OpenVZ User’s Guide

July 26, 2016

Parallels IP Holdings GmbH
Vordergasse 59
8200 Schaffhausen
Switzerland
Tel: + 41 52 632 0411
Fax: + 41 52 672 2010
http://www.virtuozzo.com

Table of Contents

Chapter 1. Learning OpenVZ Basics

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

1.1. OpenVZ Overview

OpenVZ 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.

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

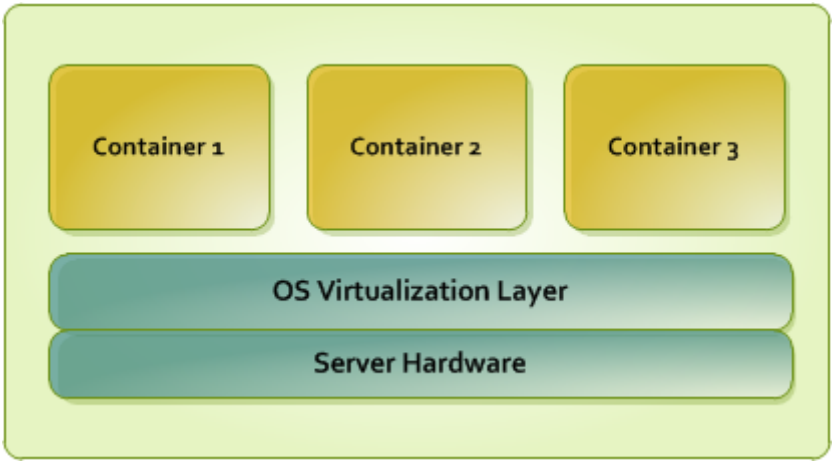
- standardize server hardware platforms,
- effectively consolidate server resources,
- consolidate and support legacy OSs and applications,
- streamline server and application deployment, maintenance, and management,
- simplify software testing and development,
- optimize server and application availability.

1.2. OS Virtualization Layer

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

1.2.1. 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.

1.2.2. OpenVZ Containers

From the point of view of applications and container users, each container is an independent system. This independence is provided by the OpenVZ 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 OpenVZ 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*.

Table 1.1. Key Container directories and files

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 16 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.

1.2.2.1. OpenVZ 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 32
RAM	Up to total amount of physical RAM on the host	Up to 256 GB
Disk drives	Up to 15: hard disk drives mapped to QCOW2 image files and DVD drives mapped to ISO image files, up to 16 TB each	
Network Interfaces	Up to 15	

1.2.3. Templates

A template (or a package set) in OpenVZ is a set of original application files repackaged for use by OpenVZ. Usually, it is just a set of RPM packages for Red Hat like systems. OpenVZ 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. OpenVZ 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.

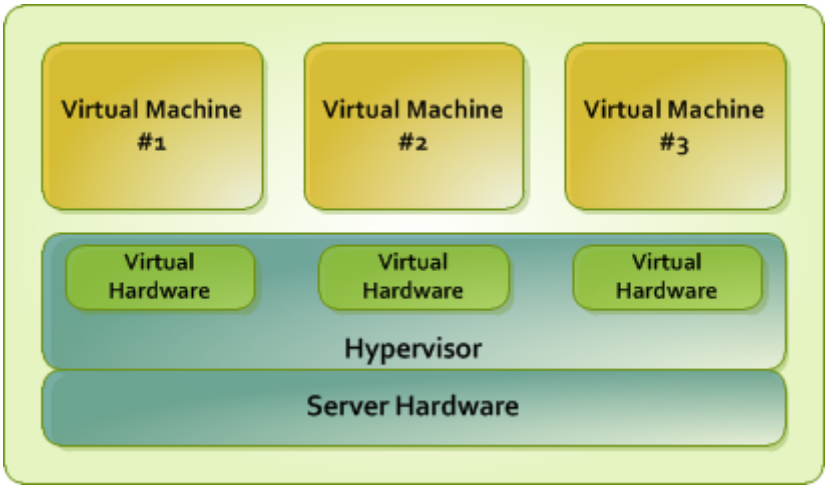
1.3. Hardware Virtualization Layer

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

1.3.1. Hardware Virtualization Basics

OpenVZ 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, OpenVZ 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, OpenVZ 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.

1.3.2. OpenVZ 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 OpenVZ 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 OpenVZ 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.

1.3.2.1. Intel and AMD Virtualization Technology Support

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

1.3.3. Virtual Machine Hardware

A OpenVZ 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 Section 2.1.3, “Creating Virtual Machines” on page 17).

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

CPU	Up to 32
RAM	Up to 256 GB
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 or a physical floppy drive
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
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

1.3.4. 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 OpenVZ 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.

1.3.5. Support of Virtual and Real Media

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

1.3.5.1. Supported Types of Hard Disks

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

1.3.5.2. Virtual Hard Disks

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

OpenVZ 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.

1.3.5.3. Split disks

A virtual disk of either format can be a single-piece disk or a split disk. A split disk is cut into 2 GB pieces and is stored as a single `.hdd` file.

1.3.5.4. CD/DVD Discs and Images

OpenVZ can access real CD/DVD discs and images of CD/DVD discs.

OpenVZ has no limitations on using multi-session CD/DVD discs. A virtual machine can play back audio CDs without any limitations on copy-protected discs.

If your server has a recordable optical drive, you can use it to burn CD or DVD discs in a virtual machine.

OpenVZ supports whatever CD/DVD disc images are supported by the guest OS.

1.4. OpenVZ Configuration

OpenVZ 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.

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

1.5. Resource Management

OpenVZ 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 OpenVZ than for a standalone server since server resource utilization in such a system is considerably higher than that in a typical system.

1.6. Physical Server Availability Considerations

The availability of a physical server running OpenVZ 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 OpenVZ 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. 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 AMD64/Intel EM64T) EZ OS Template
```

2.1.2. Creating Containers

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

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

OpenVZ 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
```

Notes:

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 Section 7.3.1, “Using Golden Image Functionality” on page 104.

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 Section 7.4.1, “Enabling VNC Access to Virtual Machines” on page 107).

