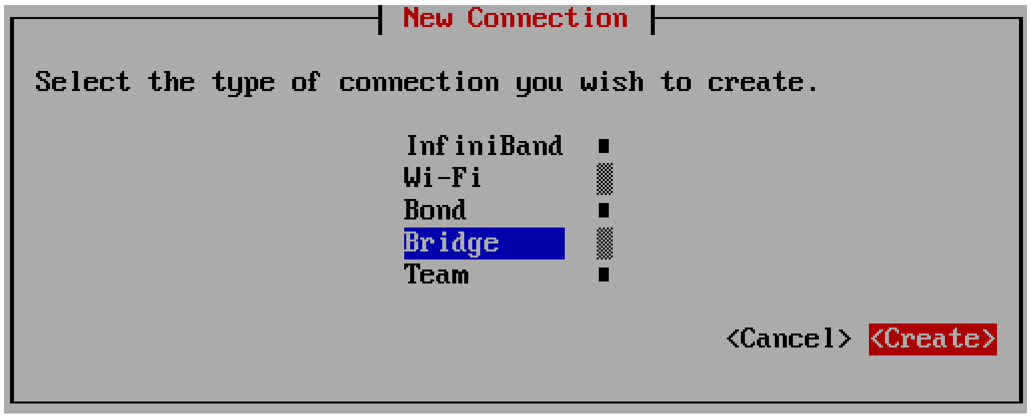
2. On the **NetworkManager TUI** screen, select **Edit a connection** from the menu.



3. On the next screen, select **Add** to add a new connection.



4. To add a new network bridge, choose **Bridge** from the drop-down list in the **New connection** window and press **Create**.



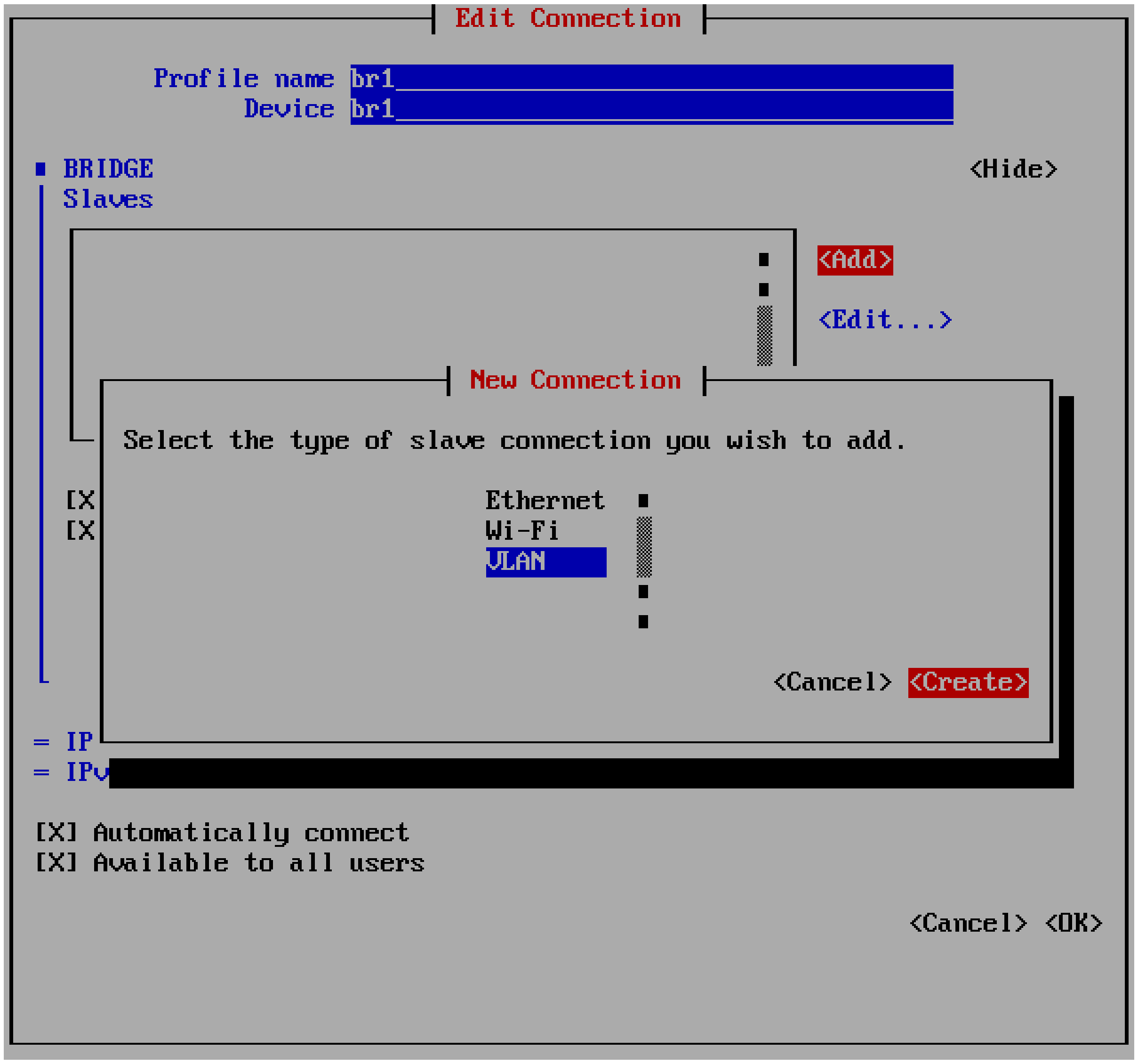
5. On the **Edit connection** screen:

   5.1. In the **Profile name** field, enter the connection profile name. This name with the `ifcfg-` prefix will be used for creating the interface configuration file in the `/etc/sysconfig/network-scripts/` directory.

   5.2. In the **Device** field, specify the device name of the new network bridge.

5.3. Press **Add** to specify a slave network interface.

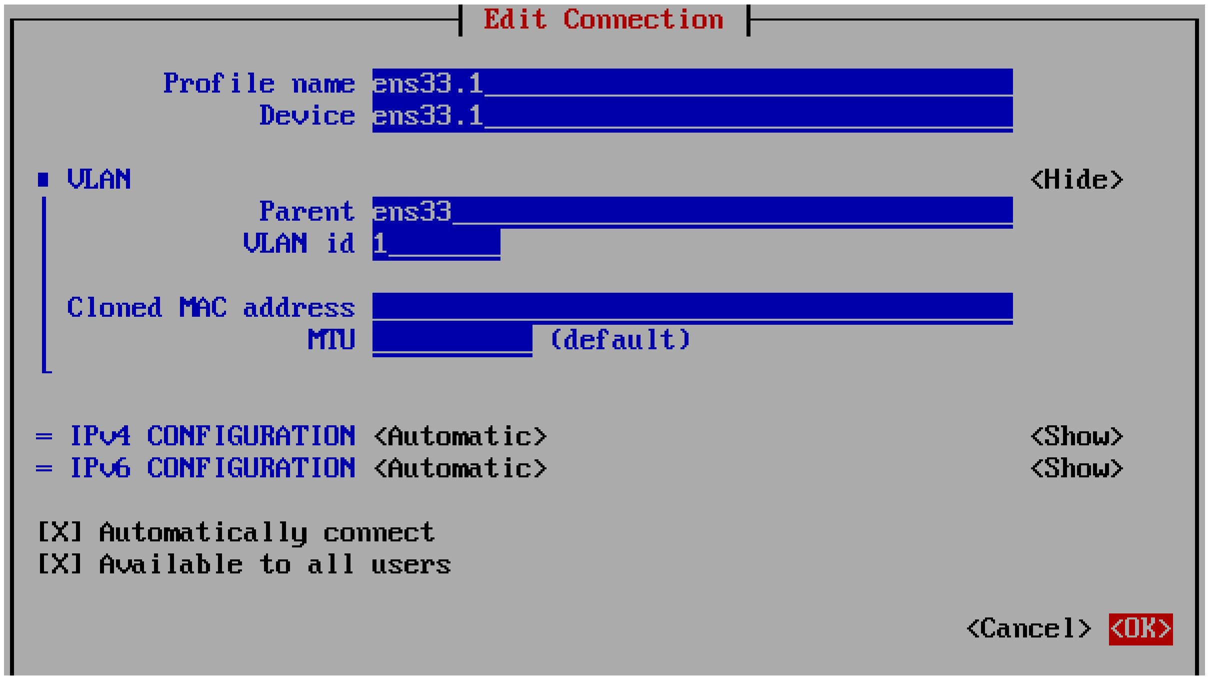
5.4. In the **New connection** window, choose **VLAN** from the drop-down list and press **Create**.



```
                         ┤ Edit Connection ├

        Profile name  br1
              Device  br1

■ BRIDGE                                                  <Hide>
│ Slaves
                                                   <Add>
                                                   <Edit...>
                     ┤ New Connection ├

   Select the type of slave connection you wish to add.

                     Ethernet
                     Wi-Fi
                     VLAN

                                          <Cancel> <Create>

= IP
= IPv

[X] Automatically connect
[X] Available to all users

                                                 <Cancel> <OK>
```

5.5. In the **Edit connection** window, specify the profile name and device name of the VLAN interface in the **Profile name** and **Device** fields, respectively, and press **OK**.

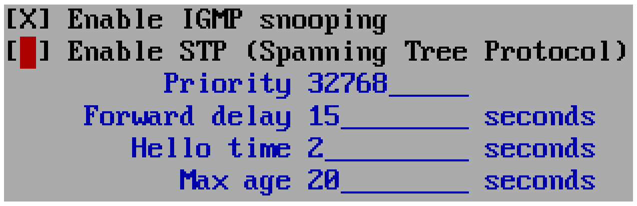
The **Parent** and **VLAN ID** fields are filled in automatically.



```
                    ┤ Edit Connection ├

        Profile name ens33.1
              Device ens33.1

■ VLAN                                              <Hide>
│             Parent ens33
│            VLAN id 1
│
│  Cloned MAC address
│                MTU          (default)
└

═ IPv4 CONFIGURATION <Automatic>                    <Show>
═ IPv6 CONFIGURATION <Automatic>                    <Show>

[X] Automatically connect
[X] Available to all users

                                             <Cancel> <OK>
```

The chosen VLAN interface will appear in the **Slaves** section.

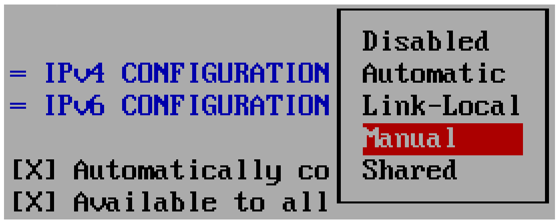
5.6. Clear the **Enable STP (Spanning Tree Protocol)** check box.



```
[X] Enable IGMP snooping
[ ] Enable STP (Spanning Tree Protocol)
          Priority 32768_____
     Forward delay 15________ seconds
        Hello time 2_________ seconds
           Max age 20________ seconds
```

5.7. Configure static IP parameters of the network bridge:

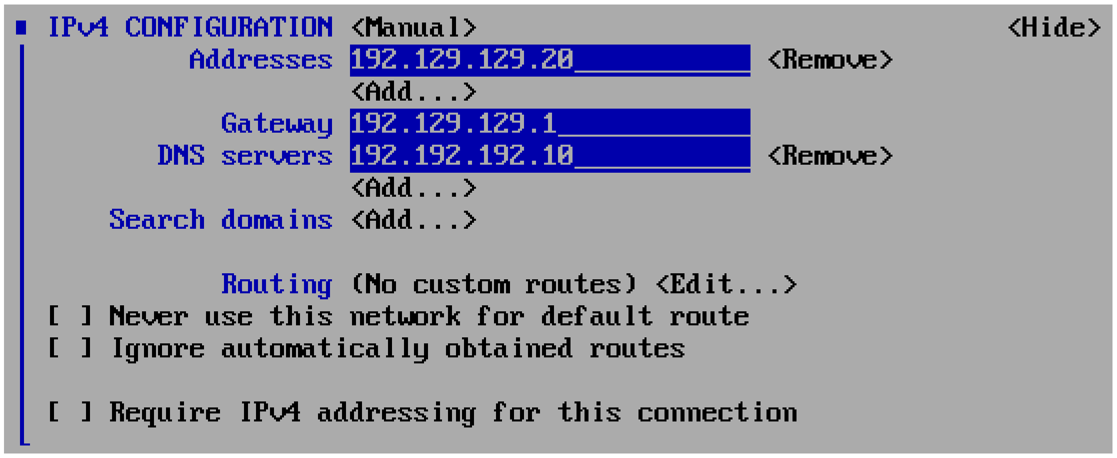
5.7.1. In the **IPv4 CONFIGURATION** section, press **Automatic** and choose **Manual** from the drop-down list.



```
                      Disabled
= IPv4 CONFIGURATION  Automatic
= IPv6 CONFIGURATION  Link-Local
                      Manual
[X] Automatically co  Shared
[X] Available to all
```

5.7.2. Press **Show** to expand the section.

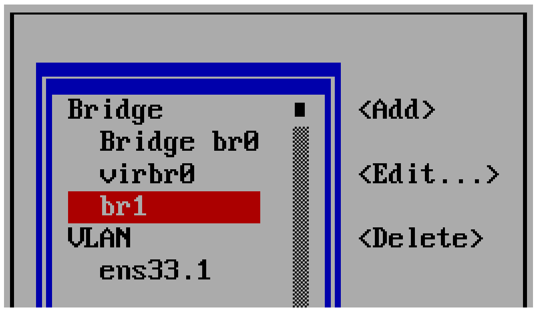
5.7.3. Assign a static IP address, set the default gateway and the DNS server for the network bridge in the corresponding fields.



```
■ IPv4 CONFIGURATION <Manual>                                  <Hide>
           Addresses 192.129.129.20______________ <Remove>
                     <Add...>
             Gateway 192.129.129.1_______________
         DNS servers 192.192.192.10______________ <Remove>
                     <Add...>
      Search domains <Add...>

             Routing (No custom routes) <Edit...>
  [ ] Never use this network for default route
  [ ] Ignore automatically obtained routes

  [ ] Require IPv4 addressing for this connection
```

5.8. Configure other network parameters if required and press **OK**.

The network bridge for the VLAN adapter will appear in the list of existing connections.



```
Bridge          <Add>
  Bridge br0
  virbr0        <Edit...>
  br1
VLAN            <Delete>
  ens33.1
```

6. To exit `nmtui`, press **Back** and then **Quit**.

After creating the network bridge, you can check its configuration file stored in the `/etc/sysconfig/network-scripts/` directory. For example:

```
# cat /etc/sysconfig/network-scripts/ifcfg-br1
```

5.4.1.3 Configuring Virtual Network Parameters

Virtuozzo allows you to configure the following parameters for a virtual network:

- The networking mode in which the virtual network is operating.

> **Note:** Before changing the virtual network type to bridged, a network bridge must be created for the virtual network. See *Creating Network Bridges for Network Adapters* on page 123.

- The description of the virtual network.

All these operations can be performed using the `prlsrvctl` utility. For example, if you need to configure the `network1` virtual network. This virtual network is currently configured as a host-only network and has the following description: `This is a host-only virtual network`. To change these parameters, you can execute the following command:

```
# prlsrvctl net set network1 -t bridged --ifname enp0s6 -d "This is now a bridged \
virtual network"
```

This command configured the `network1` virtual network as follows:

1. Changes the virtual network type to bridged.
2. Changes the virtual network description to the following: "This is now a bridged virtual network".

5.4.1.4 Listing Virtual Networks

To list the virtual networks existing on the hardware node, you can use the `prlsrvctl` utility as shown below.

```
# prlsrvctl net list
Network ID        Type      Bound To       Bridge
Host-Only         host-only                virbr1
Bridged           bridged   enp0s5         br0
```

This utility displays the following information on virtual networks:

Column	Description
Network ID	The name assigned to the virtual network.
Type	The networking mode set for the virtual network.
Bound To	The adapter on the hardware node connected to the virtual networks, if any.

5.4.1.5 Connecting Virtual Networks to Adapters

By connecting an adapter on the physical server to a virtual network, you can join all virtual machines and containers included in the virtual network to the network to which the corresponding adapter is connected.

Consider the following example:

- The `enp0s6` physical adapter and the `network1` virtual network exist on the hardware node. For information on creating virtual networks, see *Creating Virtual Networks* on page 123.
- The `enp0s6` physical adapter is connected to the local network.
- The `br1` network bridge for the `enp0s6` physical adapter is created. For information on creating network

bridges, see *Creating Network Bridges for Network Adapters* on page 123.

- The container MyCT is connected to the network1 virtual network. Detailed information on joining virtual machines and containers to virtual networks is given in *Connecting Virtual Environments to Virtual Networks* on page 131.

To connect the enp0s6 adapter to the network1 virtual network and thus to join the container MyCT to the network behind enp0s6, run this command on the server:

```
# prlsrvctl net set network1 -i enp0s6
```

To check that the enp0s6 physical adapter has been successfully added to the network1 virtual network, you can execute the following command:

```
# prlsrvctl net list
Network ID         Type      Bound To       Bridge
Host-Only          host-only                virbr1
Bridged            bridged   enp0s5         br0
network1           bridged   enp0s6         br1
```

As you can see, the enp0s6 adapter is now joined to the network1 virtual network. That means that the container MyCT whose virtual network adapter is connected to network1 can access the local network behind enp0s6.

5.4.1.6 Deleting Virtual Networks

At any time, you can remove a virtual network that you do not need any more from the physical server. To do this, you can use the prlsrvctl utility. For example, you can delete the network1 virtual network by running the following command:

```
# prlsrvctl net del network1
```

To check that network1 has been successfully removed, execute this command:

```
# prlsrvctl net list
Network ID     Type        Bound To
Host-Only      host-only
Bridged        bridged     enp0s5
```

5.4.2 Managing Virtual Network Adapters in Virtual Environments

Virtuozzo provides you with ample opportunities of configuring virtual network adapters in virtual environments and including them in different network environments. This section shows you the way to perform the following operations:

- Create new virtual network adapters and delete existing ones.
- Configure the parameters of an existing virtual network adapter.
- Join virtual network adapters to virtual networks.

All these operations are described in the following subsections in detail.

5.4.2.1 Creating and Deleting Virtual Adapters

A virtual environment can have up to 15 virtual network adapters. Each adapter can be connected to a different network. For example, if you need to create a new virtual adapter for the virtual machine `MyVM`. To do this, you can execute the following command:

```
# prlctl set MyVM --device-add net
```

To check that the network adapter (`net1`) has been successfully added to the virtual machine, run this command:

```
# prlctl list --info MyVM
ID: {f3b3d134-f512-324b-b0b1-dbd642f5220b}
Name: MyVM
...
net1 (+) dev='vme4208fa77' network='Bridged' mac=001C4208FA77 card=virtio
```

At any time, you can remove the newly created network adapter (`net1`) by executing the following command:

```
# prlctl set MyVM --device-del net1
```

5.4.2.2 Configuring Virtual Adapter Parameters

Virtuozzo allows you to configure the following parameters of virtual adapters:

Configuring MAC Addresses

If you need for some reason to regenerate the current MAC address of a network adapter, you can use the following command:

```
# prlctl set MyVM --device-set net1 --mac 00:1C:42:2D:74:00
```

This command sets the MAC address of `00:1C:42:2D:74:00` for the `net1` adapter in the virtual machine `MyVM`. If do not know what MAC address to assign to your virtual adapter, you can make `prlctl set` automatically generate a new MAC address. To do this, run the following command:

```
# prlctl set MyVM --device-set net1 --mac auto
```

Configuring IP Parameters

As any other standalone server, each virtual environment must have a number of TCP/IP settings configured in the proper way to successfully operate on the network. These settings include:

- IP address
- default gateway
- DNS server

Note: You can configure the network parameters only of virtual machines that have Virtuozzo guest tools installed.

Usually, you define all these settings when you create a virtual environment. However, if you have not yet set any of the settings or want to modify any of them, you can use the `prlctl set` command. For example, you can execute the following command to assign the IP address of `192.129.129.20` to the `net1` adapter in the virtual machine `MyVM`, set the default gateway to `192.129.129.1` and the DNS server to `192.192.192.10`:

```
# prlctl set MyVM --device-set net1 --ipadd 192.129.129.20 --gw 192.129.129.1 \
--nameserver 192.192.192.10
```

Along with a static assignment of network parameters to a virtual adapter, you can make the adapter receive its TCP/IP settings automatically using the Dynamic Host Configuration Protocol (DHCP). For example, you can run this command to make the `net1` adapter in the virtual machine `MyVM` get its IP settings through DHCP:

```
# prlctl set MyVM --device-set net1 --dhcp yes
```

5.4.2.3 Connecting Virtual Environments to Virtual Networks

In Virtuozzo, you can connect virtual environments to virtual networks of the following types:

- Bridged virtual network allows a virtual environment to use one of the physical server network adapters, which makes it appear as a separate computer on the network the corresponding adapter belongs to.
- Host-only virtual network allows a virtual environment to access only the physical server and the virtual environments joined to this network.

By default, any newly created adapter is connected to the Bridged network. To join a virtual machine to another network, use the `prlctl set` command. For example, the following session demonstrates how you can connect the `net0` adapter of the virtual machine `MyVM` to the `network1` virtual network.

Before connecting the virtual machine `MyVM` to the `network1` virtual network, you may wish to check the network adapter associated with this virtual network. You can do it, for example, using the following command:

```
# prlsrvctl net list
Network ID        Type      Bound To
Host-Only         host-only
Bridged           bridged   enp0s5
network1          bridged   enp0s6
```

From the command output, you can see that the `network1` virtual network is attached to the `enp0s6` physical adapter on the physical server. That means that, after connecting the virtual machine `MyVM` to the `network1` virtual network, the virtual machine will be able to access all computers on the network where the `enp0s6` adapter is connected.

Now you can run the following command to join the `net1` adapter of the virtual machine `MyVM` to the `network1` virtual network:

```
# prlctl set MyVM --device-set net1 --network network1
```

To check that the network adapter (`net1`) has been successfully joined to the `network1` virtual network, execute

```
# prlctl list --info MyVM
ID: {f3b3d134-f512-324b-b0b1-dbd642f5220b}
Name: MyVM
...
net1 (+) dev='vme4208fa77' network='network1' mac=001C4208FA77 card=virtio
```

CHAPTER 6

Managing Licenses

This chapter describes how to install, update, view, and transfer licenses on your servers.

Note: If you want to use the Virtuozzo Storage functionality, you need to install a separate license in addition to a Virtuozzo license. For detailed information on managing Virtuozzo Storage licenses, consult the Virtuozzo Storage Administrator's Guide.

6.1 Installing the License

Virtuozzo requires that a different license be installed on each server. You can activate a license during or after the installation. In the latter case, you can install a license from a product key, an activation code, or a license file.

If your Virtuozzo node cannot access the Internet directly, you can activate a license via a proxy server.

6.1.1 Setting Up Proxy Server for License Activation

In situations when a direct Internet connection is not possible, you can specify an HTTP proxy to access the KA server through. To do that, add the HTTP proxy information to the `/etc/vz/vz.conf` file as follows:

```
HTTP_PROXY="http://<host>:<port>/"
HTTP_PROXY_USER="<username>"
HTTP_PROXY_PASSWORD="<password>"
```

Note:

1. You may need to create `/etc/vz/vz.conf` first.
2. The proxy server must have the port 5224 approved for SSL traffic.
3. The `vzlicutils` package is required for license auto-updating to work. If the package is not installed on your system for some reason, you can install it manually with `yum`. To enable license auto-updates after installing the package, launch the `vzlicmon` service with `systemctl start vzlicmon`. The service will show as `/usr/sbin/vzlicmonitor` in the running processes list.

Once the proxy server is set up, proceed to installing the license.

Virtuozzo licenses are updated automatically by default. A few days before the current license expires, the `vzlicmon` service (a part of the `vzlicutils` RPM package) attempts to contact the Virtuozzo KA server over the Internet and obtain a new license.

6.1.2 Installing the License from Product Keys, Activation Codes, or License Files

To install a license from a product key or an activation code, run the `vzlicload -p` command or, to install from a license file, the `vzlicload -f` command. For example:

```
# vzlicload -p <key_or_code>
# vzlicload -f <license_file>
```

When you install a product key or activate a code, a license file is generated and installed in the `etc/vz/licenses/` directory on the server. The difference between the product key and activation code is that the code needs to be activated online, so the server must be connected to the Internet. To activate the code, the installation tool accesses the Key Authentication (KA) licensing server and transmits the specified activation code to it. The KA server generates a license file, sends it back, and the license file is installed on the server automatically.

6.2 Updating the License

You can use the `vzlicupdate` utility to update the license currently installed on the server. When executed, the utility tries to connect to the Key Authentication (KA) server, retrieve a new license, and install it on the server.

To update your license, do the following:

1. Make sure that the host where you wish to update the license is connected to the Internet.
2. Run `vzlicupdate` on the server (your server must have at least one public IPv4 address).

   By default, `vzlicupdate` tries to access the KA server at ka.virtuozzo.com. However, you can explicitly specify what KA server to use using the `--server` option, e.g., `vzlicupdate --server ka.server.com`.

6.2.1 Switching License to a New HWID

If your Virtuozzo license has become invalid due to changed HWID (e.g., after adding or removing a network card), you can have the license switch to the new HWID as follows:

```
# vzlicupdate -t -a <activation_code>
```

6.3 Transferring the License to Another Server

In case you need to transfer a license installed on one server to another server, you can do so as described below.

If you have a Virtuozzo product key:

1. Remove the installed license from the source server with the `vzlicload -r <serial>` command (the serial is shown in `vzlicview` output).
2. Install the product key on the destination server:

   ```
   # vzlicload -p <product_key>
   ```

   A new license file will be generated and installed in the `/etc/vz/license/` directory on the server.

If you have a Virtuozzo activation code:

1. Shut down the source server or remove the installed license from it with the `vzlicload -r <serial>` command (the serial is shown in `vzlicview` output).
2. Make sure that the destination server is up, connected to the Internet, and has at least one public IPv4 address.
3. On the destination server, run

```
# vzlicupdate -t -a <activation_code>
```

When executed, `vzlicupdate` sends the activation code to the KA server. The KA server verifies the code, generates a new license file, sends it back, and the license file is installed on the server automatically.

4. To check that the license transfer has been successful, run `vzlicview`. The information about the newly installed license should be displayed.

6.4 Viewing the License

You can use the `vzlicview` tool to view the information on the installed license and find out its current status. For example:

```
# vzlicview
Searching for installed licenses...
VZSRV
        owner_name="Autogenerated Trial Licenses (Virtuozzo)"
        status="ACTIVE"
        version=7.0
        owner_id=70295522.10134133
        hwid="3D6F.FFB2.1CD5.1E47.1325.BBE0.6C66.ACF5"
        serial="BF3D.E943.18C5.5555.5D10.99D9.0310.DFA0"
        expiration="07/08/2016 03:00:00"
        start_date="06/07/2016 03:00:00"
        issue_date="06/08/2016 16:27:11"
        graceperiod=259200 (259200)
        key_number="PSBM.10134133.0001"
        cpu_total=8 (1)
        ct_total="unlimited" (0)
        architecture="x86"
        architecture="x86_64"
        platform="Linux"
        product="PSBM"
        nr_vms="unlimited" (2)
        subscription="70295522:dd482d1a-5268-4980-ae06-2a288a4fb7ab"
```

You can also view contents of license files with the `vzlicview -f` command. The output does not differ from that in the example above.

The license parameters are described in the following table.

Column	Description
owner_name	The name of the license owner.

Column	Description
status	License status. For description of license statuses, see *License Statuses* on page 137.
version	The version of Virtuozzo for which the license was issued.
owner_id	The ID of the license owner.
hwid	Server hardware identifier.
serial	License serial number. In particular, used to identify license files in `/etc/vz/licenses/`.
expiration	License expiration date, if the license is time-limited.
start_date	The date on which the license becomes active.
issue_date	The date on which the license was issued.
graceperiod	Period, in seconds, during which Virtuozzo continues to function after the license has expired or if the number of running virtual machines and containers exceeds the limit defined by the license.
license_update_date	The date on which the license was last updated.
key_number	Number under which the license is registered on the Key Authentication server.
cpu_total	The total number of physical CPUs that the server is allowed to have.
ct_total	The total number of containers that are allowed to be simultaneously running on the server.
architecture	System architecture with which the license is compatible.
platform	Operating system with which the license is compatible.
product	Name of the product for which the license has been issued.
keyserver_host	The hostname and port of the Key Authentication server.
nr_vms	The number of virtual machines that are allowed to be simultaneously running on the server.
subscription	Virtuozzo feature subscription key.

6.4.1 License Statuses

When viewing information on your licenses, pay special attention to the license status that can be one of the following:

Status	Description

Status	Description
ACTIVE	The license installed on the server is valid and active.
VALID	The license is valid and can be installed on the server.
EXPIRED	The license has expired and cannot be installed on the server.
GRACED	The license is installed on the server but is currently on the grace period because it has expired or the number of running virtual machines and containers exceeds the limit defined by the license.
INVALID	The license is invalid (e.g., because of server architecture mismatch) or corrupt.

CHAPTER 7

Keeping Your System Up To Date

This chapter explains the ways to keep your hardware node up to date. The components you need to take care of are the following:

- Virtuozzo software,
- virtual machines and containers hosted on the server.

To apply major updates that usually include new kernel versions, follow the instructions below:

Note: You can find out if a major update is available by running `yum info virtuozzo-release` and comparing the `Version` values of the installed and available packages. If the versions differ in the third number, for example, `7.0.4` is installed and `7.0.5` is available, a major update has been released.

1. Stop all running virtual environments on the server that is to be updated, or migrate them to other Virtuozzo 7 servers to avoid their downtime.
2. Update the server as described in *Updating All Components* on page 141.
3. Restart the server to update the kernel.
4. Start the virtual environments or migrate them back to the updated server, depending on what you did in step 1.
5. Perform steps 1-4 for other Virtuozzo servers that need to be updated.

   Note: Steps 1-4 can be also performed via Virtuozzo Automator.

6. If you use Virtuozzo Automator, update the container `va-mn` either via VA (see Updating System software) or by running the following commands on the server:

   ```
   # prlctl exec va-mn yum update
   # prlctl restart va-mn
   ```

7. If you use Virtuozzo Storage management panel, update the container `vstorage-ui` by running the following commands on the server:

   ```
   # prlctl exec vstorage-ui yum update
   # prlctl restart vstorage-ui
   ```

7.1 Updating Virtuozzo

Important: It is strongly recommended to have every machine run the same version of Virtuozzo. At the very least make sure that product versions are only one major update apart. For example, before you update a machine to Virtuozzo 7.0 Update 7, make sure that all other machines in the cluster already run Update 6. The reason is that VMs created on newer nodes may fail to start on older nodes, thus causing issues with migration, restore from backup, high availability, and such.

Virtuozzo allows quick and easy updates with the `yum` tool standard for RPM-compatible Linux operating systems. The main components you may need to update are the following:

- tools and libraries,
- EZ templates,
- kernel,
- ReadyKernel patch,
- KVM/QEMU hypervisor in running virtual machines.

Note: To see which updates are available before installing them, you can run `yum check-update`.

7.1.1 Updating All Components

The easiest way to update all components of the Virtuozzo software is to run the `yum update` command. `yum` will do the following:

1. Access Virtuozzo repositories.
2. Check for available updates for tools, libraries, EZ templates, Virtuozzo kernel, and the latest ReadyKernel patch for the current kernel. If a new Virtuozzo kernel is available, check for the corresponding ReadyKernel patch as well.
3. Install the updates on your system.

7.1.2 Updating Kernel

Updating the Virtuozzo kernel requires updating the `vzkernel` and `vzkernel-devel` packages:

```
# yum update vzkernel vzkernel-devel
```

7.1.3 Updating KVM/QEMU Hypervisor in Virtual Machines

Virtuozzo can update KVM/QEMU hypervisor live in running virtual machines that have KVM/QEMU version 2.6.0 or newer.

To do this, install the `vz-qemu-engine-updater` package and update the `qemu-kvm-vz` package:

```
# yum install vz-qemu-engine-updater
# yum update qemu-kvm-vz
```

Updating `qemu-kvm-vz` starts a 10-minute timer (to give `yum` time to complete the operation), after which the `vz-qemu-engine-updater` tool is started and begins updating KVM/QEMU in each running virtual machine, one at a time.

Immediate updating may not be possible in several cases:

- `yum` is currently locked on the node (in this case, no VMs can be updated automatically until the lock is released),
- a VM is changing states (e.g., from `running` to `stopped`),
- configuration changes are being applied to a VM,

- a backup of a VM is being created, or
- any other `prlctl` operation is executed on a VM.

The KVM/QEMU updater will skip such VMs and queue them for a later update. The updater will perform a set number of retries to update VMs that have been skipped, each retry after a set delay. If retries are exhausted or the update fails for some reason, the virtual machine is left running with the outdated KVM/QEMU.

To manually disable automatic updates, mask the updater service:

```
# systemctl mask vz-qemu-engine-updater.service
```

To re-enable updates, unmask the updater service:

```
# systemctl unmask vz-qemu-engine-updater.service
```

To check if virtual machines have been successfully updated, view the log file:

```
# journalctl -u vz-qemu-engine-updater
<...>VM MyVM (was using qemu-kvm-vz-2.6.0-28.3.6.vz7.30) has been successfully updated.
<...>Finished updating VMs.
<...>Successfully updated QEMU engine for all running VMs.
```

To configure the number of retries, the delay between them, and other parameters, refer to the `vz-qemu-engine-updater.json` man page and edit the `/var/lib/vz-qemu-engine-updater.json` configuration file.

7.1.3.1 Updating KVM/QEMU Hypervisor Manually

If, for some reason, the KVM/QEMU hypervisor was not automatically updated in a running VM, you can do it manually as follows:

1. Make sure that the VM uses an outdated version of the hypervisor.

   Check the version installed on the node:

   ```
   # rpm -qa qemu-kvm-vz
   qemu-kvm-vz-<version>.x86_64
   ```

   And compare it with the version the VM currently uses:

   ```
   # virsh qemu-monitor-command <VM_name> '{"execute":"query-version"}'
   <...>"package":" (qemu-kvm-vz-<version>)"}<...>
   ```

2. Make sure no `prlctl` operations are being executed on the VM.

3. Update the hypervisor that the VM uses:

```
# prlctl update-qemu <VM_name>
```

7.1.4 Updating EZ Templates

You can update an EZ template like any other RPM package using the `yum update` command. For example:

```
# yum update centos-6-x86_64-ez
...
Updated:
  centos-6-x86_64-ez.noarch 0:4.7.0-1
Complete!
```

Note:

1. Updating an OS EZ template requires that you append `ez` to template name.
2. You can also use the `vzpkg update template` command to update EZ templates.