When choosing a distribution to install, have in mind that OpenVZ 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 Section 2.2.1, “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 2012 R2
- Windows Server 2012
- Windows Server 2008 R2 with Service Pack 1
- CentOS 7.x (x64)
- CentOS 6.x (x64)
- Debian 8.x (x64)
- Debian 7.x (x64)
- Ubuntu 16.04 LTS (x64)

- Ubuntu 15.10 (x64)
- Ubuntu 14.04 LTS (x64)
- Virtuozzo Linux 7.x (x64)
- Virtuozzo Linux 6.x (x64)
- openSUSE 42.x (x64)
- Fedora 23 (x64)

2.1.4.2. Containers

- CentOS 7.x (x64)
- CentOS 6.x (x64)
- Debian 8.x (x64)
- Debian 7.x (x64)
- Ubuntu 16.04 LTS (x64)
- Ubuntu 15.10 (x64)
- Ubuntu 14.04 LTS (x64)
- Virtuozzo Linux 7.x (x64)
- Virtuozzo Linux 6.x (x64)
- openSUSE 42.x (x64)
- Fedora 23 (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

OpenVZ 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: mounting the guest tools image, 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 OpenVZ Guest Tools

OpenVZ guest tools enable you to configure running virtual machines from the physical host. These operations include:

- Running commands in VMs with the `prlctl exec` command.
- Setting passwords for users in VMs with the `prlctl set --userpasswd` command. If the user does not exist, it will be created.
- Obtaining and changing VM network settings.

To install OpenVZ guest tools in a Linux or Windows virtual machine `MyVM`, do the following:

1. Run the `prlctl installtools` command on the host. For example:

   ```
   # prlctl installtools MyVM
   ```

   The guest tools image shipped with OpenVZ will be mounted to the virtual machine's optical drive.

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, the installer will run automatically. Otherwise, navigate to the optical drive and launch the installer manually.

Note: OpenVZ 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.

2.2.3. 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
```

Notes:
1. You can configure the network settings only for virtual machines that have OpenVZ 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 OpenVZ, 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 OpenVZ 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 `onboot` startup parameter for virtual machines and containers. Setting this parameter to `yes` 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 --onboot yes
```

- For the virtual machine `MyVM`:

```
# prlctl set MyVM --onboot yes
```

Notice that the `onboot` 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 OpenVZ 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 both stopped and running virtual machines and 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 Section 2.2.3, “Configuring Network Settings” on page 19).

2.5.1. 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 name of the resulting virtual machine). To store the files of the `ClonedVM` virtual machine in a custom directory, you can run the following command:

  ```
  # prlctl clone MyVM --name ClonedVM --dst /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 `MyCT2`, you can execute the following command:

```
# prlctl clone MyCT1 --name MyCT2 --dst /vz/private/customCTs
```

2.6. Suspending Virtual Machines and Containers

OpenVZ 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

OpenVZ 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...
```

Notes:

1. You can use the `prlctl exec` command only inside virtual machines that have OpenVZ 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.

Option	Description
`Home`	Directory storing virtual machine files.
`Guest tools`	Shows whether OpenVZ 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 Templates

A template in OpenVZ 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

These operations are described in the following subsections in detail.

2.10.1. Creating Templates

In OpenVZ, 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.

2.10.2. 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 centos1_tmpl
{92cd331e-0572-46ac-8586-f19b8d029c4d}  centos          CT ct201_tmp1
{fc40e38e-8da4-4b26-bb18-6098ec85f7b4}  debian          VM deb7_tmpl
{0dea5912-b114-45a9-bd1a-dc065c1b8e9f}  ubuntu          VM ubuntu1_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.

2.10.3. Deploying Templates

To convert a template into a virtual machine or container, use the `--ostemplate` option of the `prlctl create` command. For example, to convert the `template1` template to the `ConvertedVM` virtual machine, you can run this command:

```
# prlctl create ConvertedVM --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     ConvertedVM
```

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

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

2.11. Managing Snapshots

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

2.11.1. Creating Snapshots

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

2.11.1.1. 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.

2.11.1.2. 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.

2.11.1.3. 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**.

2.11.1.4. 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.

2.11.2. 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.
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.

2.11.3. 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 Section 2.11.1, “Creating Snapshots” on page 28).

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.

2.11.4. 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.

2.12. Migrating Virtual Machines and Containers

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

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.

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 destserver.com/MyVM localhost
```

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

Once migration is complete, the original virtual machine is removed from the source server. However, you can keep the original VM with the `--keep-src` option.

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

2.12.1. Migrating Virtual Machines and Containers Between OpenVZ Servers

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

- Offline migration for stopped and suspended containers and virtual machines.
- Online (live) migration for running containers and running and paused virtual machines.

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

2.12.1.1. 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.

2.12.1.2. 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, OpenVZ 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, OpenVZ provides connection tunneling between the source and destination servers. Tunnelling increases migration 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`.

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.

- 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.

2.13. Performing Container-specific Operations

This section provides the description of operations specific to containers.

2.13.1. 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:

```
# vzctl reinstall MyCT
```

Note: If any of the container application templates cannot be added to the container in a normal way, reinstallation will fail. This may happen, for example, if an application template was added to the container using the `--force` option of the `vzpkgadd` or `vzpkg install` command.

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

2.13.1.1. 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.

2.13.2. 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 OpenVZ is started:

```
# lsmod
```

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

```
# vzctl set MyCT --devices c:10:200:rw --save
```

3. Create the corresponding device inside the container and set the proper permissions:

```
# prlctl exec MyCT mkdir -p /dev/net
# prlctl exec MyCT mknod /dev/net/tun c 10 200
# prlctl exec MyCT chmod 600 /dev/net/tun
```

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.

2.13.3. 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
```

Note: If the container is running, restart it for the changes to take effect.

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.

2.13.4. 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.

2.13.5. Adding Multiple Virtual Disks to Containers

Even though new containers are created with just one virtual hard disk, you can add more disks to a container and keep the corresponding ploop images at locations of your choice, be it directly attached HDDs or SSDs. Such functionality allows creating more flexible containers, in which, for example, the operating system is kept on a fast SSD and user content is stored on a capacious HDD.

To add a virtual hard disk to a container, whether stopped or running, use the `prlctl set --device-add hdd` command. For example:

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

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,
- image location: `/hdd/MyCT`,
- size: 102400 MB,
- mount point inside the container `MyCT`: `/userdisk`. A corresponding entry is also added to container's `/etc/fstab` file.

2.13.6. 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.

2.13.7. Creating SimFS-based Containers

In OpenVZ, 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 `vzctl create <CT_name>`.

The limitations of simfs in OpenVZ are:

1. No support for first- or second-level quotas.

2. No support for live migration of simfs-based containers.

2.14. Performing Virtual Machine-specific Operations

This section focuses on operations specific to virtual machines.