7.2 Updating Virtuozzo Kernel with ReadyKernel

Virtuozzo ReadyKernel is a kpatch-based service shipped with Virtuozzo 7 and available out-of-the-box on hardware nodes with active licenses. ReadyKernel offers a more convenient, rebootless alternative to updating the kernel the usual way and allows you not to wait for scheduled server downtime to apply critical security updates. ReadyKernel enables you to receive cumulative kernel patches that fix critical security issues and apply these patches without having to reboot the server. ReadyKernel updates are released for Virtuozzo kernels younger than 18 months. When a kernel becomes older, it should be updated, e.g., to the latest one, so you can keep receiving ReadyKernel updates.

Upon installation, the patches are loaded into server RAM and immediately applied to the kernel. If the server reboots, these patches are reapplied to the kernel on boot.

If later you install a new kernel or major kernel update that requires a reboot, the downloaded patches will remain on the server but will not be applied.

Note: At any time, you can check the details of the applied ReadyKernel patch with `readykernel info`.

ReadyKernel patches can be received and installed automatically or manually as described in the following sections.

7.2.1 Installing ReadyKernel Patches Automatically

If automatic updating was not disabled during the installation, ReadyKernel will check for new patches daily at 12:00 server time. If a patch is available, ReadyKernel will download, install, and load it for the current kernel.

If automatic updating is disabled, you can re-enable it with the following command:

```
# readykernel autoupdate enable <hour>
```

The service will check for patches daily at the specified `<hour>` (set in 24-hour format, server time) by means of the `cron.d` script.

To disable automatic updating, run

```
# readykernel autoupdate disable
```

7.2.2 Managing ReadyKernel Patches Manually

7.2.2.1 Dowloading, Installing, and Loading ReadyKernel Patches

To download, install, and instantly load the latest ReadyKernel patch for the current kernel, do the following:

1. Check for new ReadyKernel patches:

   ```
   # readykernel check-update
   ```

2. If a new patch is available, download, install, and instantly load it for the current kernel by running:

   ```
   # readykernel update
   ```

   Note: You can also do this with `yum update`.

ReadyKernel patches are cumulative, i.e. the latest patch includes all the previous ones. To keep the kernel secure, you only need to install and load the latest patch.

7.2.2.2 Loading and Unloading ReadyKernel Patches

To manually load the latest installed ReadyKernel patch to the kernel, do one of the following:

- If an older patch is already loaded, unload it first, then load the latest patch by running:

  ```
  # readykernel load-replace
  ```

- If no older patches are loaded, load the latest patch by running:

  ```
  # readykernel load
  ```

To unload the patch from the current kernel, run

```
# readykernel unload
```

7.2.2.3 Installing and Removing ReadyKernel Patches for Specific Kernels

If multiple kernels are installed on the server, you can install a ReadyKernel patch for a specific kernel:

```
# yum install readykernel-patch-<kernel_version>
```

To remove a specific ReadyKernel patch from the server, run

```
# yum remove readykernel-patch-<kernel_version>
```

7.2.2.4 Downgrading ReadyKernel Patches

If you experience problems with the latest ReadyKernel patch, you can downgrade it to an older version if one is available.

To downgrade a patch for the current kernel to the previous version, run

```
# yum downgrade readykernel-patch-$(uname -r)
```

To downgrade a patch for a specific kernel to the previous version, run

```
# yum downgrade readykernel-patch-<kernel_version>
```

You can run these commands multiple times to downgrade to the patch version you need. Alternatively, you can downgrade a patch to a specific version by specifying the desired patch version. For example:

```
# yum downgrade readykernel-patch-12.7-0.4-17.vl7
```

7.2.3 Disabling Loading of ReadyKernel Patches on Boot

If for some reason you do not want ReadyKernel patches to be applied at boot time, run the following command:

```
# readykernel autoload disable
```

To re-enable automatic loading of ReadyKernel patches on boot, run

```
# readykernel autoload enable
```

7.2.4 Managing ReadyKernel Logs

ReadyKernel logs event information in `/var/log/messages` and `/var/log/kpatch.log`. You can specify logging parameters for the latter in the configuration file `/etc/logrotate.d/kpatch`. For more information on parameters you can use, see the `logrotate` man page.

7.3 Updating Software in Virtual Machines

To keep software in your virtual machines up to date, you can use the same means you would use on standalone computers running the corresponding operating systems:

- In Linux-based virtual machines, you can use the native Linux updaters (`up2date`, `yum`, or `yast`).
- In Windows-based virtual machines, you can use the native Windows updaters (e.g., the Windows Update tool).

7.3.1 Updating Virtuozzo Guest Tools in Virtual Machines

Starting from Virtuozzo 7.0.4 (Update 4), Virtuozzo guest tools in virtual machines are updated automatically via a weekly `cron` job that starts the `vz-guest-tools-updater` tool.

The following requirements must be met:

- The `vz-guest-tools-updater` package must be installed on the node.
- The virtual machine must have the `--tools-autoupdate` parameter set to `on` (this is the default behavior).

Note: Starting from Virtuozzo 7.0.7 (Update 7), guest tools are also automatically installed in virtual machines on their next start (see *Installing Virtuozzo Guest Tools* on page 18). To disable automatic installation of guest tools, set the `InstallTools` parameter to `false` in the `/etc/vz/tools-update.conf` configuration file.

The `vz-guest-tools-updater` tool builds a list of VMs with the enabled `--tools-autoupdate` parameter and outdated guest tools. After that, a 5-minute timer triggers simultaneous guest tools update in a configurable number of VMs. If an update attempt fails, the tool will queue that VM for another try. If the second attempt fails, the VM's guest tools will be left outdated.

Warning: During the update, Virtuozzo guest tools image is forcibly mounted to VM's optical disk drive even if it is already in use.

Windows virtual machines need to be restarted to complete the update of guest tools. On every such update, administrators inside these VMs receive a reboot notification upon login or immediately if they are logged in.

You can configure the number of VMs whose guest tools are to be updated simultaneously by changing the value of the `MaxVMs` parameter in the `/etc/vz/tools-update.conf` configuration file.

To check the update status of guest tools in one or more VMs, use the `--get-state` option for the `vz-guest-tools-updater` tool and specify VM names in a sequence. For example:

```
# vz-guest-tools-updater --get-state <VM1_name> [<VM2_name> ...]
```

If the guest tools in the given virtual machine are up to date, the command output will be as follows:

```
{<VM_UUID>} (<VM_name>): Tools are up to date
```

To disable automatic updating of Virtuozzo guest tools for a VM, run the following command:

```
# prlctl set <VM_name> --tools-autoupdate off
```

To manually update guest tools in one or more VMs, start the `vz-guest-tools-updater` script by specifying VM names in a sequence. For example:

```
# vz-guest-tools-updater <VM1_name> [<VM2_name> ...]
```

7.4 Updating Containers

Virtuozzo provides two means of keeping your containers up to date:

- Updating EZ templates software packages inside a particular container by means of the `vzpkg` utility. Using this facility, you can keep any of the containers existing on your hardware node up to date.
- Updating caches of the OS EZ templates installed on the hardware node. This facility allows you to create new containers already having the latest software packages installed.

7.4.1 Updating EZ Template Packages in Containers

Virtuozzo allows you to update packages of the OS EZ template a container is based on and of any application EZ templates applied to the container. You can do it by using the `vzpkg update` utility. Assuming that the container `MyCT` is based on the `centos-6-x86_64` OS EZ template, you can issue the following command to update all packages included in this template:

```
# vzpkg update 26bc47f6-353f-444b-bc35-b634a88dbbcc centos-6-x86_64
...
  Updating: httpd                ### [1/4]
  Updating: vzdev                ### [2/4]
  Cleanup : vzdev                ### [3/4]
  Cleanup : httpd                ### [4/4]
Updated: httpd.i386 0:2.0.54-10.2 vzdev.noarch 0:1.0-4.swsoft
Complete!
Updated:
 httpd                  i386      0:2.0.54-10.2
 vzdev                  noarch    0:1.0-4.swsoft
```

Note:

1. Updating EZ templates is supported for running containers only.
2. If you are going to update the cache of a commercial OS EZ template (e.g., Red Hat Enterprise Server 5 or SLES 10), you should first update software packages in the remote repository used to handle this OS EZ template and then proceed with updating the EZ template cache.

As you can see from the example above, the `httpd` and `vzdev` applications have been updated for the `centos-6-x86_64` OS EZ template. If you wish to update all EZ templates (including the OS EZ template) inside the container `MyCT` at once, execute this command:

```
# vzpkg update 26bc47f6-353f-444b-bc35-b634a88dbbcc
...
Running Transaction
  Updating  : hwdata               #### [1/2]
  Cleanup   : hwdata               #### [2/2]
Updated: hwdata.noarch 0:1.0-3.swsoft
```

```
Complete!
Updated:
 hwdata                 noarch    0:0.158.1-1
```

In the example above, only the `hwdata` package inside the container `MyCT` was out of date and updated to the latest version.

7.4.2 Updating OS EZ Template Caches

With the release of new updates for the corresponding Linux distribution, the created OS EZ template cache can become obsolete. Virtuozzo allows you to quickly update your OS EZ template caches using the `vzpkg update cache` command.

Note: If you are going to update the cache of a commercial OS EZ template (e.g., Red Hat Enterprise Server 6 or SLES 11), you should first update software packages in the remote repository used to handle this OS EZ template and then proceed with updating the EZ template cache.

When executed, `vzpkg update cache` checks the `cache` directory in the template area (`/vz/template/cache` by default) on the hardware node and updates all existing tarballs in this directory. However, you can explicitly indicate the tarball for what OS EZ template should be updated by specifying the OS EZ template name. For example, to update the tarball for the `centos-6-x86_64` OS EZ template, run this command:

```
# vzpkg update cache centos-6-x86_64
Loading "rpm2vzrpm" plugin
Setting up Update Process
Setting up repositories
base0            100% |=========================|  951 B    00:00
base1            100% |=========================|  951 B    00:00
base2            100% |=========================|  951 B    00:00
base3            100% |=========================|  951 B    00:00
...
```

Upon the `vzpkg update cache` execution, the old tarball name gets the `-old` suffix (e.g., `centos-x86.tar.gz-old`).

You can also pass the `-f` option to `vzpkg update cache` to remove an existing tar archive and create a new one instead of it.

If the `vzpkg update cache` command does not find a tarball for one or several OS EZ templates installed on the server, it creates tar archives of the corresponding OS EZ templates and puts them to the `/vz/template/cache` directory.

CHAPTER 8

Managing High Availability Clusters

This chapter explains managing high availability (HA) for servers that participate in Virtuozzo Storage clusters.

High availability keeps virtual machines, containers, and iSCSI targets operational even if the node they are located on fails. In such cases, the affected virtual environments continue working on other, healthy nodes in the cluster. High availability is ensured by:

- Metadata redundancy. For a Virtuozzo Storage cluster to function, not all but just the majority of MDS servers must be up. By setting up multiple MDS servers in the cluster you will make sure that if an MDS server fails, other MDS servers will continue controlling the cluster.
- Data redundancy. Copies of each piece of data are stored across different storage nodes to ensure that the data is available even if some of the storage nodes are inaccessible.

  Note: The redundancy is achieved by one of two methods: replication or erasure coding (for details, see Understanding Data Redundancy).

- Monitoring of node health.

You may need to follow different instructions in this chapter based on your scenario which can be one of these:

- You use Virtuozzo Storage with CLI management, you have not enabled HA for virtual machines and containers by means of the client server role during installation, and you want to enable it.

  In this case, continue reading this chapter.

- You use Virtuozzo Storage with CLI management, you have enabled HA for virtual machines and containers by means of the client server role during installation, and you want to change the default resource relocation mode.

  In this case, HA in the DRS mode is automatically enabled for virtual machines and containers on the node. To change the resource relocation mode, e.g., to round-robin, follow the instructions in *Configuring Resource Relocation Modes* on page 154.

- You use Virtuozzo Storage with GUI management, you have not assigned iSCSI network roles or created S3 clusters yet in the Virtuozzo Storage management panel, and you want to configure HA manually (e.g., to manually choose a resource relocation mode).

  In this case, continue reading this chapter.

- You use Virtuozzo Storage with GUI management, you have assigned the network role “ISCSI” to a node’s network interface or joined a node to an S3 cluster in the Virtuozzo Storage management panel, and you want to change the default resource relocation mode.

  In this case, HA in the round-robin mode is automatically enabled for virtual machines and containers on the node. To change the resource relocation mode, e.g., to DRS, follow the instructions in *Configuring Resource Relocation Modes on Nodes Participating in S3 or iSCSI Export* on page 156.

8.1 Prerequisites for High Availability

For the high availability feature to work, the following prerequisites must be met:

- A Virtuozzo Storage cluster must be set up in your network. High availability is supported only for nodes joined to Virtuozzo Storage clusters.
- The Virtuozzo Storage cluster must have 5 or more nodes.
- Virtual machines and containers residing on a node must be stored in the cluster:
  - If you use Virtuozzo Storage with GUI management, log in to the management panel, create datastores, and place your virtual machines and containers in them as described in the Virtuozzo Storage Administrator’s Guide.
  - If you use Virtuozzo Storage with CLI management, virtual machines and containers are automatically configured to be stored in the cluster on nodes for which the **Client Server Role** was chosen during installation. You can also configure them manually.

- The chosen redundancy mode must protect your data against a simultaneous failure of two or more nodes. To see the list of redundancy modes and their differencies, refer to Understanding Data Redundancy.
    - If you use Virtuozzo Storage with GUI management, you select the redundancy mode while creating datastores.
    - If you use Virtuozzo Storage with CLI management, you need to change the default redundancy parameters for the directories storing your virtual machines and containers.

      For example, to set the 3:2 replicas mode for the `vmprivate` and `private` directories (with VMs and containers, respectively) in the cluster `stor1`, run the following commands:

      ```
      # vstorage set-attr -R /vstorage/stor1/vmprivate replicas=3
      # vstorage set-attr -R /vstorage/stor1/private replicas=3
      ```

      Note: It is not recommended to use the erasure coding redundancy mode for Virtuozzo virtual machines and containers.

- (Virtuozzo 7.0.6 and newer) Each node in the cluster must have the `SERVER_UUID` parameter set in the `/etc/vz/vz.conf` file. Virtuozzo Storage requires it to provide access to container disks. For the parameter description, see the Virtuozzo 7 Command Line Reference.

8.2 Enabling and Disabling High Availability on Nodes

To enable high availability on a node means to enable it for all virtual machines and containers on this node that are stored in the Virtuozzo Storage cluster.

8.2. Enabling and Disabling High Availability on Nodes

Note:

1. When you assign the network role "ISCSI" to a node's network interface or join a node to an S3 cluster in the Virtuozzo Storage management panel, high availability is automatically enabled for virtual machines and containers on this node in the round-robin mode.
2. When you enable HA for virtual machines and containers for the client server role during installation of Virtuozzo Storage with CLI management, high availability is automatically enabled for virtual machines and containers on this node in the DRS mode.

To enable high availability on a node, do the following:

1. Update Virtuozzo Storage to the latest version with `yum update`.
2. Run the `hastart` script:

   ```
   # hastart -c <cluster> -n <storage_network/network_mask>
   ```

   where `<cluster>` is the cluster name, e.g., `vstor1`, `<storage_network>` is the cluster internal network, and `<network_mask>` covers all the nodes in the cluster.

   The script will automatically install, configure, and start the HA services on the node as well as add the node to the HA configuration.

After enabling HA for the node, you can check the result with the `shaman stat` command.

8.2.1 Disabling High Availability for Specific Virtual Machines and Containers

By default, if high availability is enabled on a node, it affects all virtual machines and containers on said node. If necessary, you can disable HA for specific virtual machines and containers using the `prlctl set` command. For example:

```
# prlctl set MyVM --ha-enable no
```

To re-enable HA support, run:

```
# prlctl set MyVM --ha-enable yes
```

8.2.2 Enabling High Availability for iSCSI Targets

Note: If you use Virtuozzo Storage with GUI management, enable HA for iSCSI targets by assigning the network role "ISCSI" to a node's network interface. Keep in mind that doing so also enables HA for virtual machines and containers on this node that are stored in the Virtuozzo Storage cluster.

If you use Virtuozzo Storage with CLI management, high availability for iSCSI targets is disabled by default. To enable it on a cluster node, do the following:

1. If not yet done, enable HA on the node using the `hastart` script as described in the previous section.
2. Add the iSCSI `shaman` role to the node. For example, on a node in the cluster `vstor1`, run:

   ```
   # shaman -c vstor1 add-role ISCSI
   ```

If you want HA to work only for iSCSI targets, change the `shaman` roles on the node by running the following command:

```
# shaman -c vstor1 set-roles ISCSI
```

8.2.3 Disabling High Availability on Nodes

Disabling HA on a node disables it for all virtual environments and iSCSI targets on this node that are stored in the Virtuozzo Storage cluster.

To disable HA on a node, do the following:

1. Disable and stop HA services:

   ```
   # systemctl disable shaman.service
   # systemctl stop shaman.service
   # systemctl disable pdrs.service
   # systemctl stop pdrs.service
   ```

2. Remove the node from the HA configuration. For example, for a node in the cluster `vstor1`, run:

   ```
   # shaman -c vstor1 leave
   ```

8.3 Configuring Resource Relocation Modes

You can configure how the cluster will deal with situations when a node fails. Three modes are available:

8.3. Configuring Resource Relocation Modes

- DRS (default). In this mode, virtual machines and containers which were running on a failed node are relocated to healthy nodes based on available RAM and license capacity. This mode can be used for nodes on which the `pdrs` service is running.

  Note: If CPU pools are used, virtual machines and containers can only be relocated to other nodes in the same CPU pool. For details, see *Managing CPU Pools* on page 158.

  The DRS mode works as follows. The master DRS continuously collects the following data from each healthy node in the cluster via SNMP:

  - total node RAM,
  - total RAM used by virtual machines,
  - total RAM used by containers,
  - maximum running virtual machines allowed,
  - maximum running containers allowed,
  - maximum running virtual machines and containers allowed.

  If a node fails, the `shaman` service sends a list of virtual machines and containers which were running on that node to the master DRS that sorts it by most required RAM. Using the collected data on node RAM and licenses, the master DRS then attempts to find a node with the most available RAM and a suitable license for the virtual environment on top of the list (requiring the most RAM). If such a node exists, the master DRS marks the virtual environment for relocation to that node. Otherwise, it marks the virtual environment as broken. Then the master DRS processes the next virtual environment down the list, adjusting the collected node data by the requirements of the previous virtual environment. Having processed all virtual environments on the list, the master DRS sends the list to the `shaman` service for actual relocation.

- Spare. In this mode, virtual machines and containers from a failed node are relocated to a spare node—an empty node with enough resources and a license to host all virtual environments from any given node in the cluster. Such a node is required for high availability to work in this mode.

  Before switching to this mode, make sure the spare node is added to the HA configuration and has no resources (virtual machines, containers, iSCSI targets, and S3 clusters) stored on it. To check this, run the `shaman stat` command on any node in the cluster and check that **RESOURCES** column shows zeroes for the node:

```
# shaman stat
Cluster 'stor1'
Nodes: 3
Resources: 12

   NODE_IP       STATUS     ROLES                       RESOURCES
*  10.10.20.1    Active     VM:QEMU,CT:VZ7,ISCSI,S3  0 CT, 0 S3, 0 VM, 0 ISCSI
 M 10.10.20.2    Active     VM:QEMU,CT:VZ7,ISCSI,S3  4 CT, 1 S3, 3 VM, 0 ISCSI
   10.10.20.3    Active     VM:QEMU,CT:VZ7,S3        1 CT, 1 S3, 1 VM, 1 ISCSI
```

In the example above, the current node (marked by the asterisk) is empty and can be used as spare.

If the node is not empty, you can free it:

- from VMs and containers by migrating them to the other cluster nodes,
- from iSCSI targets by unregistering them on said node and registering them on the other cluster nodes,
- from S3 resources by releasing said node from the S3 cluster.

Once you have a spare node in your cluster, you can switch to the spare mode by running:

```
# shaman set-config RESOURCE_RELOCATION_MODE=spare
```

- Round-robin (default fallback). In this mode, virtual machines, containers, and iSCSI targets from a failed node are relocated to healthy nodes in the round-robin manner. To switch to this mode, run:

```
# shaman set-config RESOURCE_RELOCATION_MODE=round-robin
```

Additionally, you can set a fallback relocation mode in case the chosen relocation mode fails. For example:

```
# shaman set-config RESOURCE_RELOCATION_MODE=drs,spare
```

8.4 Configuring Resource Relocation Modes on Nodes Participating in S3 or iSCSI Export

Important: Follow instructions in this section only if you use Virtuozzo Storage with GUI management in the scenario described below. Otherwise skip this section.

- If you use Virtuozzo Storage with GUI management and have assigned the network role “ISCSI” to a node’s network interface in the Virtuozzo Storage management panel, high availability is automatically enabled for virtual machines, containers, and iSCSI targets on this node in the round-robin resource relocation mode.

- If you use Virtuozzo Storage with GUI management and have joined a node to an S3 cluster in the Virtuozzo Storage management panel, high availability is automatically enabled for virtual machines, containers, and S3 resourses on this node in the round-robin resource relocation mode.

Note: All resource relocation modes are described in *Configuring Resource Relocation Modes* on page 154.

If you are fine with the round-robin mode, you do not need to configure anything else on the node. If, however, you want to change the resource relocation mode, e.g., to DRS, log in to the node via SSH, and run the `hastart` script as described in *Enabling and Disabling High Availability on Nodes* on page 152.

After executing the script, the HA mode will automatically change to DRS. If you need to change it again, follow the instructions in *Configuring Resource Relocation Modes* on page 154.

8.5 Configuring HA Priority for Virtual Machines and Containers

High availability priority defines which virtual machines and containers will be relocated first if the node they are located on fails. The higher is priority, the higher is the chance a virtual machine or container has to be relocated to a healthy node, if the Virtuozzo Storage cluster does not have enough disk resources.

By default, all newly created virtual machines and containers have the priority set to 0. You can use the `prlctl set` command to configure the default priority of a virtual machine or container, for example:

```
# prlctl set MyVM1 --ha-prio 1
# prlctl set MyVM2 --ha-prio 2
```

These commands set the HA priority for the `MyVM1` and `MyVM2` virtual machines to 1 and 2, respectively. If the node where these VMs are located fails, `MyVM2` will be the first to relocate to a healthy node, followed by `MyVM1` and then by all other virtual machines that have the default priority of 0.

8.6 Managing CPU Pools

Warning: This feature is experimental. Libvirt may not be aware of new CPU features that may already be used in CPU pools. This may lead to issues with migration to destination nodes that do not have these unreported CPU features.

In Virtuozzo, you can avoid stopping virtual environments on a node (e.g., for node maintenance) by temporarily migrating them live to another node. For live migration to be possible, the CPUs on the source and destination nodes must be manufactured by the same vendor, and the CPU features of the destination node must be the same or exceed those of the source node.

Two issues may arise from this requirement. First, if the target node has more CPU features than the source node, live migration back to the source server will not be possible. Second, if a node in a high-availability cluster fails and its virtual environments are relocated to another node, that destination node may have a CPU from a different vendor or with a different set of features that will prevent live migration back to the original node when it goes up again.

CPU pools solve these two issues by dividing your Virtuozzo nodes into groups (pools) in which live migration between any two nodes is always guaranteed. This is achieved by determining CPU features common for all nodes in a pool and masking (disabling) the rest of the CPU features on nodes that have more of them. So a CPU pool is a group of nodes with equal CPU features.

Note: Adding nodes with same CPUs to different CPU pools does not prevent live migration between such nodes.

8.6.1 Adding Nodes to CPU Pools

Note: Nodes with CPUs from different vendors cannot be added to same CPU pools.

A node that is to be added to a CPU pool must not have running virtual machines and containers on it. To meet this requirement while avoiding virtual environment downtime, you can migrate all running virtual machines and containers live to a different node (and migrate them back live after the node has been added to a pool).

8.6. Managing CPU Pools

The easiest way to add a node to a CPU pool is to run the following command on it:

```
# cpupools join
```

The node will be added to a default CPU pool.

Default pools have the following features and limitations:

- The naming pattern is `default_{intel}N`, e.g., `default_intel0`, `default_intel1`, etc.
- A preset, unchangeable basic CPU mask provides maximum hardware compatibility at the expense of advanced CPU features. Different CPU feature masks are used for different CPU vendors.
- Nodes which do not support the basic CPU feature mask are placed in different default CPU pools, e.g., `default_intel1`, `default_intel2`, etc.
- Nodes cannot be added to specific default CPU pools on purpose.

To make sure that as many common CPU features as possible are enabled for nodes in a pool for best performance, you can move the required nodes to a custom CPU pool. To do this:

1. On the node to be added to a custom CPU pool, run the `cpupools move` command. For example:

   ```
   # cpupools move mypool
   ```

   The node will be moved to the CPU pool `mypool`. If the CPU pool does not exist, it will be created.

   Note: Custom CPU pools are created with the same basic CPU feature mask as default pools.

2. On any node in the new pool, run the `cpupools recalc` command to update the CPU feature mask and make sure that as many common CPU features as possible are enabled. For example:

   ```
   # cpupools recalc mypool
   ```

Now that node is in the desired CPU pool, you can migrate the node's virtual machines and containers back live.

The general recommendation is to group nodes with CPUs of the similar microarchitecture, generation, or family as they have similar features. This way most of the CPU features will remain available for nodes after applying the CPU feature mask to the pool. This approach will help ensure the best possible performance for nodes and at the same time guarantee live migration compatibility.

8.6.2 Monitoring CPU Pools

To see which CPU pools exist in your cluster and which nodes are in them, run the `cpupools stat` command on any node in the cluster. For example:

```
# cpupools stat
default_intel0:
320117e17894401a
bec9df1651b041d8
eaea4fc0ddb24597
mypool:
ca35929579a448db
* f9f2832d4e5f4996
```

The identifiers listed are Virtuozzo Storage node IDs which you can obtain with the `shaman -v stat` command. For example:

```
# shaman -v stat
Cluster 'vstor1'
Nodes: 5
Resources: 1
   NODE_IP        STATUS    NODE_ID               RESOURCES
   10.29.26.130   Active    bec9df1651b041d8      0 CT
*  10.29.26.134   Active    f9f2832d4e5f4996      0 CT
   10.29.26.137   Active    ca35929579a448db      0 CT
   10.29.26.141   Active    320117e17894401a      0 CT
 M 10.29.26.68    Active    eaea4fc0ddb24597      1 CT
...
```

Note: The asterisk marks the current node (on which the command has been run).

8.6.3 Removing Nodes from CPU Pools

To remove the current node from a CPU pool, run the `cpupools leave` command on it:

```
# cpupools leave
```

8.7 Monitoring Cluster Status

You can use the `shaman top` and `shaman stat` commands to monitor the overall status of a cluster and cluster resources. The `shaman stat` command shows a snapshot of the current cluster status and clustering resources while `shaman top` displays a dynamic real-time view of the cluster. The following shows an example output of the `shaman stat` command:

```
# shaman stat
Cluster 'vstor1'
Nodes: 3
Resources: 8

        NODE_IP          STATUS         RESOURCES
      M 10.30.24.176    Active         0 CT, 0 VM, 1 ISCSI
     *  10.30.25.33     Active         1 CT, 3 VM, 1 ISCSI
        10.30.26.26     Active         0 CT, 0 VM, 2 ISCSI

        CT ID    STATE     STATUS       OWNER_IP         PRIORITY
        101      stopped   Active       10.30.25.33      0

        VM NAME  STATE     STATUS       OWNER_IP        PRIORITY
        vm1      stopped   Active       10.30.25.33     0
        vm2      running   Active       10.30.25.33     0
        vm3      stopped   Active       10.30.25.33     0

        ISCSI ID                STATE    STATUS   OWNER_IP         PRIORITY
iqn.2014-04.com.pstorage:ps1    running  Active   10.30.24.176     0
iqn.2014-04.com.pstorage:ps2    running  Active   10.30.25.33      0
iqn.2014-04.com.pstorage:ps3    running  Active   10.30.26.26      0
```

The table below explains all output fields:

Field	Description
Cluster	Name of the cluster. The “M” next to the server denotes that the server is the main server in the cluster (called the master server), and the asterisk (*) indicates the server where the `shaman stat` command was executed.
Nodes	Number of servers with enabled HA support in the cluster.
Resources	Number of resources `shaman` keeps control of.
NODE_IP	IP address assigned to the server.
STATUS	Server status. It can be one of the following: • Active. The server is up and running and controlled by `shaman`. • Inactive. The server is up and running, but `shaman` does not control the server (e.g., the `shamand` service is stopped).
RESOURCES	Resources hosted on the server.
CT ID	Container ID.
VM NAME	Virtual machine name.

Field	Description
ISCSI ID	iSCSI target name.
STATE	Denotes whether the virtual machine/container is running or stopped.
STATUS	HA status of the virtual machine, container, or iSCSI target (as reported by `shaman`). It can be one of the following: • Active. Healthy virtual machines, containers, or iSCSI targets hosted in the cluster. • Broken. Virtual machines, containers, or iSCSI targets that could not be relocated from a failed server to a healthy one. • Pool. Virtual machines, containers, or iSCSI targets waiting for relocation from a failed server to a healthy one.
OWNER_IP	IP address of the server where the virtual machine, container, or iSCSI target is hosted.
PRIORITY	Current priority of the virtual machine/container. For details, see *Configuring HA Priority for Virtual Machines and Containers* on page 157.

The output of the `shaman top` command is similar to that of `shaman stat`. Additionally, you can use keyboard keys to change the command output on the fly. The following keys are supported:

- g: Group or ungroup resources by their status.
- v: Show or hide additional information.
- ENTER or SPACE: Update the screen.
- q or Esc or CTRL-C: Quit.
- h: Show the help screen.

8.8 Managing Cluster Resources with Scripts

Virtuozzo Storage comes with a number of scripts used by the `shaman` utility to manage and monitor cluster resources. There are two types of scripts:

- Common scripts. The common scripts are located in the `/usr/share/shaman` directory and used by the `shaman` utility to call resource-specific scripts.

- Resource-specific scripts. For each common script, there are one or more resource-specific scripts. Resource-specific scripts are peculiar to each cluster resource and located in separate subdirectories. For virtual machines and containers, these directories are `/usr/share/shaman/vm-` and `/usr/share/shaman/ct-`, respectively. Resource-specific scripts are used to perform various actions on cluster resources.

The following example describes the process of using the relocate script:

1. The `shaman-monitor` daemon checks at regular intervals whether some virtual machines and containers require relocation (usually, when a server fails and the virtual machines and containers hosted on it should be relocated to a healthy server).
2. If some virtual machines or containers are scheduled for relocation, `shaman-monitor` calls the common script `/usr/share/shaman/relocate`.
3. The common script `relocate` calls the scripts `/usr/share/shaman/vm-/relocate` and `/usr/share/shaman/ct-/relocate` to relocate the virtual machines and containers to a healthy server.

If necessary, you can customize any of the scripts to meet your demands.

For the full list of scripts and their descriptions, see the `shaman-scripts` man page.

CHAPTER 9

Hardening Your Virtuozzo Server

Similar to other hypervisor-based technologies, the health of a Virtuozzo server is critical to ensure the uptime of all resident virtual environments.

This chapter provides information on best practices necessary to harden a Virtuozzo server and ensure its continuous uptime.

9.1 Update Policy

Keeping Virtuozzo up to date is critical for the security and reliability of the system. To mitigate any potential security risks, Virtuozzo software must be updated in a timely manner.

Software packages that have not been updated for a significant period of time can result in security vulnerabilities and other serious functionality issues compromising the entire system.

For more details, see *Keeping Your System Up To Date* on page 139.

9.2 Audit Policy

The audit policy defines the significant events which need to be logged on server. Logs have two important roles: provide a means for near-real-time monitoring of the system and allow you to investigate past actions. When considering system security, audit events will often identify unauthorized attempts to access resources. The events originate from interactive user sessions or system processes and services.

As defined by the Filesystem Hierarchy Standard (FHS), events are logged to files which reside in the `/var/log` directory. Files that you need to pay attention to are listed in the table:

File	Description	How to examine
`/var/log/lastlog`	Records of each user's last login	`lastlog`
`/var/log/messages`	System messages from `syslogd`	`cat /var/log/messages`
`/var/log/wtmp`	Records of all logins and logouts	`who /var/log/wtmp`

9.2.1 Storing Logs Remotely

It is recommended to store logs remotely. This will let you detect intrusion even if an attacker gained root privileges and modified local logs to hide their presence. You can change log location by configuring the `rsyslogd` daemon.

For example, you can add the following lines to the end of the `/etc/rsyslog.conf` configuration file:

```
kern.warning;*.err;authpriv.none\t@<remote_host>
*.info;mail.none;authpriv.none;cron.none\t@<remote_host>
*.emerg\t@<remote_host>
local7.*\t@<remote_host>
```

where `<remote_host>` is the FQDN of the destination server where logs need to be stored.

9.2.2 Viewing Critical Audit Messages

The most important security messages are tracked by `syslog authpriv` and stored in the `/var/log/secure` log file by default. It tracks all attempts to access the computer from a local interactive logon, network logon, network servce startups, change of privileges, etc. Failed logon attempts may show a trend for password attacks. Successful logon messages are important for identifying which user logged on at a given time.

9.3 Mount Policy

The mount policy can be defined by mount options that can help you prevent unexpected usage of files. These options are listed in the table:

Option	Description
`noexec`	Do not allow direct execution of any binaries on the mounted file system.
`nodev`	Do not interpret character or block special devices on the file system.
`nosuid`	Do not allow set-user-identifier or set-group-identifier bits to take effect.
`nouser`	Forbids an ordinary (i.e., non-root) user to mount the file system.

You can add these mount options to corresponding partitions in `/etc/fstab`. For example, the `noexec` option can be applied to the `/tmp` partition, while all of the above options can be applied to removable media mounts (CDROMs, DVDROMs, floppy drives, USB memory cards, etc.).

9.4 Service Policy

To be able to log in to your Virtuozzo server for administration purposes, make sure that services listed in the table are enabled on the server.

Service	Description
`network`	Provides network connectivity for the Virtuozzo server itself and virtual environments residing on it.
`sshd`	Most of the Virtuozzo servers reside in datacenters and are managed remotely.
`crond`	Virtuozzo uses a number of cron-based tools for periodical checking and reporting of system health parameters.
`rsyslogd`	System events logging.
`prl-disp-service`	Virtuozzo management service.
`libvirtd`	Performs management tasks on virtual environments.