2.14.1. 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.
```

2.14.2. Managing Virtual Machine Devices

OpenVZ 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

2.14.2.1. 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:

- You can connect up to 4 IDE devices, and 8 SCSI devices (virtual disks or CD/DVD-ROM drives) to a virtual machine.
- 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.

2.14.2.2. Initialize a Newly Added Disk

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 file system you want to use on this disk. It is recommended to use `ext3` or `ext2`.

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.

2.14.2.3. Configuring Virtual Devices

In OpenVZ, 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
  fdd0 () real='/dev/fd0' state=disconnected
  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 () ide:1 real='Default CD/DVD-ROM'
  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 Devices

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 OpenVZ, 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.
```

2.14.2.4. 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:** IDE and SCSI disks can be removed from stopped virtual machines only.
`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 example, to obtain the list of virtual devices in the `MyVM` virtual machine, 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
  fdd0 () real='/dev/fd0' state=disconnected
  hdd0 () scsi:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk.hdd' type='expanded' subtype=virtio-scsi
  hdd1 () ide:0 image='/vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk1.hdd' type='expanded'
  cdrom0 () ide:1 real='Default CD/DVD-ROM'
  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.

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 if from the virtual machine using the `--disable` option.

2.14.3. Assigning USB Devices to Virtual Machines

In OpenVZ, you can assign a USB device to a virtual machine so that the device is automatically connected to the virtual machine when you connect the USB device to the hardware node or start the virtual machine. To assign a USB device to a virtual machine, you need to specify two parameters:

- **ID of the USB device**. To get this information, use the `prlsrvctl info` command, for example:

```
# prlsrvctl info
...
Hardware info:
         hdd                                                '/dev/sda'
hdd-part NTFS                                               '/dev/sda2'
hdd-part Linux                                              '/dev/sda3'
hdd-part Linux                                              '/dev/sda5'
hdd-part Linux swap                                         '/dev/sda6'
 cdrom Optiarc DVD RW AD-7260S                             '/dev/scd0'
  net enp0s5                                                'enp0s5'
  usb Broadcom - USB Device 3503                           '2-1.4.3|0a5c|3503|full|KM|Empty'
  sb Broadcom - USB Device 3502                            '2-1.4.2|0a5c|3502|full|KM|Empty'
  usb LITEON Technology - USB Multimedia Keyboard '1-1.6|046d|c312|low|KM|Empty'
  serial /dev/ttyS0                                         '/dev/ttyS0'
 serial /dev/ttyS1                                          '/dev/ttyS1'
```

All USB devices available on the hardware node are listed in the **Hardware info** section and start with `usb`.

- ID of the virtual machine. To get this information, use the `prlctl list --info` command, for example:

```
# prlctl list --info
ID: {d8d516c9-dba3-dc4b-9941-d6fad3767035}
Name: Windows 7
...
```

  The first line in the command output indicates the virtual machine ID; in our case, it is `{d8d516c9-dba3-dc4b-9941-d6fad3767035}`.

Once you know the USB device and virtual machine IDs, you can use the `prlsrvctl usb set` command to assign the USB device to the virtual machine. For example:

```
# prlsrvctl usb set '1-1.6|046d|c312|low|KM|Empty' {d8d516c9-dba3-dc4b-9941-\
d6fad3767035}
The server has been successfully configured.
```

This command assigns the USB device `LITEON Technology - USB Multimedia Keyboard with ID '1-1.6|046d|c312|low|KM|Empty'` to the virtual machine with ID `{d8d516c9-dba3-dc4b-9941-`

d6fad3767035}. When running the command, remember to specify the single quotes and curly brackets with the USB device and virtual machine IDs, respectively.

To check that the USB device has been successfully assigned to the virtual machine, use the `prlsrvctl usb list` command:

```
# prlsrvctl usb list
Broadcom - USB Device 3503                  '2-1.4.3|0a5c|3503|full|KM|Empty'
Broadcom - USB Device 3502                  '2-1.4.2|0a5c|3502|full|KM|Empty'
LITEON Technology - USB Multimedia Keyboard '1-1.6|046d|c312|low|KM|Empty' \
      {d8d516c9-dba3-dc4b-9941-d6fad3767035}
```

The command output shows that the USB device with ID `'1-1.6|046d|c312|low|KM|Empty'` is now associated with the virtual machine with ID `{d8d516c9-dba3-dc4b-9941-d6fad3767035}`. This means that the device is automatically connected to the virtual machine every time you start this virtual machine and connect the device to the hardware node.

To remove the assignment of the USB device with ID `'1-1.6|046d|c312|low|KM|Empty'`, use the `prlsrvctl usb del` command:

```
# prlsrvctl usb del '1-1.6|046d|c312|low|KM|Empty'
The server has been successfully configured.
```

When assigning USB devices to virtual machines, keep in mind the following:

- You cannot migrate a running virtual machine having one or more USB devices assigned.
- After migrating a stopped virtual machine, all its assignments are lost.
- All USB assignments are preserved if you restoring a virtual machine to its original location and are lost otherwise.
- The USB device assignment and a virtual machine is created for the user currently logged in to the system.

2.14.4. 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 OpenVZ 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 OpenVZ 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. Managing Virtual Machines and Containers with virt-manager

Note: This feature is experimental.

As OpenVZ VMs and containers are managed via the libvirt API, you can manage them not only with OpenVZ CLI tools but also with Virtual Machine Manager (`virt-manager`, see https://virt-manager.org/). The detailed instructions are provided at https://kb.virtuozzo.com/en/129047.

Chapter 3. Managing Resources

The main goal of resource control in OpenVZ 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 containers
- CPU hotplug 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 containers, see also Section 3.1.3.4, “CPU Limit Specifics” on page 49.

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 the virtual machine or container may use. For example, to allow the container `MyCT` to use only 2 cores, run 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, OpenVZ 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, OpenVZ 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 containers to use CPUs from specific NUMA nodes only. Let us assume the following:

- 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 (both x86 and x64 versions)

- Linux operating systems based on the RHEL 5 kernel and higher (Red Hat Linux Enterprise 5, CentOS 5, and so on)

Windows

- 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.

3.2. 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 OpenVZ, you can manage virtual disks as follows:

- increase the capacity of your virtual disks,

- reduce the capacity of your virtual disks,
- compact virtual disks (reduce the size they occupy on the physical hard drive),
- change the interface of your virtual disks.