The following best practices apply:

`sshd`:

- Configure your SSH daemon to use protocol version 2.
- Prohibit remote root login as most attacks are performed to this account. Login as a non-privileged user and switch to the root credentials using `sudo` package if required.
- Prohibit authentication based on `hosts` and `rhosts` as they are known to be vulnerable.

`rsyslogd`:

- Do not use remote logging over UDP protocol.
- Use TCP transport and SSH tunnel for remote logging, if packets pass through an untrusted network.

`prl-disp-service`:

- Block the remote access to `prl-disp-service` if you do not use virtual environment migration, remote backup/restoration, or remote access to Virtuozzo servers via `prlctl` or Virtuozzo SDK.

Additionally, it is recommended to have only hardware-related services running on your Virtuozzo server. For example, you can run `smartd` or `snmpd` on the server, but make sure to isolate services like web or mail servers inside virtual environments in case they are attacked.

9.5 Account Policy

It is recommended to minimize the number of accounts in the host OS to make it more secure.

The general recommendations for all Linux distributions are:

- create a non-privileged account for performing non-privileged tasks in the system;
- use `sudo` for performing privileged tasks;
- disable remote root logon, use a non-privileged user for this;
- disable system user logon;
- force periodical password changes;
- disable accounts after a number of login failures.

9.6 Networking Policy

A Virtuozzo server should be isolated from the Internet. It should only have an internal IP address and firewall policies applied to it. To still be able to update the license and install updates from Virtuozzo repositories, you can set up a proxy server and follow the instructions in *Setting Up Proxy Server for License Activation* on page 133.

Virtuozzo enables Linux kernel firewall during installation. To see the list of ports opened by default, refer to the Virtuozzo 7 Installation Guide.

CHAPTER 10

Advanced Tasks

This chapter collects miscellaneous configuration and management tasks, some of which require a deeper knowledge of Linux and Virtuozzo and should be performed with caution.

10.1 Configuring Automatic Memory Management Policies

Virtuozzo offers several techniques for automatically managing VM and container memory. The management is performed by the `vcmmd` daemon by means of policies described below. Setting the correct policy may provide performance and density benefits to your Virtuozzo installation.

The following policies are currently available:

- `performance` (default). Best for hardware nodes with lots of small virtual environments (relative to a NUMA node size).

  Important: It is strongly recommended to avoid overcommitting resources when the `performance` policy is enabled. For resource overcommit, switch to the `density` policy.

- `density`. Best for hardware nodes with memory overcommit.
- `NoOpPolicy`. This policy is set automatically if no license is installed on a hardware node or if the current license is expired.

If you activated a license after installing Virtuozzo or updated an expired license, you need to restart the `vcmmd` daemon to apply changes. To do this:

1. Stop all virtual environments on the host or temporarily migrate them live to another host.
2. Run the following command:

```
# systemctl restart vcmmd
```

3. Start the virtual environments or migrate them back to the server, depending on what you did in step 1.

After restarting `vcmmd`, the default `performance` policy will be applied.

You can switch between the policies with the `prlsrvctl` tool.

Note: Before setting a VCMMD policy, stop all virtual machines and containers on the host or temporarily migrate them live to another host.

For example, to switch to the `density` policy, do the following:

1. Make sure that there are no running virtual machines or containers on the host:

```
# prlctl list
UUID                                        STATUS    IP_ADDR      T   NAME
```

2. Set the policy with `prlsrvctl set --vcmmd-policy`:

```
# prlsrvctl set --vcmmd-policy density
```

To check the currently enabled policy, run

```
# prlsrvctl info | grep "policy"
Vcmmd policy: density config=density
```

10.1.1 Optimizing Virtual Machine Memory with Kernel Same-Page Merging

To optimize memory usage by virtual machines, Virtuozzo uses a feature of Linux called Kernel Same-Page Merging (KSM). The KSM daemon `ksmd` periodically scans memory for pages with identical content and merges those into a single page. Said page is marked as copy-on-write (COW), so when its contents are changed by a virtual machine, the kernel creates a new copy for that virtual machine.

KSM enables the host to:

- avoid swapping due to merging of identical pages;
- run more virtual machines;
- overcommit virtual machine memory;

- speed up RAM and hence certain applications and guest operating systems.

In Virtuozzo 7, KSM is managed by `vcmmd` and work with both `performance` and `density` policies. However, with `performance` policy enabled, KSM merges identical pages for each NUMA node independently which slows its operation down by as many times as the number of NUMA nodes on the host (e.g., by four times if there are four NUMA nodes).

Note: For more details, see *Configuring Automatic Memory Management Policies* on page 168.

10.1.2 Managing Host Services with VCMMD

In Virtuozzo 7, you can set memory limits and guarantees managed by `vcmmd` to host services. The daemon will take into account the service memory while allocating resources to virtual environments.

Note: The setting is removed on hardware node reboot.

For example, to have a service named `service` managed by `vcmmd`, do as follows:

1. Place the service processes in `/sys/fs/cgroup/memory/service.slice` directory.
2. Register `service.slice` in `vcmmd` and set the desired memory parameters. For example:

   ```
   # vcmmdctl register SRVC service.slice --guarantee 100000000 --limit 100000000 --swap 100000
   ```

3. Activate the service:

   ```
   # vcmmdctl activate service.slice
   ```

To check if the service is managed with `vcmmd`, run

```
# vcmmdctl list
name                                        type  active  guarantee     limit    swap
<...>
service.slice                               SRVC     yes      97656     97656      97
```

You can change the service memory parameters at runtime using the `vcmmdctl update` command.

For example, to change the `service.slice` parameters, run:

```
# vcmmdctl update service.slice --guarantee 200000000 --limit 200000000
```

To check if the parameters have been correctly applied, run

```
# vcmmdctl list
name                                        type  active  guarantee     limit    swap
```

```
<...>
service.slice                             SRVC     yes     195312    195312     max
```

You can unregister the service from vcmmd at runtime with the `vcmmdctl unregister` command.

10.1.3 Managing Virtuozzo Storage Services with VCMMD

For Virtuozzo Storage hardware nodes, `vstorage.slice/vstorage-services.slice` is automatically created in the memory cgroup on hardware node start.

If you deployed Virtuozzo Storage on top of a Virtuozzo server and started the `vstorage` services manually without rebooting the server, you need to restart the `vcmmd` daemon to apply changes. To do this:

1. Stop all virtual environments on the host or temporarily migrate them live to another host.
2. Run the following command:

   ```
   # systemctl restart vcmmd
   ```

3. Start the virtual environments or migrate them back to the server, depending on what you did in step 1.

After restarting `vcmmd`, `vstorage.slice/vstorage-services.slice` will be created in the memory cgroup.

To check if `vstorage` is managed with `vcmmd`, run

```
# vcmmdctl list
name                                      type  active  guarantee     limit    swap
<...>
vstorage.slice/vstorage-services.slice    SRVC     yes    4194304 103854973       0
```

You can temporarily change memory limits and guarantees for `vstorage` services at runtime using the `vcmmdctl update` command. For example:

```
# vcmmdctl update vstorage.slice/vstorage-services.slice --guarantee 100000000 --limit 100000000
```

The parameters will be reset to the default values on the next `vcmmd` restart or node reboot.

To configure limits and guarantees for `vstorage` services permanently, do as follows:

1. Edit the memory parameters in the `/etc/vz/vstorage-limits.conf` file. (Set `-1` for unlimited memory.)
2. Restart `vcmmd` to apply changes:

   ```
   # systemctl restart vcmmd
   ```

10.2 Creating Customized Containers

If you wish to use custom applications in multiple identical containers, you can create containers with necessary applications already preinstalled and tuned to meet your demands.

Virtuozzo offers several ways to create customized containers with preinstalled applications:

- From a golden image (an OS EZ template cache with preinstalled application templates).
- From a custom OS EZ template that specifies a custom application package list.
- From a custom configuration sample file that specifies custom application EZ templates.

10.2.1 Using Golden Image Functionality

The golden image functionality allows you to preinstall application templates to OS EZ template caches to speed up creating multiple containers based on the same set of OS and application templates. Previously, you could either install application templates to each container after creating it or embed them directly into a custom OS template. Golden image is currently the easiest and fastest way to create containers with preinstalled applications.

The best way to create such a cache is:

1. Make a custom sample configuration file with information on the OS EZ template to cache and application EZ templates to preinstall. For example:

   ```
   # cp /etc/vz/conf/ve-basic.conf-sample \
   /etc/vz/conf/ve-centos-6-x86_64-mysql-devel.conf-sample
   ```

   Note: If you already have a custom sample configuration file with application EZ templates specified in it, you can reuse it instead of creating a new one.

2. Add the OS EZ template and application EZ template information to the new configuration file. Each OS and application template name must be preceded by a dot. Multiple consecutive application EZ template names must be separated by white spaces. For example:

   ```
   # cd /etc/vz/conf
   # echo OSTEMPLATE=".centos-6-x86_64" >> ve-centos-6-x86_64-mysql-devel.conf-sample
   # echo TEMPLATES=".mysql .devel" >> ve-centos-6-x86_64-mysql-devel.conf-sample
   ```

3. Run the `vzpkg create appcache` command with your configuration file as an option. For example:

```
# vzpkg create appcache --config centos-6-x86_64-mysql-devel
```

Note: If the resulting application cache already exists, it will not be recreated and you will see a corresponding message. To recreate an application cache, use the `vzpkg update appcache` command.

The resulting archive can be found in the `/vz/template/cache` directory on the hardware node. You can check that it exists and includes necessary application templates with the following command:

```
# vzpkg list appcache
centos-6-x86_64             2012-07-20 16:51:36
     mysql
     devel
```

10.2.1.1 Disabling Golden Image Functionality

The Golden Image functionality is enabled by default in the `/etc/sysconfig/vz/vz.conf` global configuration file. Should you wish to disable it, do one of the following:

- Set the `GOLDEN_IMAGE` option to `no` in the Virtuozzo global configuration file. The Golden Image functionality will be disabled globally.
- Set the `GOLDEN_IMAGE` option to `no` in the container sample configuration file. The Golden Image functionality will be disabled for commands that use this specific sample configuration file.
- Create a file named `golden_image` containing `no` in the OS EZ template's configuration directory. The Golden Image functionality will be disabled for this specific OS EZ template.
- Create a file named `golden_image` containing `no` in the application template's configuration directory. The Golden Image functionality will be disabled for this specific application template, so it will not be preinstalled into any OS EZ template caches.

10.2.2 Using Customized EZ Templates

You can create custom OS and application templates tailored to your needs. In such a template, you only need to specify parameters that differ from those in the default template. All other parameters—that are not explicitly set in the custom template—are inherited from the corresponding default template.

To create a custom template, do the following:

1. If required, install the default OS template on the hardware node. For example:

   ```
   yum install centos-7-x86_64-ez
   ```

2. Create a directory for your template at the location where the default template directory is. For example, for a custom CentOS 7 64-bit template `mytmpl`, create the directory `/vz/template/centos/7/x86_64/config/os/mytmpl`.

3. If you are creating a custom OS template, specify repositories. For example, copy the file `mirrorlist` from the default template directory to your template directory:

   ```
   # cp /vz/template/centos/7/x86_64/config/os/default/mirrorlist \
   /vz/template/centos/7/x86_64/config/os/mytmpl
   ```

4. In your template directory, create the file `packages` listing the RPMs you need, one per line. For example,

   ```
   systemd
   yum
   ```

   Note: The minimal list of packages to include in a custom template may vary depending on guest OS. For example, CentOS 7 templates require that `systemd` be specified in the `packages` file for the `prlctl enter` command to work on resulting containers.

5. Optionally, change more template parameters according to your needs (for a description of parameters, see the next section).

Your custom template is ready. In this example, it is an OS template that contains `systemd`, `yum`, and all their prerequisites. You now can create containers based on it. For example:

```
# prlctl create MyCT --vmtype ct --ostemplate centos-7-x86_64-mytmpl
```

If you created an application template, you now can add it to the container configuration file as described in *Using Golden Image Functionality* on page 172.

10.2.2.1 EZ Template Configuration Files

All EZ templates are stored in `/vz/template`, in subdirectories named according to OS name, version, and architecture. For example, `/vz/template/centos/7/x86_64`. Each template includes a set of configuration files stored in the `/config/os/<template_name>` subdirectory (OS templates) or the `/config/app/<template_name>` subdirectory (application templates).

The following files can be in the template configuration subdirectory:

- `ct2vm` – Container to virtual machine migration script.
- `description` – Detailed information on the EZ template.
- `distribution` – OS templates only. The name of the Linux distribution for which the EZ template is created.
- `environment` – OS templates only. A list of environment variables set in the form of `<key>=<value>`.
- `mirrorlist` – Links to files with lists of repositories from which the packages in the EZ template are to be downloaded.
- `osrelease` – OS templates only. Contains native CentOS 7 distribution kernel version.
- `package_manager` – OS templates only. Specifies the packaging system used to handle the EZ template.
- `packages` – Contains a list of package names included in the corresponding EZ template.
- `pre-cache`, `post-cache` – OS templates only. Scripts that are executed before and after the packages in the EZ template are installed on the hardware node.
- `pre-install`, `post-install` – Scripts that are executed inside the container before and after the package management transaction.
- `pre-install-hn`, `post-install-hn` – Scripts that are executed on the hardware node before and after the package management transaction.
- `pre-upgrade`, `post-upgrade` – OS templates only. Scripts that are executed before and after updating packages inside the container.
- `pre-remove`, `post-remove` – Scripts that are executed before and after removing the application EZ template or package from the container.
- `release` – Contains template release number.
- `repositories` – Contains a list of repositories where the packages in the EZ template are stored.
- `summary` – A brief summary of the EZ template.
- `upgradable_versions` – OS templates only.
- `version` – Contains template version number.

10.2.3 Creating Customized EZ Template RPMs

To share a custom EZ template between hardware nodes, you can create an RPM package with it as follows:

1. Download the default OS template source from http://download.openvz.org/virtuozzo/releases/7.0/source/SRPMS.
2. Edit the template according to your needs, e.g., change OS template parameters, add, change or remove application templates, and such.
3. Build the RPM from the `.spec` file in a clean environment using standard tools. Do not build more than one template at once.

10.3 Setting Up Docker in Virtuozzo Containers

The current version of Virtuozzo supports Docker containers inside Virtuozzo system containers. You can install Docker in a Virtuozzo container from a Virtuozzo's Docker application template, from the official Docker repository as described in the Docker installation guide, or from your favorite OS repository. If you install Docker from the Docker or OS repository, make sure that your Docker version uses the overlayfs storage driver (for more information on Docker storage drivers, see https://docs.docker.com/engine/userguide/storagedriver/selectadriver/).

In the current version of Virtuozzo, you can run Docker in containers based on CentOS 7, Debian 8, Ubuntu 14.04 and 16.04, and Virtuozzo Linux 7. The Virtuozzo container will just need a network connection and enough diskspace and RAM to run the Docker containers you need.

To set up Docker in a Virtuozzo container, do the following:

1. Create a Virtuozzo container based on one of the aforementioned guest operating systems. For example:

   ```
   # prlctl create MyCT --vmtype ct --ostemplate centos-7-x86_64
   ```

2. Configure network in the Virtuozzo container:

   ```
   # prlctl set MyCT --netif_add eth0
   # prlctl set MyCT --ifname eth0 --dhcp yes --network Bridged
   ```

   For more details, see *Configuring Network Settings* on page 21.

3. Start the Virtuozzo container. For example:

```
# prlctl start MyCT
```

4. Install Docker in the Virtuozzo container:

```
# prlctl exec MyCT yum install docker -y
```

5. Start the Docker daemon:

```
# prlctl exec MyCT systemctl start docker.service
```

The Virtuozzo container is now ready to run Docker containers.

To check that Docker is installed properly, you can create and run test Docker containers in the Virtuozzo Container. For example, MySQL and Wordpress:

1. If required, increase container RAM and disk space. For example:

```
# prlctl set MyCT --memsize 4G --device-set hdd0 --size 50G
```

2. Launch MySQL:

```
# prlctl exec MyCT docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=123qwe -d mysql
```

3. Launch WordPress:

```
# prlctl exec MyCT docker run --name test-wordpress --link test-mysql:mysql -p 8080:80 \
-d wordpress
```

4. Visit the IP address of the Virtuozzo container on port 8080. You should see a standard WordPress installation screen.

10.3.1 Setting Up Docker for Running in Swarm Mode

Virtuozzo supports running Docker in swarm mode inside Virtuozzo containers. Swarm mode is enabled by either creating a swarm or joining an existing swarm. A swarm is a cluster of Docker nodes.

To be able to run Docker in swarm mode, you need to set up Docker in a Virtuozzo container as follows:

1. Perform steps 1-4 from the instruction given in *Setting Up Docker in Virtuozzo Containers* on page 176.

2. Create the `.dockerenv` file in the container:

```
# prlctl exec MyCT touch  /.dockerenv
```

3. Load the `ip_vs_ftp` module on the server:

```
# modprobe ip_vs_ftp
```

4. Start the Docker daemon:

```
# prlctl exec MyCT systemctl start docker.service
```

Once Docker is set up in the Virtuozzo container, you can proceed to creating a swarm. To do this, refer to Getting started with swarm mode.

10.3.2 Restrictions and Limitations for Docker in Virtuozzo Containers

1. Virtuozzo 7 does not support checkpointing and live migration of containers with Docker installed.
2. Virtuozzo 7 supports only overlayfs storage driver for Docker inside Virtuozzo containers.
3. Modules and third party add-ons that depend on operations prohibited in containers (loading of kernel modules, mounting of block devices, direct access to physical hardware) may not work in containers.