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

3.3.1. Increasing Disk Capacity

If you find that the capacity of the virtual hard disk of your virtual machine or container does not fit your needs anymore, you can increase it using the `prl_disk_tool resize --size` command. For example:

```
# prl_disk_tool resize --hdd /vz/vmprivate/d35d28e5-11f7-4b3f-9065-8fef6178bc5b/ \
harddisk.hdd --size 80G
```

Note: To virtual machines, additional disk space is added as unallocated. You can use standard means (e.g., the Disk Management tool in Windows-based virtual machines) to allocate added space by creating a new or expanding the existing partition.

When increasing the disk capacity, keep in mind the following:

- You can increase the capacity of virtual disks of both stopped and running virtual machines.
- The virtual machine using the virtual disk you want to configure must not have any snapshots. In this case, you should delete all existing snapshots and run the command again. To learn how to delete snapshots of a virtual machine, refer to Section 2.11.4, “Deleting Snapshots” on page 31.
- The capacity of an expanding virtual disk shown from inside the virtual machine or container and the size the virtual disk occupies on the server’s physical disk may differ.

3.3.2. Reducing Disk Capacity

OpenVZ provides a possibility to reduce the size of an expanding virtual disk by setting the limit the disk cannot exceed. To do this, use 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 reducing the disk capacity, keep in mind the following:

- You can only reduce the capacity of virtual disks of stopped virtual machines.
- The virtual machine using the virtual disk you want to configure must not have any snapshots. In this case, you should delete all existing snapshots and run the command again. To learn how to delete snapshots of a virtual machine, refer to Section 2.11.4, “Deleting Snapshots” on page 31.
- The capacity of an expanding virtual disk shown from inside the virtual machine or container and the size the virtual disk occupies on the server’s physical disk may differ.
- You cannot reduce XFS filesystems (the default choice for CentOS 7 and Red Hat Enterprise Linux 7).

3.3.2.1. Checking the Minimum Disk Capacity

If, before reducing disk capacity, you want to know the minimum 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
...
```

3.3.3. Compacting Disks

In OpenVZ, you can decrease the space your virtual machines and containers occupy on the physical server's disk drive by compacting their virtual disks. Compacting virtual disks allows you to save your server's disk space and host more virtual machines and containers on the server.

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`).

3.3.4. Managing Virtual Machine Disk Interfaces

By default, any virtual machine is created with a SCSI (Small Computer System Interface) virtual hard disk. If necessary, you can change the interface type of a disk from SCSI to IDE (Integrated Drive Electronics) 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 fdd0 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 (both virtual hard disks and CD/DVD-ROM drives) you can add to a virtual machine is given below:

- 4 IDE devices
- 8 SCSI devices

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.
```

Notes:

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 OpenVZ:

- 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 OpenVZ.

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.

Parameter	Description
`rate`	If traffic shaping is turned on, this parameter specifies the bandwidth guarantee for virtual machines and containers.
`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

OpenVZ 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 OpenVZ 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 OpenVZ 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 OpenVZ, 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.

Notes:

1. Traffic within a host cannot be shaped in the current version of OpenVZ. 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 OpenVZ.

The following parameters control traffic shaping in OpenVZ:

- `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 OpenVZ 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 OpenVZ systems.

3.5.1. Configuring Priority Levels for Virtual Machines and Containers

In OpenVZ, 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 OpenVZ, 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 OpenVZ, 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 OpenVZ, 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 Section 3.5.3.1, “Setting the Direct Access Flag Inside Containers” on page 62 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 to the container `MyCT` and the virtual machine `MyVM`, check the cgroup `/sys/fs/cgroup/beancounter/<UUID>/beancounter.iopslimit.speed`. For example:

```
# cat /sys/fs/cgroup/beancounter/`vzlist MyCT -Ho uuid`/beancounter.iopslimit.speed
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 OpenVZ, 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.

3.6. 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

OpenVZ 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,
- kernel same-page merging.

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 Section 3.7.3, “Enabling Virtual Machine Memory Hotplugging” on page 67.

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 Section 3.7.3, “Enabling Virtual Machine Memory Hotplugging” on page 67). 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 Section 3.7.1, “Configuring Virtual Machine Memory Size” on page 66), 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 40% by default. To change the default value, use the `prlctl set --memguarantee` command. For example:

```
# prlctl set MyVM --memguarantee 80
```

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.

3.7.5. Optimizing Virtual Machine Memory with Kernel Same-Page Merging

To optimize memory usage by virtual machines, OpenVZ 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.

KSM can be managed by means of two services:

- The `ksm` service that starts and stops the KSM kernel thread.
- The `ksmtuned` service that controls and tunes the `ksm` using the parameters set in the `/etc/ksmtuned.conf` file.

You can start the `ksm` and `ksmtuned` services by executing the following commands:

```
service ksm start
service ksmtuned start
```

To check that the feature works, you can check the number of currently shared memory pages in `/sys/kernel/mm/ksm/pages_sharing` with virtual machines running. For example:

```
# cat /sys/kernel/mm/ksm/pages_sharing
3159
```

To stop the services, run:

```
service ksm stop
service ksmtuned stop
```

Notes:

1. It is not advisable to use the KSM daemon if CPU resources may become a bottleneck.
2. It is recommended to avoid cross-node memory merging when KSM is in use as this may result in a significant performance drop after a lot of pages are shared. To disable merging pages across NUMA nodes, replace the contents of `/sys/kernel/mm/ksm/merge_across_nodes` with `0`.
3. Using KSM may affect virtual machine security. For more details, see http://kb.virtuozzo.com/en/126594.

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 OpenVZ. These files are used when a new container is being created (for details, see Section 1.2.2, “OpenVZ Containers” on page 9). 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 OpenVZ 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

OpenVZ 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 Section 3.9.3, “Parameters Applied from Configuration Samples” on page 72 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/config.pvs /etc/parallels/samples/configMySQL.pvs
```

The latter command copies the `config.pvs` file to the `configMyDB.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 OpenVZ, 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 OpenVZ. 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 OpenVZ, 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 OpenVZ 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 OpenVZ 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 OpenVZ, 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

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 OpenVZ, 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 12
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 OpenVZ 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, OpenVZ registers all physical network adapters available on the server. Once OpenVZ has been successfully installed, you can manage network adapters on the hardware node using native RHEL7 utilities.

Note: You can also use VLAN interfaces instead of physical.

5.2. Networking Modes in OpenVZ

This section describes networking modes available in OpenVZ.

In OpenVZ, any virtual machine or container can operate in one of the two networking modes: host-routed or bridged.