Please contact Virtuozzo technical support in case of any issues with Docker-related solutions.

10.4 Managing Container Virtual Hard Disk Encryption

Virtuozzo offers container virtual hard disk encryption capabilities based on `dm-crypt` and `cryptsetup`. The current implementation uses the AES-256 encryption algorithm. The encryption mechanism is separated from encryption key management, enabling you to use your own key management system (KMS) to issue and manage encryption keys.

Important: Only the root user must have access to encryption operations on the host.

The overall encryption procedure may be described as follows. An end user requests encryption for their container disk (create a new container with an encrypted disk, encrypt an existing disk, etc.). The KMS stores a secret encryption key and a public encryption key ID assigned to the end user. The administrator (or an automation tool) obtains the end user's encryption key ID from the KMS and passes it to the corresponding `prlctl` command. The `prlctl` command passes the key ID to the `getkey` executable that accesses the KMS and returns the encryption key which is passed to `cryptsetup`. The `cryptsetup` tool issues a master key unique for the container disk, encrypts the disk contents with the master key, then encrypts the master key stored in container disk's LUKS header with the encryption key passed from `getkey`.

The double encryption saves host resources in situations when the encryption key has to be changed (e.g., for key rotation purposes). In such cases, only the master key in the LUKS header has to be re-encrypted instead of the entire disk contents.

The only configuration step required to start using container disk encryption capabilities in Virtuozzo is to set up the encryption key requester as described further to be able to obtain encryption keys by their IDs for corresponding `prlctl` commands.

10.4.1 Setting Up Encryption Key Requester

The encryption mechanism communicates with the KMS as follows: executes the file `/usr/libexec/ploop/crypt.d/getkey` with the only string parameter–encryption key ID–and reads the returned encryption key value from the standard output. The `getkey` executable can be a script that calls the KMS binary and passes the specified encryption key ID to it.

On success, the executable is expected to exit with zero code and the key value is expected to be printed to the standard output in the binary form. On failure, the script is expected to exit with non-zero code.

Note: Key value may contain arbitrary bytes (e.g., `\x00`, `\n`, and such) that may be treated differently by various scripting languages. For example, assigning the key value to a Bash variable would strip the zero bytes from it, e.g., `key_va\x00lue` would become `key_value`.

To set up the encryption key requester on a Virtuozzo host, place the script to `/usr/libexec/ploop/crypt.d/getkey` and make it executable and only accessible by the root user:

```
# chown root:root /usr/libexec/ploop/crypt.d/getkey
# chmod 700 /usr/libexec/ploop/crypt.d/getkey
```

10.4.2 Encrypting and Decrypting Container Virtual Hard Disks

Following is a list of encryption-related operations you can perform on container virtual hard disks. All operations except decrypt require an encryption key ID obtained from your KMS.

Note: PFCache is disabled for encrypted disks.

1. Create a container with an encrypted root disk.

   ```
   # prlctl create <CT_name|CT_UUID> --vmtype ct --encryption-keyid <key_id>
   ```

2. Add a new encrypted disk to a container.

```
# prlctl set <CT_name|CT_UUID> --device-add hdd --encryption-keyid <key_id>
```

3. Encrypt an unencrypted disk. During this operation, a new empty encrypted disk is created, the data is moved to it from the unencrypted disk which is then securely erased with `shred` (unless `--no-wipe` is added). For this operation, the host needs free disk space equal to the size of the container disk being encrypted.

```
# prlctl set <CT_name|CT_UUID> --device-set hdd0 --encrypt --encryption-keyid <key_id> \
[--no-wipe]
```

4. Change the encryption key ID of an encrypted disk with or without re-encrypting the entire disk contents. By default, only the LUKS header of an encrypted disk is re-encrypted to save host resources. The entire disk contents can be re-encrypted by adding `--reencrypt` to the command. During this operation, a new empty encrypted disk is created, the data is moved to it from the old encrypted disk which is then securely erased with `shred` (unless `--no-wipe` is added). For this operation, the host needs free disk space equal to the size of the container disk being re-encrypted.

```
# prlctl set <CT_name|CT_UUID> --device-set hdd0 --encryption-keyid <key_id> [--reencrypt] \
[--no-wipe]
```

5. Decrypt an encrypted container disk. During this operation, a new empty unencrypted disk is created and the data is moved to it from the encrypted disk. If the operation completes successfully, the encrypted disk is deleted. If the operation fails, the partially decrypted data is securely erased with `shred` (unless `--no-wipe` is added) and the encrypted disk remains intact. For this operation, the host needs free disk space equal to the size of the container disk being decrypted.

```
# prlctl set <CT_name|CT_UUID> --device-set hdd0 --decrypt [--no-wipe]
```

10.4.3 Encrypting System Swap

Even if containers are encrypted, their memory may still be swapped to an unencrypted system swap partition. On nodes intended to run encrypted containers, consider encrypting the swap partition as well.

To encrypt a swap partition with a random encryption key, do the following:

1. Identify swap partitions in the system:

```
# swapon -s
Filename         Type            Size    Used    Priority
/dev/<partition> partition       XXXXXXX 0       -1
```

2. Add a swap partition's entry to the `/etc/crypttab` file. For example:

```
swap  /dev/<partition>  /dev/urandom   swap,noearly
```

Once the encryption is initiated, this will map the `/dev/<partition>` to `/dev/mapper/swap` as an encrypted swap partition.

Note: To create an encryption key, you may use `/dev/random` instead of `/dev/urandom`. However, in this case, the system may take a long time to boot.

3. Find out the UUID of the swap partition:

```
# blkid /dev/<partition>
/dev/<partition>: UUID="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" TYPE="swap"
```

4. Comment the swap partition's entry in the `/etc/fstab` file and add an entry for the mapped `/dev/mapper/swap` file. For example:

```
#UUID=<partition_UUID> swap swap defaults 0 0
/dev/mapper/swap none swap sw 0 0
```

5. Reboot to initiate the encryption.

6. Run the `lsblk` command to ensure the swap is encrypted. In the output, the swap partition's `TYPE` should be `crypt`.

```
# lsblk
NAME                 MAJ:MIN    RM  SIZE RO TYPE  MOUNTPOINT
...
   -<swap_partition>   X:X      0   XG   0  crypt [SWAP]
...
```

The output of `swapon -s` will show a different filename:

```
# swapon -s
Filename                                Type            Size    Used    Priority
/dev/dm-X                               partition       XXXXXXX 0       -1
```

10.5 Connecting to Virtual Machines and Containers via VNC

You can use your favorite VNC clients to connect to and manage containers and virtual machines. To do this, you need to complete these steps:

1. (Recommended) Secure VNC connections on the node with SSL.

2. Enable VNC access in the desired virtual machine or container.

3. Connect to the virtual machine or container with a VNC client.

The sections below describe these steps in details.

10.5.1 Securing VNC Connections with SSL

To set up SSL for all VNC connections on the node, do the following:

1. Acquire an SSL certificate and key from a trusted certificate authority.

2. Configure the VNC server to use the certificate and key:

   ```
   # prlsrvctl set --vnc-ssl-certificate <path_to_crt_file> --vnc-ssl-key <path_to_key_file>
   ```

To disable VNC encryption, specify empty arguments. For example:

```
# prlsrvctl set --vnc-ssl-certificate '' --vnc-ssl-key ''
```

10.5.2 Enabling VNC Access to Virtual Machines

To enable VNC access to a virtual machine, you need to do the following:

1. Enable VNC support in the virtual machine.

2. Specify the TCP port number on the physical server that will be used to listen to VNC connections for the virtual machine.

   Note: A unique port number must be specified for each virtual machine where you plan to connect via VNC.

3. Set a password to secure your VNC connection.

You can perform all these operations with a single command. For example:

```
# prlctl set MyVM --vnc-mode manual --vnc-port 5901 --vnc-passwd XXXXXXXX
```

The changes will come into effect on the next virtual machine start.

10.5.3 Enabling VNC Access to Containers

To enable VNC access to a container, you need to do the following:

1. Make sure you have a valid user account in the container to be able to log into it.
2. Make sure the container is running.
3. Set the VNC mode and password for the container. For example:

```
# prlctl set MyCT --vnc-mode manual --vnc-port 6501 --vnc-passwd XXXXXXXX
```

Note: Port number must be unique for each container you open VNC access to. In the auto mode, correct port numbers are assigned automatically. In the manual mode, you need to make sure port numbers are unique yourself.

10.5.4 Connecting with a VNC Client

After you have enabled VNC access to the virtual machine or container, you can connect to it with your favorite VNC client. To do this, you need to pass the following parameters to the VNC client:

- IP address of the server where the virtual machine or container is hosted.
- Port number and password you specified when enabling VNC access.
- Valid user account in the virtual machine or container.

10.6 Managing iptables Modules

This section describes how to manage `iptables` modules for both physical servers and containers.

10.6.1 Using iptables Modules in Virtuozzo

Filtering network packets on hardware nodes running Virtuozzo does not differ from doing so on a typical Linux server. You can use the standard `iptables` tool to control how network packets enter, move through, and exit the network stack within the Virtuozzo kernel.

Connection tracking on the hardware node is disabled by default. Setting `iptables` rules that require `conntrack` functionality enables tracking of new connections and makes the node vulnerable to DoS attacks, since the number of `conntrack` slots is limited. However, setting such rules for particular virtual machines and containers (e.g., for NAT) leaves other containers, virtual machines and the hardware node reachable in case of a DoS attack.

Note: Once `conntrack` is enabled for a container, it cannot be disabled until the restart of the hardware node or said container.

To detect active connections tracked on the hardware node, check if the `/proc/net/nf_conntrack` file contains any entries:

```
# cat /proc/net/nf_conntrack
```

For your reference, below are several resources you can consult to get detailed information on using `iptables` on Linux servers:

- Red Hat Enterprise Linux 7 Security Guide contains a section focusing on packet filtering basics and explaining various options available for `iptables`.
- iptables Tutorial 1.2.2 explains in great detail how `iptables` is structured and works.

10.6.2 Using iptables Modules in Containers

Using `iptables` modules in containers requires additional configuration on your part.

10.6.2.1 Configuring iptables Modules

To set the state of `iptables` modules for backup/restore or live migration, use the `prlctl set --netfilter` command. If some of the `iptables` modules allowed for a container are not loaded on the hardware node where that container has been restored or migrated, they will be automatically loaded when that container starts. For example, the command

```
# prlctl set MyCT --netfilter stateful
```

will make sure that all modules except NAT-related will be allowed and loaded for the container `MyCT` (if required) on a hardware node where it has been restored or migrated.

Note: The default setting is `full`, which allows all modules.

10.6.2.2 Using conntrack Rules and NAT Tables

To limit the maximum number of `conntrack` slots available for each container on the hardware node, set the `net.netfilter.nf_conntrack_max` variable. For example:

```
# sysctl -w net.netfilter.nf_conntrack_max=50000
```

The value of `net.netfilter.nf_conntrack_max` cannot exceed the value of `net.nf_conntrack_max`.

Note: Even if a container is under a DoS attack and all its `conntrack` slots are in use, other containers will not be affected, still being able to create as many connections as set in `net.netfilter.nf_conntrack_max`.

10.7 Using SCTP in Containers and Virtual Machines

Virtuozzo supports the Stream Control Transmission Protocol (SCTP) in both containers and virtual machines. In virtual machines, SCTP can be set up and used like on any physical machine. In containers, SCTP support is disabled by default. To enable SCTP in containers, load the `sctp` kernel module on the hardware node with

```
# modprobe sctp
```

After the module is loaded, you can create and use SCTP sockets inside containers with standard Linux tools.

If required, add an `sctp` entry to the `/etc/modules-load.d/vz.conf` file to automatically load the `sctp` kernel module on hardware node start.

Note: Live migration via SCTP is not supported.

10.8 Creating Configuration Files for New Linux Distributions

Distribution configuration files are used to distinguish among containers running different Linux versions and to determine what scripts should be executed when performing the relevant container-related operations (e.g., assigning a new IP address to the container).

All Linux distributions shipped with Virtuozzo have their own configuration files located in the `/usr/libexec/libvzctl/dists/` directory on the hardware node. However, you may wish to create your own distribution configuration files to support new Linux versions released. Let us assume that you wish your containers to run the CentOS 7 Linux distribution and, therefore, have to make the `centos-7.conf` distribution configuration file to define what scripts are to be executed while performing major tasks with containers running this Linux version. To do this:

1. In the container configuration file (with the name of `/etc/vz/conf/<UUID>.conf`), specify `centos-7` as the value of the `DISTRIBUTION` variable (for example, `DISTRIBUTION="centos-7"`).

2. Create the `centos-7.conf` configuration file in the `/usr/libexec/libvzctl/dists/` directory. The easiest way to do it is copy one of the existing configuration files by executing the following command in the `/usr/libexec/libvzctl/dists/` directory:

   ```
   # cp fedora.conf centos-7.config
   ```

   In the example above, we assume that the `fedora.conf` file is present in the `/usr/libexec/libvzctl/dists/` directory on the hardware node. In case it is not, you may use any other distribution configuration file available on your server.

3. Open the `centos.conf` file for editing, go to the first entry and, in the right part of the entry, specify the name of the script you wish to be run on issuing the `prlctl` command with the parameter specified in the left part of the entry. For example, if you wish the script to be executed while assigning a new IP address to your container and the script has the `my_centos_script` name, your entry should look as follows:

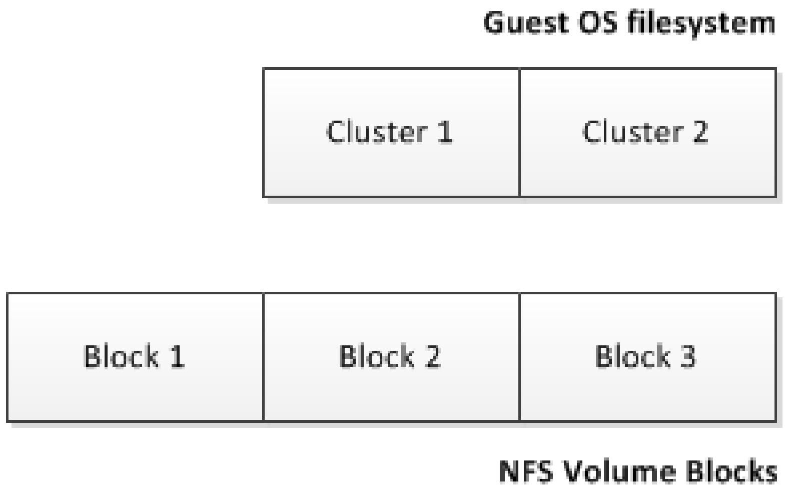
   ```
   ADD_IP=my_centos_script-add_ip.sh
   ```

4. Repeat Step 3 for all entries in the file.

5. Place the scripts for the new Linux distribution to the `/usr/libexec/libvzctl/dists/scripts` directory on the Node. Make sure the names of these scripts coincide with those specified in the `centos-7.conf` file.

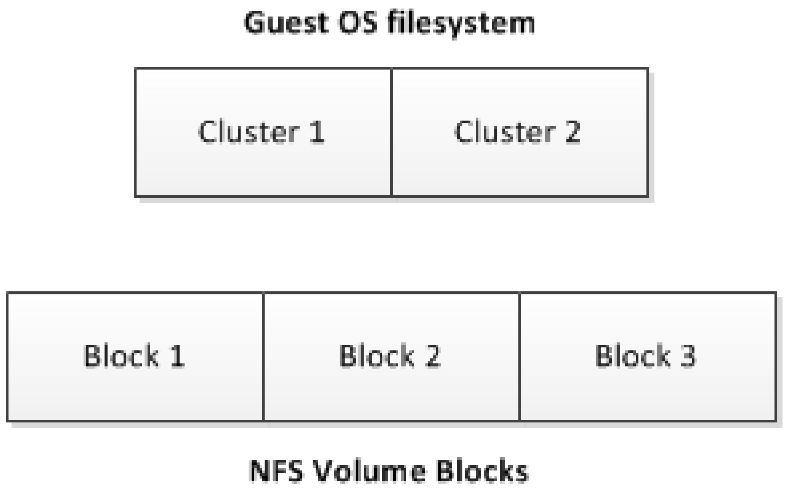
10.9 Aligning Disks and Partitions in Virtual Machines

Most of the modern operating systems automatically align partitions when they are installed in virtual machines. For example, Windows Server 2008 creates a default partition offset of 1024 KB to satisfy the partition alignment requirements. The following figure shows an example of correct partition alignment:



In this example, any cluster (the smallest unit of data) in the guest OS file system is aligned with the boundaries of an NFS block, and reading from or writing to a cluster requires only access to one NFS block. For example, reading from Cluster 1 causes only a read from Block 1.

At the same time, virtual machines running non-modern systems (for example, Windows Server 2008 or Red Hat Enterprise Linux 5) do usually have misaligned partitions, which is shown in the figure below:



In this example, clusters of the guest OS file system do not match the boundaries of NFS blocks, and reading from or writing to a cluster requires access to several NFS blocks. For example, reading from Cluster 1 causes two reads: from Block 1 and from Block 2. This results in a slower read time as compared to properly aligned partitions and leads to performance degradation.

10.9.1 Aligning Partitions

Basically, to align disks and partitions in virtual machines, you need to set an offset so that clusters in the guest OS file system match the volume block size on your NFS storage. Usually, the block size of most network storages is 512 bytes or a multiple of 512 bytes. As an example, the following sections describe the procedure of aligning disks and partitions for Linux and Windows virtual machines assuming that the size of your NFS blocks is 512 bytes.