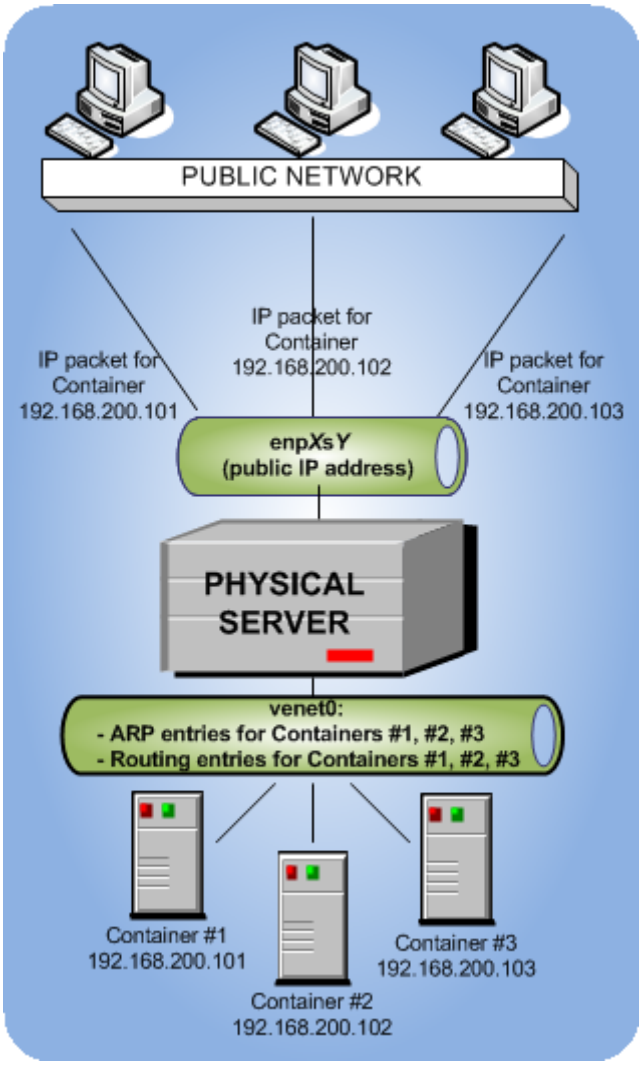
5.2.1. Container Network Modes

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.



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
```

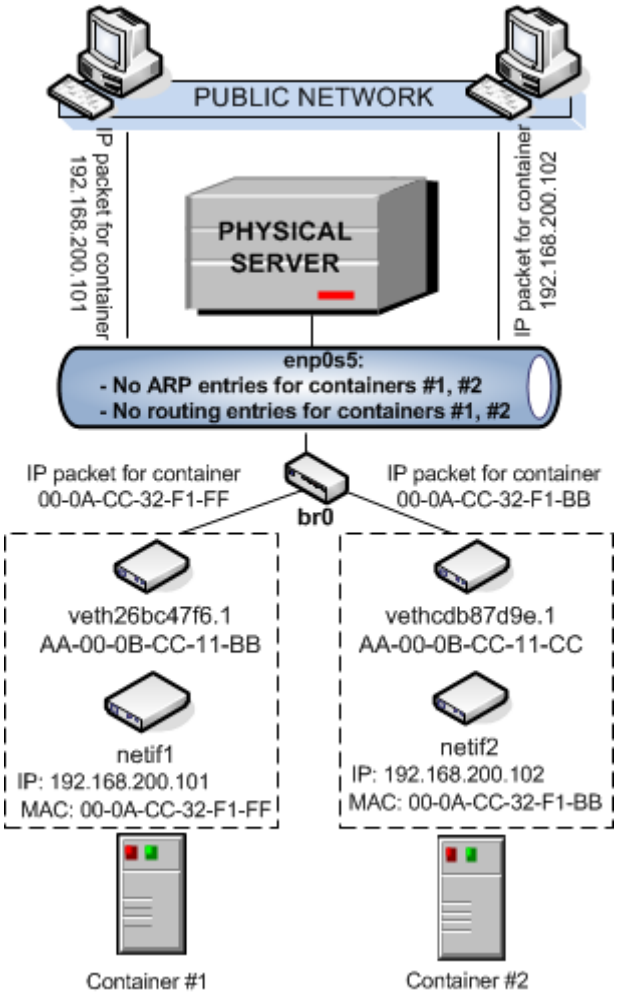
```
192.168.200.103   *               255.255.255.255 UH    1000   0        0 venet0
10.30.0.0         *               255.255.0.0     U     0      0        0 enp0s5
default           virtuozzo.com   0.0.0.0         UG    0      0        0 enp0s5
```

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.



In this configuration:

- Container #1 and container #2 have separate virtual adapters consisting of two network interfaces:
  - An `enp<X>s<Y>` interface in the container (**enp0s5** 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.

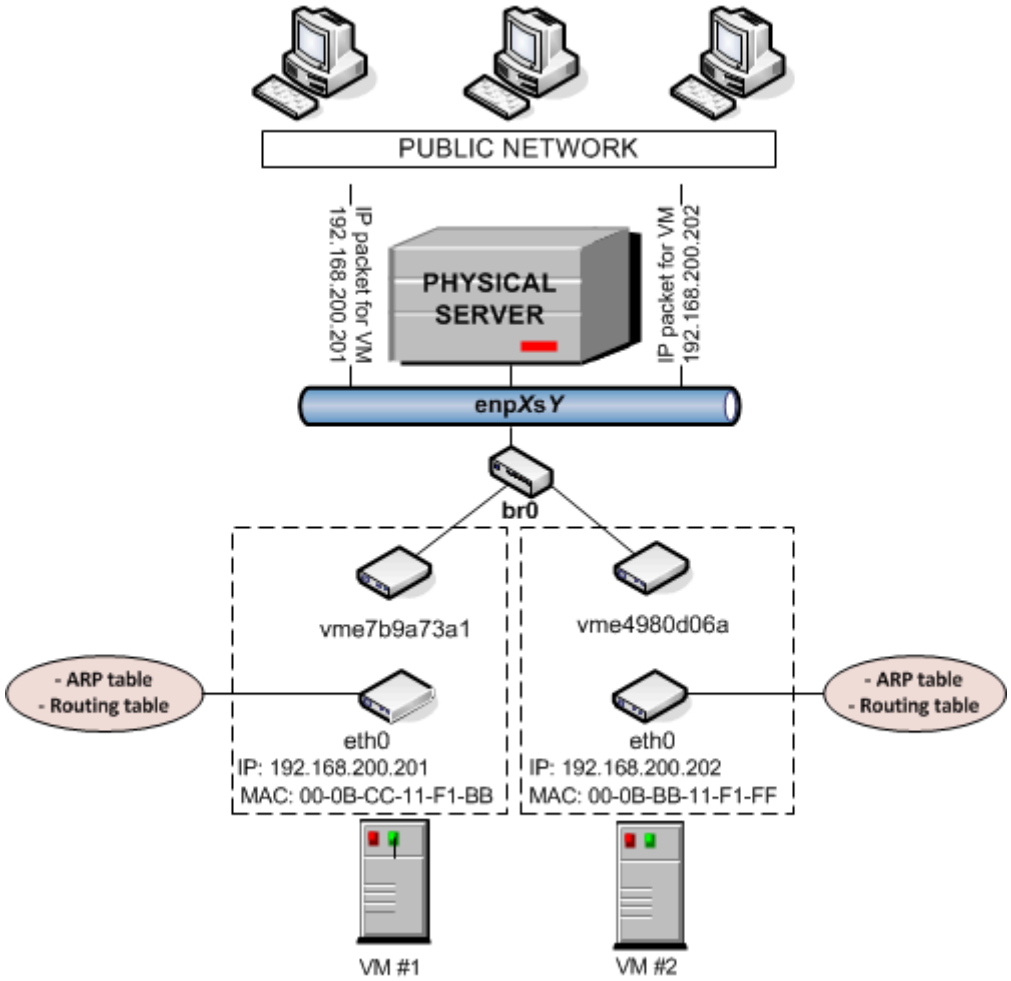
- The `veth` adapters of both containers are bridged through the bridge `br0` to the physical network adapter `enp0s5`.
- All container outgoing traffic comes via the `veth` adapters to the bridge and are then transmitted through the `enp0s5` physical adapter to the destination, according to the routing 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:

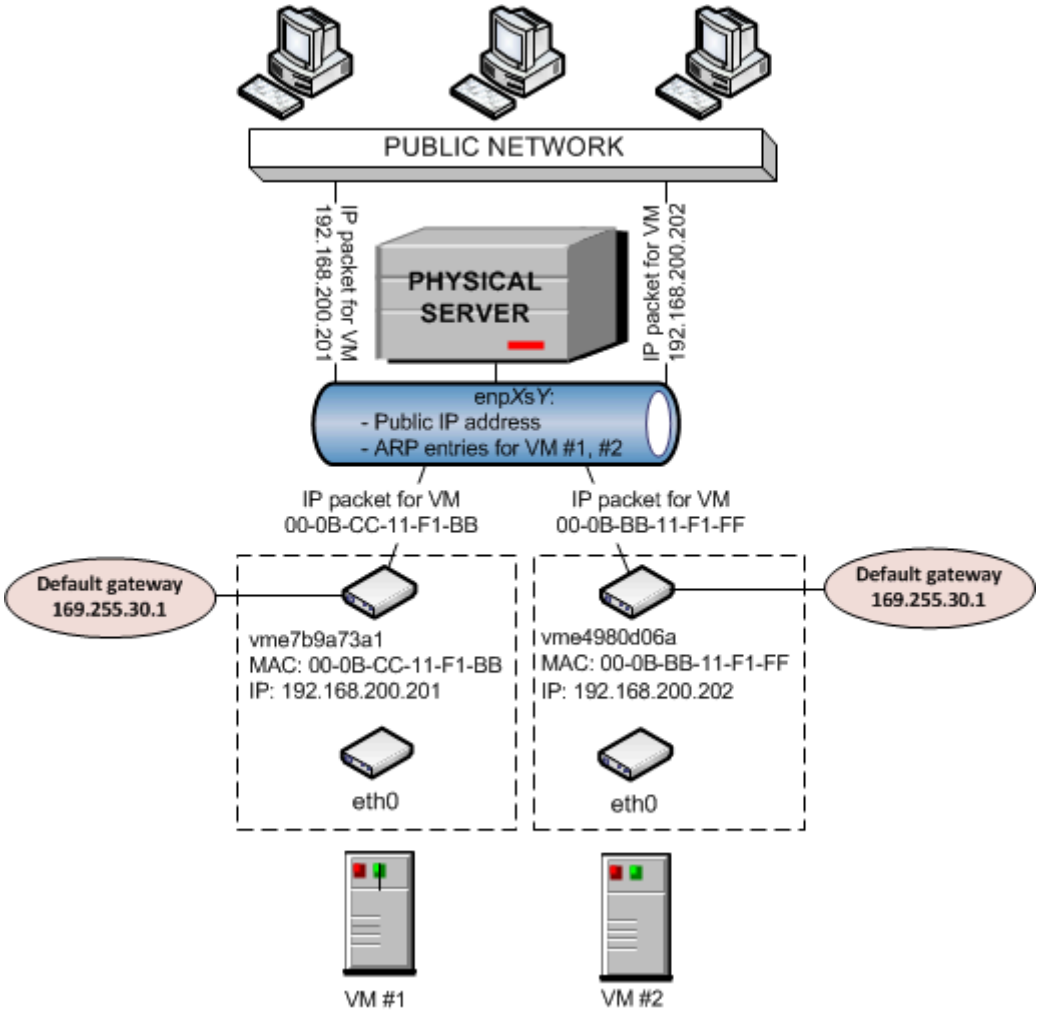
- Each virtual machine has a separate virtual adapter that exposes two interfaces: (1) an `ethX` interface in the virtual machine (`eth0` in the figure) and a `vme` interface on the server (`vme7b9a73a1` and `vme4980d06a` in the figure). Both interfaces are closely linked to each other, which means that 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 `eth0`.
- All outgoing data packets are sent from the virtual machines through the bridge and `eth0` physical adapter to the destination, according to their routing tables.
- All incoming data packets for VM #1 and VM #2 reach the eth0 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.



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 `eth0` 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` 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` adapters can be bridged among themselves and with other devices. If several `veth` adapters are united into a bridge, this bridge can be used to handle network traffic for the virtual machines and containers whose `veth` adapters are included in the bridge.
- Due to the fact that `veth` 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` 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
```

Notes:

1. You can configure the network settings only of virtual machines that have OpenVZ guest tools installed.
2. Network adapters operating in the routed mode must have at least one static IP address assigned.
3. To assign network masks to containers operating in the `venet0` networking mode, you must set the `USE_VENET_MASK` parameter in the `/etc/vz/vz.conf` configuration file to `yes`.
4. Containers can have only one network adapter operating in the host-routed mode. This adapter is automatically created when you create a virtual machine.