When deciding on aligning disks and partitions, take into account that this process destroys all data on these disks and partitions. So if you want to have a correctly aligned system partition, you need to align your disks and partitions before creating a virtual machine and installing a guest operating system in it. If you do not want an aligned system partition, you can first create a virtual machine and install a guest OS in it, and then align your data disks from inside the virtual machine.

The sections below demonstrate how to align disks and partitions before you start installing a guest OS. You can, however, use a similar procedure to align data disks and partitions from inside your virtual machines.

10.9.2 Checking Partition Alignment in Existing Virtual Machines

First of all, you may wish to know how you can check that the partitions of a virtual machine are not aligned. Depending on the operating system installed in the virtual machine, you can do the following.

10.9.2.1 Linux Virtual Machines

To check the partition alignment in a Linux virtual machine, log in to this virtual machine and run the following command:

```
# fdisk -l -u /dev/device_name
```

For example, to check the partition alignment on the sdc device, you can run this command:

```
# fdisk -l -u /dev/sdc
Disk /dev/sdc: 73.0 GB, 73014444032 bytes
```

```
255 heads, 63 sectors/track, 8876 cylinders, total 142606336 sectors
Units = sectors of 1 * 512 = 512 bytes
   Device       Boot    Start         End       Blocks  Id      System
/dev/sdc1       *          63      208844       104391  83      Linux
/dev/sdc2              208845   142592939    71192047+  8e      Linux LVM
```

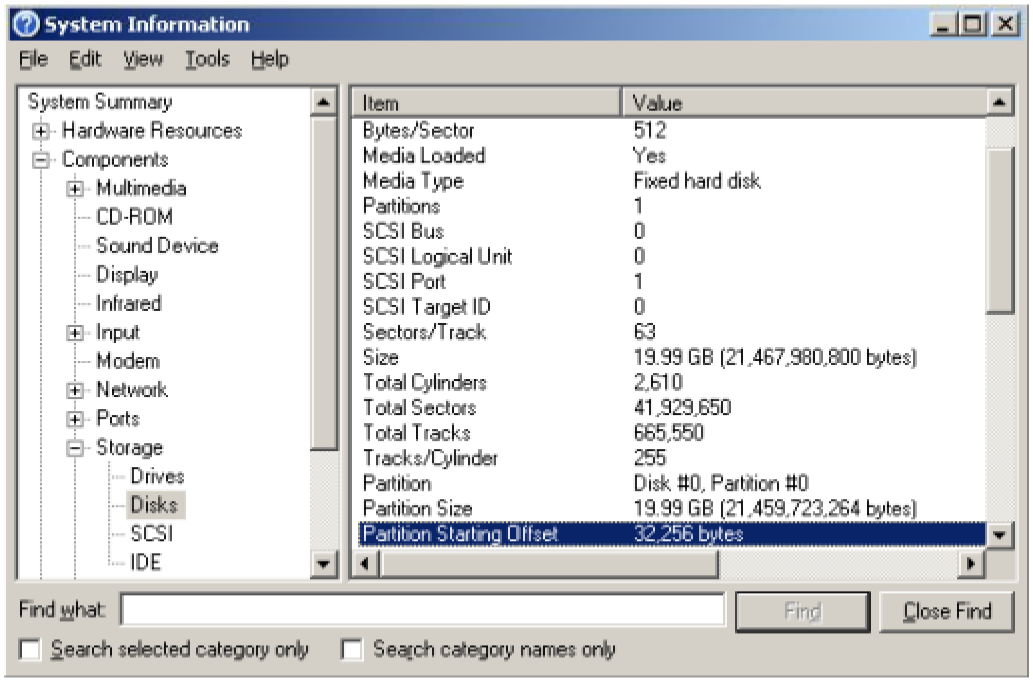
Pay attention to the number of sectors in the Start column. Usually, a sector contains 512 bytes, which makes up 32256 bytes for 63 sectors for the `/dev/sdc1` partition and 26105625 bytes for 208845 for the `/dev/sdc2` partition. For a partition to be properly aligned, it must align with 4096 byte boundaries (assuming that the block size of your storage is 4 KB). As 32256 and 106928640 is not a multiple of 4096, the partitions `/dev/sdc1` and `/dev/sdc2` are not aligned properly. To align them, you should offset

- the `/dev/sdc1` partition by 1 sector so that it starts at 64. In this case, 64 sectors each containing 512 bytes make up 32768 that is a multiple of 4096.
- the `/dev/sdc2` partition by 3 sectors so that it starts at 208848. In this case, 208848 sectors each containing 512 bytes make up 106930176 that is a multiple of 4096.

10.9.2.2 Windows Virtual Machines

To check the partition alignment in a Windows virtual machine, do the following:

1. Click Start > Run, type `msinfo32.exe`, and press Enter to open System Information.
2. Navigate to Components > Storage > Disks, and look for the Partition Starting Offset field in the right part of the window.



Item	Value
Bytes/Sector	512
Media Loaded	Yes
Media Type	Fixed hard disk
Partitions	1
SCSI Bus	0
SCSI Logical Unit	0
SCSI Port	1
SCSI Target ID	0
Sectors/Track	63
Size	19.99 GB (21,467,980,800 bytes)
Total Cylinders	2,610
Total Sectors	41,929,650
Total Tracks	665,550
Tracks/Cylinder	255
Partition	Disk #0, Partition #0
Partition Size	19.99 GB (21,459,723,264 bytes)
Partition Starting Offset	32,256 bytes

To find out if the partition is aligned properly, use the method described above for Linux virtual machines.

10.9.3 Aligning Disks for Linux Virtual Machines

To align partitions for use in a Linux virtual machine, you need a working Linux virtual machine. Once you have it at hand, follow the steps below:

1. Create a new disk for the virtual machine. On this disk, you will create aligned partitions. Then you will connect the disk to a new virtual machine and install your Linux guest OS on this disk.

2. Start the virtual machine and log in to it using SSH.

3. Run the `fdisk` utility for the disk you want to align.

4. Create a primary partition, and set the starting block number for the created partition.

5. Repeat steps 3-4 to create and align all partitions you plan to have in your new virtual machine.

The following example creates partition #1 with the size of 1 GB on the `/dev/sda` device and uses the offset of 64 KB.

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
```

```
content won't be recoverable.
The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
        (e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
Command (m for help): n
Command action
        e       extended
        p       primary partition (1-4)
p
Partition number (1-4): 1
First sector (63-16777215, default 63): 64
Last sector or +size or +sizeM or +sizeK (64-16777215, default 16777215): 208848
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

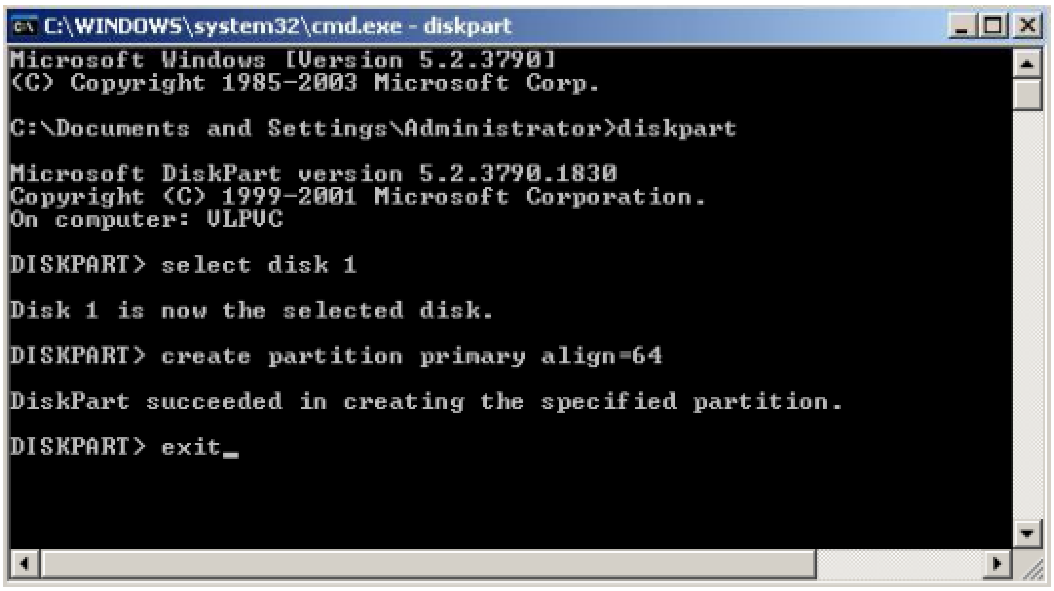
Once you align all the necessary partitions, disconnect the disk from the virtual machine. When creating a new virtual machine, choose this disk for use with this virtual machine.

10.9.4 Aligning Partitions for Windows Virtual Machines

To align a disk for a Windows virtual machine, you need a working Windows virtual machine. Once you have it at hand, you can use the `diskpart` or `diskpar` utility (depending on your operating system) to align the disk:

1. Create a new disk for the virtual machine. On this disk, you will create aligned partitions. Then you will connect the disk to a new virtual machine and install your Windows guest OS on this disk.
2. Open the command-line prompt, and run the `diskpart` or `diskpar` utility.
3. Select the disk you want to align.
4. Create the primary partition on the disk, and align it.
5. Exit the `diskpart` or `diskpar` utility, and close the command-line prompt.

The following example demonstrates how to use the `diskpart` utility to align disk 1 by setting the offset of 64 for it:



```
C:\WINDOWS\system32\cmd.exe - diskpart
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>diskpart

Microsoft DiskPart version 5.2.3790.1830
Copyright (C) 1999-2001 Microsoft Corporation.
On computer: VLPVC

DISKPART> select disk 1

Disk 1 is now the selected disk.

DISKPART> create partition primary align=64

DiskPart succeeded in creating the specified partition.

DISKPART> exit_
```

Once you align the virtual disk, disconnect it from the virtual machine. When creating a new virtual machine, choose this disk for use with this virtual machine.

10.9.5 Creating a Template of a Virtual Machine with Aligned Partitions

To facilitate the procedure of creating virtual machines that have aligned system partitions, you can create a template of the aligned virtual machine and deploy new virtual machines from this template.

For example, if you align a disk by following the steps in *Aligning Partitions for Windows Virtual Machines* on page 191, then create a new virtual machine that uses this disk, and then install Windows Server 2008 operating system in the virtual machine, you will have a clean Windows Server 2008 installation on the correctly aligned disk. Now you can create a template of this virtual machine and use this template each time you need to deploy a new virtual machine with Windows Server 2008.

10.10 Uninstalling Virtuozzo Guest Tools from Virtual Machines

Note: Running virtual machines that do not have Virtuozzo guest tools installed cannot be configured from the physical server.

10.10. Uninstalling Virtuozzo Guest Tools from Virtual Machines

If you find out that Virtuozzo guest tools are incompatible with some software inside a virtual machine, you can uninstall them from that VM. The steps you need to perform to remove guest tools differ depending on the guest operating system and are described in the sections below.

10.10.1 Uninstalling Guest Tools from Linux Virtual Machines

To uninstall Virtuozzo guest tools from a Linux guest, log in to the virtual machine and do as follows:

1. Remove the packages:

   1.1. On RPM-based systems (CentOS and other):

   ```
   # yum remove dkms-vzvirtio_balloon prl_nettool qemu-guest-agent-vz vz-guest-udev
   ```

   1.2. 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 Virtuozzo guest tools, restart the virtual machine.

Note: After Virtuozzo guest tools are removed from a virtual machine, their state is shown as `possibly installed`.

10.10.2 Uninstalling Guest Tools from Windows Virtual Machines

To uninstall Virtuozzo guest tools for Windows, log in to the virtual machine and do as follows:

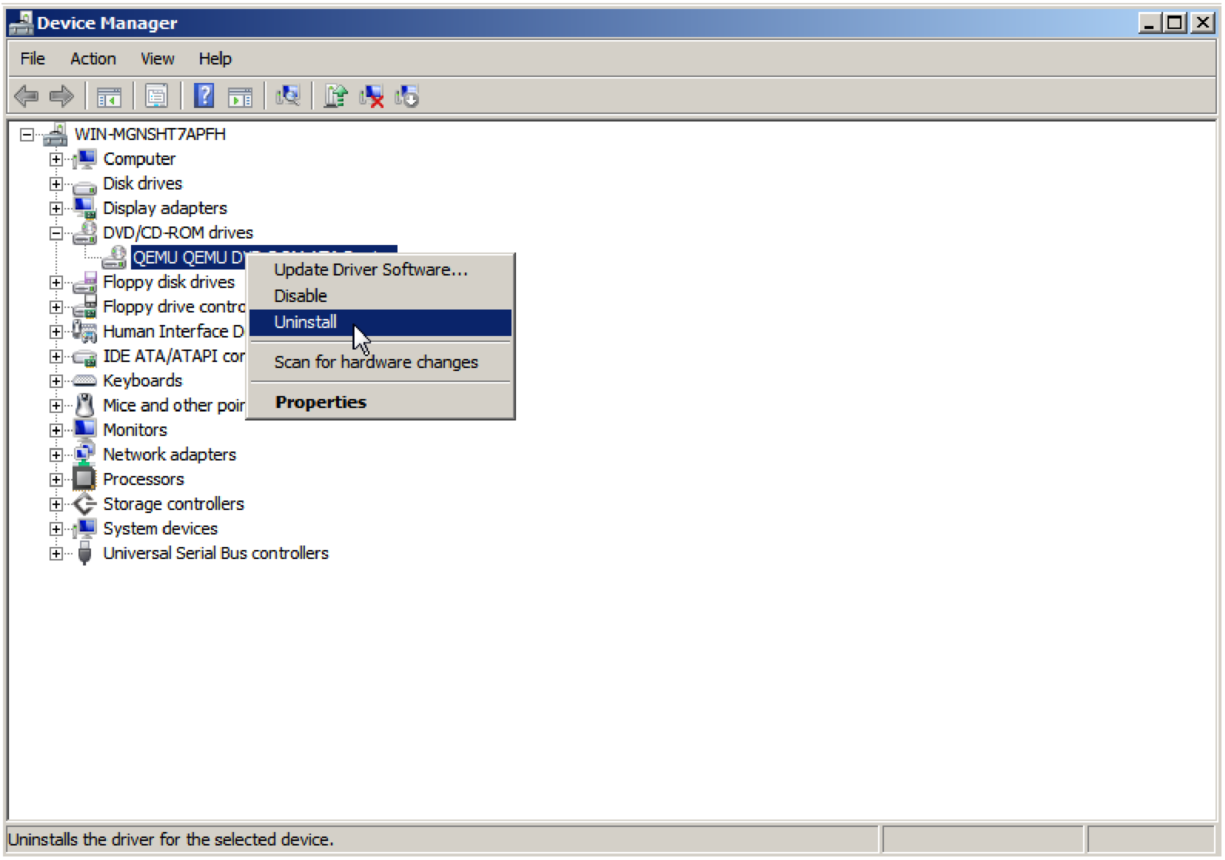
1. Remove virtualized device drivers:

   Important: Do not remove the VirtIO/SCSI hard disk driver and NetKVM network driver. Without the former, the virtual machine will not boot; without the latter, it will lose network connectivity.

1.1. From the Windows **Start** menu, open **Control Panel** > **System and Security** > **System** > **Device Manager**.

1.2. Double-click the device to expand the list of installed drivers.

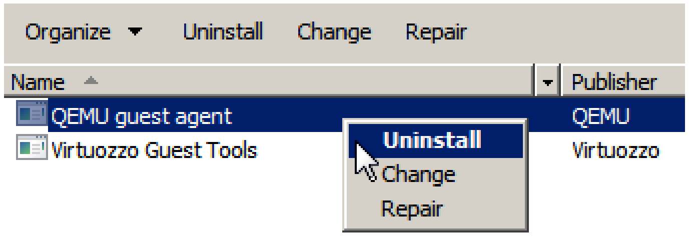
1.3. Right-click the driver to be removed and select **Uninstall** from the drop-down menu.



2. Uninstall QEMU guest agent and Virtuozzo Guest Tools:

2.1. From the Windows **Start** menu, open **Control Panel** > **Programs** > **Programs and Features**.

2.2. Right-click **QEMU guest agent** and select **Uninstall** from the drop-down menu.



2.3. Right-click **Virtuozzo Guest Tools** and select **Uninstall** from the drop-down menu.

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 Virtuozzo guest tools, restart the virtual machine.

Note: After Virtuozzo guest tools are removed from a virtual machine, their state is shown as `possibly installed`.

10.11 Enabling Legacy VM Debug Mode

The legacy VM debug mode makes sure that legacy VMs that failed to convert to the Virtuozzo 7 format during migration or restoration from backup are not deleted from the destination Virtuozzo 7 server after failed conversion. With the debug mode enabled, such virtual machines remain stopped or running with disabled network to let the technical support team study the memory dump and find out the reason for failure.

If you are trying to migrate or restore backup of a legacy virtual machine to a Virtuozzo 7 server and conversion stage fails, you can enable the debug mode on the destination Virtuozzo 7 server, make another migration or restore attempt, send the problem report, and contact the technical support team.

To enable the legacy VM debug mode on the destination Virtuozzo 7 server:

1. Stop the dispatcher:

   ```
   # systemctl stop prl-disp.service
   ```

Note: While the dispatcher is stopped, you cannot manage and collect stats of VMs and containers. Running VMs and containers are not stopped.