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 connected to the bridged mode (see Section 5.4.3.3, “Connecting Virtual Machines to Virtual Networks” on page 95 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

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. OpenVZ 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, OpenVZ creates the following virtual networks on the server:

- **Bridged** virtual network that is connected to one of the physical adapters on the hardware node (as a rule, `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 Section 5.4.1.3, “Configuring Virtual Network Parameters” on page 89).

5.4.1.2. Creating Network Bridges for Physical Network Adapters

By default, each OpenVZ server has one network bridge `br0` set up for a single physical network adapter. If your server has more than one physical network adapter, you need to create a network bridge for each of those that you need to connect to a bridged virtual network.

Assuming that two physical network adapters `enp0s5` and `enp0s6` and a bridge `br0` exist on the server, you can create a bridge `br1` for `enp0s6` as follows:

1. In the `/etc/sysconfig/network-scripts/` directory, create two configuration files: one for the physical adapter `enp0s6` and one for the bridge `br1`. You can use the configuration files for `enp0s5` and `br0`, respectively, as a basis:

```
cat /etc/sysconfig/network-scripts/ifcfg-enp0s5 > /etc/sysconfig/network-scripts/ \
ifcfg-enp0s6
cat /etc/sysconfig/network-scripts/ifcfg-br0 > /etc/sysconfig/network-scripts/ \
ifcfg-br1
```

2. In `ifcfg-enp0s6`, set `DEVICE` to `enp0s6`, `BRIDGE` to `br1`, and `UUID` to empty.

3. In `ifcfg-br1` set `DEVICE` to `br1`, `NAME` to `enp0s6`, and `UUID` to empty.

4. Restart network to apply changes:

```
/etc/init.d/network restart
```

After the network bridge is set up, you can bind a virtual network to it.

5.4.1.3. Configuring Virtual Network Parameters

OpenVZ 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 Section 5.4.1.2, “Creating Network Bridges for Physical Network Adapters” on page 88.

- The description of the virtual network.

All these operations can be performed using the `prlsrvctl` utility. Let us assume that you want 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.

Let us assume the following:

- The `enp0s6` physical adapter and the `network1` virtual network exist on the hardware node. For information on creating virtual networks, see Section 5.4.1.1, “Creating Virtual Networks” on page 88.
- 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 Section 5.4.1.2, “Creating Network Bridges for Physical Network Adapters” on page 88.
- The container `MyCT` is connected to the `network1` virtual network. Detailed information on joining virtual machines and containers to virtual networks is given in Section 5.4.2.3, “Connecting Containers to Virtual Networks” on page 93 and Section 5.4.3.3, “Connecting Virtual Machines to Virtual Networks” on page 95.

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 Containers

OpenVZ provides you with ample opportunities of configuring `veth` virtual network adapters in containers and including them in different network environments. This section shows you the way to perform the following operations:

- Create new virtual network adapters in containers and delete existing ones.
- Configure the parameters of an existing virtual network adapter.
- Join container virtual network adapters to virtual networks.

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

5.4.2.1. Creating and Deleting veth Network Adapters

By default, any container on the hardware node starts functioning in the `venet0` mode right after its creation. However, at any time you can create additional virtual adapters for containers and set them to work in the bridged mode. You can do this using the `--netif_add` option of the `prlctl set` command.

Let us assume that you wish to create a new virtual adapter with the name of `netif1` in the container `MyCT` and make it function in the bridged mode. To do this, run the following command:

```
# prlctl set MyCT --netif_add netif1
```

The settings of the newly created virtual adapter are saved as the value of the `NETIF` parameter in the configuration file of the container `MyCT` (`/etc/vz/conf/26bc47f6-353f-444b-bc35-b634a88dbbcc.conf`). So, you can use the following command to display the parameters assigned to the `veth` network adapter in the container `MyCT`:

```
# grep NETIF /etc/vz/conf/26bc47f6-353f-444b-bc35-b634a88dbbcc.conf
NETIF="ifname=netif1,mac=00:1C:42:63:B3:12,host_mac=FE:18:51:6C:0B:A8, \
network=Bridged,configure=none"
```

As you can see, the parameters set for the `veth` virtual network adapter during its creation are the following:

- `ifname`, name set for the `veth` Ethernet interface in the container `MyCT`. You specified this name when creating the container virtual network adapter.
- `mac`, MAC address assigned to the `veth` Ethernet interface in the container `MyCT`.
- `host_mac`, MAC address assigned to the `veth` Ethernet interface on the hardware node.

`ifname` is the only mandatory parameter that you need to specify when creating a container virtual network adapter. All the other parameters are optional and generated by OpenVZ automatically, if not indicated.

At any time, you can remove the `veth` virtual network adapter from the container `MyCT` by executing the following command:

```
# prlctl set MyCT --netif_del netif1
```

5.4.2.2. Configuring veth Adapter Parameters

While functioning in the bridged mode, each container virtual network adapter appears as a full participant on the network to which it is connected and needs to have its own identity on this network.

Fist of all, to start functioning on a TCP/IP network, a `veth` virtual adapter should be assigned an IP address. This can be done as follows:

```
# prlctl set MyCT --ifname netif1 --ipadd 192.168.144.123
```

This command sets an IP address `192.168.144.123` for the `netif1` adapter in the container `MyCT`. If you want to use the Dynamic Host Configuration Protocol (DHCP) to make the `netif1` adapter of the container `MyCT` automatically receive TCP/IP configuration settings, you can issue the following command instead:

```
# prlctl set MyCT --ifname netif1 --dhcp yes
```

Any static IP address assigned to the container virtual network adapter can be removed by executing the following command:

```
# prlctl set MyCT --ifname netif1 --ipdel 192.168.144.123
```

You can also delete all IP addresses set for the container `MyCT` at once:

```
# prlctl set MyCT --ifname netif1 --ipdel all
```

You may also wish to set the following parameters for a container network adapter:

- A DNS server that the container virtual adapter is supposed to use:

  ```
  # prlctl set MyCT --ifname netif1 --nameserver 192.168.100.111
  ```

- A gateway to be used for routing the traffic of the container virtual adapter:

  ```
  # prlctl set MyCT --ifname netif1 --gw 192.168.111.1
  ```

5.4.2.3. Connecting Containers to Virtual Networks

With the implementation of `veth` virtual adapters allowing containers to function as full participants on the network, it has become possible to include containers in a wide range of network configurations the most common of which are Ethernet networks. The process of connecting `veth` virtual network adapters to an Ethernet network is carried out using certain physical, respectively, available on the server and involves completing the following tasks:

1. Creating a virtual network that will act as an intermediary between the `veth` adapters and the physical adapter.
2. Connecting the `veth` virtual adapter you want to include in an Ethernet network to the virtual network.
3. Joining the virtual network where the `veth` virtual adapters are included to the corresponding physical adapter.