2. In `/etc/vz/dispatcher.xml`, change `<LegacyVmUpgrade>0</LegacyVmUpgrade>` to `<LegacyVmUpgrade>1</LegacyVmUpgrade>`.

3. Start the dispatcher:

```
# systemctl start prl-disp.service
```

10.12 Installing Optional Virtuozzo Packages

Virtuozzo comes with everything you may need already installed. However, you can also install optional Virtuozzo packages from remote repositories by means of the `yum` command.

Note: For more information on using `yum` in Virtuozzo, see *Updating Virtuozzo* on page 140 and the `yum` manual page.

10.13 Enabling Nested Virtualization in Virtual Machines

Warning: This feature is experimental and tested only on Linux guests. The operation of nested virtual machines may be unstable.

Virtuozzo supports nested Intel VT-x virtualization in virtual machines.

To enable nested virtualization, do the following:

1. Stop all running or paused virtual machines on the node.

2. Unload the `kvm-intel` module from the kernel:

```
# rmmod kvm_intel
```

3. Add the line `options kvm-intel nested=y` to the `dist.conf` file:

```
# echo 'options kvm-intel nested=y' >> /etc/modprobe.d/dist.conf
```

4. Load the `kvm-intel` module:

   ```
   # modprobe kvm_intel
   ```

5. Enable nested virtualization in the desired virtual machine:

   ```
   # prlctl set MyVM --nested-virt on
   ```

Note:

1. A guest operating system in a nested virtual machine will not be able to obtain an IP address via DHCP if `ipfilter`, `macfilter`, and `preventpromisc` parameters of the host VM's bridged network adapter are set to `no`.
2. You cannot change CPU features mask for nested virtual machines using the `prlsrvctl set --cpu-features-mask` command.

10.14 Participating in Customer Experience Program

By participating in the Customer Experience Program (CEP) you agree to send to Virtuozzo information about configuration of your physical server and virtual environments, the way you use Virtuozzo products, and technical issues that you encounter.

Note: No private information like your name, e-mail address, phone number, or keyboard input will be collected.

The program is voluntary and helps improve Virtuozzo products to better fit your needs.

When installing Virtuozzo in the attended mode, you automatically join CEP. You can, however, opt out of the program at any time by stopping and disabling the `disp-helper` service:

```
# systemctl stop disp-helper
# systemctl disable disp-helper
```

By default, Virtuozzo will collect information once a week, although you can change this interval. For example, to have the data collected every two weeks, do as follows:

1. In the configuration file `/etc/vz/disp_helper.json`, change the `report_period` value to `14d`:

   ```
   "report_period": "14d",
   ```

2. Restart `disp-helper` to apply changes:

   ```
   # systemctl restart disp-helper
   ```

To fine-tune which information about your physical server or virtual environments is collected, you can disable or enable corresponding `.py` scripts in the `/usr/share/virtuozzo/cep-scripts` directory. For example, to prevent collection of data about your virtual environments, you can disable the `libvirt_guests.py` script as follows:

```
# chmod a-x libvirt_guests.py
```

To re-enable the script, run

```
# chmod a+x libvirt_guests.py
```

When installing Virtuozzo in the unattended mode, you can specify the `cep` parameter in the kickstart file. For more details, see the Virtuozzo 7 PXE Installation Guide.

CHAPTER 11

Troubleshooting

This chapter provides the information about problems that may occur during your work with Virtuozzo and suggests the ways to solve them.

11.1 General Considerations

The general issues to take into consideration when troubleshooting your system are listed below. You should read them carefully before trying to solve more specific problems.

- You should always remember where you are currently located in your terminal. Check it periodically using the `pwd`, `hostname`, `ifconfig`, `cat /proc/vz/veinfo` commands. One and the same command executed inside a virtual machine or container and on the hardware node can lead to very different results. You can also set up the `PS1` environment variable to show the full path in the `bash` prompt. To do this, add these lines to `/root/.bash_profile`:

```
PS1="[\u@\h \w]$ "
export PS1
```

- If the hardware node slows down, use `vmstat`, `ps` (`ps axfw`), `dmesg`, `htop` (`vztop`) to find out what is happening, never reboot the machine without investigation. If no thinking helps restore the normal operation, use the Alt+SysRq sequences to dump the memory (`showMem`) and processes (`showPc`).

- Do not run any binary or script that belongs to a container directly from the hardware node, for example, do not ever do this:

```
cd /vz/root/99/etc/init.d
./httpd status
```

Any script inside a container could have been changed to whatever the container owner chooses: it could have been trojaned, replaced to something like `rm -rf`, etc. You can use only `prlctl exec/prlctl enter` to execute programs inside a container.

- Do not use init scripts on the hardware node. An init script may use `killall` to stop a service, which means that all similar processes will be killed in all containers. You can check `/var/run/Service.pid` and kill the correspondent process explicitly.
- You must be able to detect any rootkit inside a container. It is recommended to use the `chkrootkit` package for detection (you can download the latest version from http://www.chkrootkit.org), or at least run

  ```
  rpm -Va|grep "S.5"
  ```

  to check up if the MD5 sum has changed for any RPM file.

  You can also run `nmap`, for example:

  ```
  # nmap -p 1-65535 192.168.0.1
  Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
  Interesting ports on  (192.168.0.1):
  (The 65531 ports scanned but not shown below are in
    state: closed)
  Port       State       Service
  21/tcp     open        ftp
  22/tcp     open        ssh
  80/tcp     open        http
  111/tcp    open        sunrpc
  Nmap run completed -- 1 IP address (1 host up) scanned
    in 169 seconds
  ```

  to check if any ports are open that should normally be closed.

  That could however be a problem to remove a rootkit from a container and make sure it is 100% removed. If you're not sure, create a new container for that customer and migrate his/her sites and mail there.
- Check the `/var/log/` directory on the hardware node to find out what is happening on the system. There are a number of log files that are maintained by the system and Virtuozzo (the `boot.log`, `messages`, etc.), but other services and programs may also put their own log files here depending on your distribution of Linux and the services and applications that you are running. For example, there may be logs associated with running a mail server (the `maillog` file), automatic tasks (the `cron` file), and others. However, the first place to look into when you are troubleshooting is the `/var/log/messages` log file. It contains the boot messages when the system came up as well as other status messages as the system runs. Errors with I/O, networking, and other general system errors are reported in this file. So, we recommend that you read to the `messages` log file first and then proceed with the other files from the

`/var/log/` directory.

- Subscribe to bug tracking lists. You should keep track of new public DoS tools or remote exploits for the software and install them into containers or at hardware nodes.
- When using `iptables`, there is a simple rule for Chains usage to help protect both the hardware node and its containers:
  - use INPUT, OUTPUT to filter packets that come in/out the hardware node
  - use FORWARD to filter packets that are designated for containers

11.2 Kernel Troubleshooting

11.2.1 Using ALT+SYSRQ Keyboard Sequences

Press ALT+SYSRQ+H and check what is printed at the hardware node console, for example:

```
SysRq: unRaw Boot Sync Unmount showPc showTasks showMem loglevel0-8 tErm kIll \
       killalL Calls Oops
```

This output shows you what ALT+SYSRQ sequences you may use for performing this or that command. The capital letters in the command names identify the sequence. Thus, if there are any troubles with the machine and you're about to reboot it, please use the following key sequences before pressing the Power button:

- ALT+SYSRQ+M to dump memory info
- ALT+SYSRQ+P to dump processes states
- ALT+SYSRQ+S to sync disks
- ALT+SYSRQ+U to unmount filesystems
- ALT+SYSRQ+L to kill all processes
- ALT+SYSRQ+U try to unmount once again
- ALT+SYSRQ+B to reboot

If the server is not rebooted after that, you can press the Power button.

11.2.2 Saving Kernel Faults (OOPS)

You can use the following command to check for the kernel messages that should be reported to Virtuozzo developers:

```
grep -E "Call Trace|Code" /var/log/messages*
```

Then, you should find kernel-related lines in the corresponding log file and figure out what kernel was booted when the oops occurred. Search backward for the `Linux` string, look for strings like:

```
Sep 26 11:41:12 kernel: Linux version 2.6.18-8.1.1.el5.028stab043.1 \
(root@rhel5-32-build) (gcc version 4.1.1 20061011 (Red Hat 4.1.1-30)) \
#1 SMP Wed Aug 29 11:51:58 MSK 2007
```

An oops usually starts with some description of what happened and ends with the Code string. Here is an example:

```
Aug 25 08:27:46 boar BUG: unable to handle kernel NULL pointer dereference at \
virtual address 00000038
Aug 25 08:27:46 boar printing eip:
Aug 25 08:27:46 boar f0ce6507
Aug 25 08:27:46 boar *pde = 00003001
Aug 25 08:27:46 boar Oops: 0000 [#1]
Aug 25 08:27:46 boar SMP
Aug 25 08:27:46 boar last sysfs file:
Aug 25 08:27:46 boar Modules linked in: snapapi26(U) bridge(U) ip_vzredir(U) \
vzredir(U) vzcompat(U) vzrst(U) i
p_nat(U) vzcpt(U) ip_conntrack(U) nfnetlink(U) vzlinkdev(U) vzethdev(U) vzevent(U) \
vzlist(U) vznet(U) vzmo
n(U) xt_tcpudp(U) ip_vznetstat(U) vznetstat(U) iptable_mangle(U) iptable_filter(U) \
ip_tables(U) vztable(U) vzdquota(U) vzdev(U) autofs4(U) hidp(U) rfcomm(U) l2cap(U) \
bluetooth(U) sunrpc(U) ipv6(U) xt_length(U) ipt_ttl(U) xt_tcpmss(U) ipt_TCPMSS(U) \
xt_multiport(U) xt_limit(U) ipt_tos(U) ipt_REJECT(U) x_tables(U) video(U) sbs(U) \
i2c_ec(U) button(U) battery(U) asus_acpi(U) ac(U) lp(U) floppy(U) sg(U) pcspkr(U) \
i2c_piix4(U) e100(U) parport_pc(U) i2c_core(U) parport(U) cpqphp(U) eepro100(U) \
mii(U) serio_raw(U) ide_cd(U) cdrom(U) ahci(U) libata(U) dm_snapshot
(U) dm_zero(U) dm_mirror(U) dm_mod(U) megaraid(U) sym53c8xx(U) \
scsi_transport_spi(U) sd_mod(U) scsi_mod(U) ext3(U) jbd(U) ehci_hcd(U) ohci_hcd(U) \
uhci_hcd(U)
Aug 25 08:27:46 boar CPU: 1, VCPU: -1.1
Aug 25 08:27:46 boar EIP: 0060:[<f0ce6507>] Tainted: P VLI
Aug 25 08:27:46 boar EFLAGS: 00010246 (2.6.18-028stab043.1-ent #1)
Aug 25 08:27:46 boar EIP is at clone_endio+0x29/0xc6 [dm_mod]
Aug 25 08:27:46 boar eax: 00000010   ebx: 00000001   ecx: 00000000   edx: 00000000
Aug 25 08:27:46 boar esi: 00000000   edi: b6f52920   ebp: c1a8dbc0   esp: 0b483e38
Aug 25 08:27:46 boar ds: 007b   es: 007b   ss: 0068
Aug 25 08:27:46 boar Process swapper (pid: 0, veid: 0, ti=0b482000 task=05e3f2b0 \
task.ti=0b482000)
Aug 25 08:27:46 boar Stack: 0b52caa0 00000001 00000000 b6f52920 00000000f0ce64de \
00000000 02478825
```

```
Aug 25 08:27:46 boar 00000000 c18a8620 b6f52920 271e1a8c 024ca03800000000 00000000 \
00000000
Aug 25 08:27:46 boar 00000000 00000000 c18a3c00 00000202 c189e89400000006 00000000 \
05cb7200
Aug 25 08:27:46 boar Call Trace:
Aug 25 08:27:46 boar [<f0ce64de>] clone_endio+0x0/0xc6 [dm_mod]
Aug 25 08:27:46 boar [] bio_endio+0x50/0x55
Aug 25 08:27:46 boar [<024ca038>] __end_that_request_first+0x185/0x47c
Aug 25 08:27:46 boar [<f0c711eb>] scsi_end_request+0x1a/0xa9 [scsi_mod]
Aug 25 08:27:46 boar [<02458f04>] mempool_free+0x5f/0x63
Aug 25 08:27:46 boar
Aug 25 08:27:46 boar [<f0c713c3>] scsi_io_completion+0x149/0x2f3 [scsi_mod]
Aug 25 08:27:46 boar [<f0c333b9>] sd_rw_intr+0x1f1/0x21b [sd_mod]
Aug 25 08:27:46 boar [<f0c6d3b9>] scsi_finish_command+0x73/0x77 [scsi_mod]
Aug 25 08:27:46 boar [<024cbfa2>] blk_done_softirq+0x4d/0x58
Aug 25 08:27:46 boar [] __do_softirq+0x84/0x109
Aug 25 08:27:46 boar [<0242650d>] do_softirq+0x36/0x3a
Aug 25 08:27:46 boar [<024050b7>] do_IRQ+0xad/0xb6
Aug 25 08:27:46 boar [<024023fa>] default_idle+0x0/0x59
Aug 25 08:27:46 boar [<0240242b>] default_idle+0x31/0x59
Aug 25 08:27:46 boar [<024024b1>] cpu_idle+0x5e/0x74
Aug 25 08:27:46 boar =======================
Aug 25 08:27:46 boar Code: 5d c3 55 57 89 c7 56 89 ce 53 bb 01 00 00 00 83 ec 0c \
8b 68 3c 83 7f 20 00 8b 45 00 8b 00 89 44 24 04 8b 45 04 89 04 24 8b 40 04 <8b> \
40 28 89 44 24 08 0f 85 86 00 00 00 f6 47 10 01 75 0a 85 c9
Aug 25 08:27:46 boar EIP: [<f0ce6507>] clone_endio+0x29/0xc6 [dm_mod] \
SS:ESP0068:0b483e38
Aug 25 08:27:46 boar Kernel panic - not syncing: Fatal exception in interrupt
```

You can save the oops in a file to be able to provide it when asking for technical support.

11.2.3 Finding a Kernel Function That Caused the D Process State

If there are too many processes in the D state and you can't find out what is happening, issue the following command:

```
# objdump -Dr /boot/vmlinux-'uname -r' >/tmp/kernel.dump
```

and then get the process list:

```
# ps axfwln
  F UID   PID  PPID PRI NI  VSZ  RSS  WCHAN STAT TTY TIME COMMAND
100   0 20418 20417  17  0 2588  684      - R    ?   0:00 ps axfwln
100   0     1     0   8  0 1388  524 145186 S    ?   0:00 init
040   0  8670     1   9  0 1448  960 145186 S    ?   0:00 syslogd -m 0
040   0  8713     1  10  0 1616 1140 11ea02 S    ?   0:00 crond
```

Look for a number under the WCHAN column for the process in question. Then, open `/tmp/kernel.dump` in an editor, find that number in the first column and then scroll backward to the first function name, which can look like this:

```
"c011e910 <sys_nanosleep>:"
```

Then you can tell if the process “lives” or is blocked into the found function.

11.3 Container Management Issues

This section includes recommendations on how to solve certain container issues.

11.3.1 Failure to Start a Container

An attempt to start a container fails.

11.3.1.1 Solution 1

If there is a message on the system console: `IP address is already used`, issue the `cat /proc/vz/veinfo` command. The information about the container numeric identifier, container class, number of container’s processes and container IP address shall be displayed for each running container. This shall also demonstrate that your container is up, i.e. it must be running without any IP address assigned. Set its IP address using the command:

```
# prlctl set <CT_name> --ipadd <IP_address>
```

where <CT_name> is the container name and <IP_address> is the desired IP address.

11.3.1.2 Solution 2

The container might be configured incorrectly. Try to validate the container configuration and find out what parameters have caused the error. Set appropriate values using the `prlctl set` command.

11.3.1.3 Solution 3

The container might have used all its disk quota (disk space). Check the quota (see *Managing Disk Quotas* on page 77) and adjust its parameters if needed.

11.3.1.4 Solution 4

Run the `prlctl console` utility to log in and get access to the container console. The utility will provide container startup/shutdown output that may be used to pinpoint the problem. For example:

```
# prlctl console MyCT
```

where `MyCT` is the container name.

11.3.2 Failure to Access a Container from Network

11.3.2.1 Solution 1

The IP address assigned to the container might be already in use in your network. Make sure it is not. The problem container address can be checked by issuing the following command:

```
# grep IP_ADDRESS /etc/vz/conf/<UUID>.conf
IP_ADDRESS="10.0.186.101"
```

The IP addresses of other containers, which are running, can be checked by running

```
# cat /proc/vz/veinfo
```

11.3.2.2 Solution 2

Make sure the routing to the container is properly configured. Containers can use the default router for your network, or you may configure the hardware node as router for its containers.

11.3.3 Failure to Log In to a Container

The container starts successfully, but you cannot log in.

11.3.3.1 Solution 1

You are trying to connect via SSH, but access is denied. Probably you have not set the password of the `root` user yet or there is no such user. In this case, use the `prlctl set --userpasswd` command. For example, for the container `MyCT` you might issue the following command:

```
# prlctl set MyCT --userpasswd root:secret
```

11.3.3.2 Solution 2

Check forwarding settings by issuing the following command:

```
# cat /proc/sys/net/ipv4/conf/venet0/forwarding
```

If it is `0` then change it to `1` by issuing the following command:

```
# echo 1 > /proc/sys/net/ipv4/conf/venet0/forwarding
```

11.4 Getting Technical Support

You can get technical support via the mailing list, bug tracker, and support forum. For more information, visit https://virtuozzo.com/support/.