After completing these tasks, the container virtual network adapters will be able to communicate with any computer on the network where they are included and have no direct access to the computers joined to other networks.

For details on creating new virtual networks and joining physical adapters to them, see Section 5.4.1.1, “Creating Virtual Networks” on page 88 and Section 5.4.1.5, “Connecting Virtual Networks to Adapters” on page 90, respectively. In the example below we assume the following:

- The `enp0s5` physical adapter and the `network1` virtual network exist on the server.
- The `enp0s5` physical adapter is connected to the local Ethernet network and to the `network1` virtual network.
- You want to connect the container `MyCT1` and the container `MyCT2` to the local Ethernet network.

To join the containers `MyCT1` and `MyCT2` to the local Ethernet network behind the `enp0s5` adapter, you need connect these containers to the `network1` virtual network. To do this:

1. Find out the name of the `veth` Ethernet interfaces in the containers `MyCT1` and `MyCT2`:

```
# prlctl list -a -o ctid,netif,netdev
UUID                                    NETIF      NETDEV
{4e10b61a-c775-4611-a9b0-d4b946e820f2}  netif1     veth42ffa4e6
{eb0d3253-7e7a-486a-897f-02bfbd0e4c5b}  netif2     veth42a5246f
```

   The command output shows that the `veth` Ethernet interfaces in the containers `MyCT1` and `MyCT2` have the names of `netif1` and `netif2`, respectively.

   Note: To add a `veth` adapter to a virtual network, you must use the name of its Ethernet interface in the container.

2. Join the `veth` adapters to the `network1` virtual network:

```
# prlctl set MyCT1 --ifname netif1 --network network1
# prlctl set MyCT2 --ifname netif2 --network network1
```

After completing these tasks, the containers `MyCT1` and `MyCT2` will be able to access any of the servers in the network where the `enp0s5` physical adapter is connected.

At any time, you can disconnect the `veth` virtual network adapters of the containers `MyCT1` and `MyCT2` from the `network1` virtual network by executing the following commands:

```
# prlctl set MyCT1 --ifname netif1 --network ""
# prlctl set MyCT2 --ifname netif2 --network ""
```

5.4.3. Managing Adapters in Virtual Machines

This section provides information on how you can manage virtual network adapters in virtual machines. You will learn to do the following:

- 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.

5.4.3.1. Creating and Deleting Virtual Adapters

A virtual machine can have up to 16 virtual network adapters. Each adapter can be connected to a different network. Let us assume that you want 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
Creating net1 (+) type=host-only iface='default' mac=001C42AF3D69
The VM has been successfully configured.
```

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: Windows XP
...
net0 () type=host-only iface='default' mac=001C42566BCF
net1 () type=host-only iface='default' mac=001C42AF3D69
```

At any time, you can remove the newly created network adapter (`net1`) by executing the following command:

```
# prlctl set MyVM --device-del net1
Remove the net1 device.
The VM has been successfully configured.
```

5.4.3.2. Configuring Virtual Adapter Parameters

OpenVZ allows you to configure the following parameters of virtual machine 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
Creating net1 (+) network=Bridged mac=001C422D7400
The VM has been successfully configured.
```

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
Creating net1 (+) network=Bridged mac=001C42C84F3E
The VM has been successfully configured.
```

Configuring IP Parameters

As any other standalone server, each virtual machine 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

Usually, you define all these settings when you create the virtual machine. 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
Creating net1 (+) network=Bridged mac=001C42C84F3E
Enable automatic reconfiguration for this network adapter.
The VM has been successfully configured.
```

Note: You can configure the network parameters only of those virtual machines that have OpenVZ guest tools installed.

5.4.3.3. Connecting Virtual Machines to Virtual Networks

In OpenVZ, you can connect virtual machines to virtual networks of the following types:

- **Bridged** virtual network allows the virtual machine to use one of the physical server's network adapters, which makes it appear as a separate computer on the network the corresponding adapter belongs to.
- **Host-only** virtual network allows the virtual machine to access only the hardware node and the virtual machines 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 hardware node. 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 net0 --network network1
Creating net0 (+) network=network1 mac=001C422D7493
The VM has been successfully configured.
```

Chapter 6. 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:

- OpenVZ software
- virtual machines and containers hosted on the server

6.1. Updating OpenVZ

OpenVZ allows quick and easy updates with the `yum` utility standard for RPM-compatible Linux operating systems. The main components you may need to update are the following:

- utilities and libraries,
- kernel,
- EZ templates.

6.1.1. Updating All Components

The easiest way to update all components of the OpenVZ software is to simply run the `yum update` command. When executed, this command tells the `yum` utility to do the following:

1. Access remote OpenVZ repositories.
2. Check for available updates for the OpenVZ kernel, utilities, libraries, and EZ templates.
3. Install the found updates on your system.

Note that the `yum` utility can only update the packages that are already installed on the server. So if a package is not available on your system, you first need to install the package using the `yum install` command.

6.1.2. Updating Kernel

Updating the OpenVZ kernel requires updating the `vzkernel` and `vzkernel-devel` packages:

```
# yum update vzkernel vzkernel-devel
```

6.1.3. 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!
```

Notes:

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.

6.1.4. Checking for Updates

Before updating any packages, you may want to see which can be updated and to what version. You can do that with the `yum check-update` command. For example:

```
# yum check-update
```

6.1.5. Performing More Actions with yum

The `yum` command allows you to do more than just check for and install updates. Some of the other useful options that may help you in updating OpenVZ are `search`, `list`, `info`, `deplist`, `provide`. For more information on these and other options, see the `yum` manual page.

6.2. 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).

6.3. Updating Containers

OpenVZ 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.

6.3.1. Updating EZ Template Packages in Containers

OpenVZ 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
```

Notes:

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.

6.3.2. 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. OpenVZ 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 7. Advanced Tasks

This chapter collects miscellaneous configuration and management tasks, some of which require a deeper knowledge of Linux and OpenVZ and should be performed with caution.

7.1. Upgrading from OpenVZ to Virtuozzo 7

Currently two methods of upgrading from OpenVZ to Virtuozzo 7 are available:

- Containers created with OpenVZ based on kernels 2.6.18 and 2.6.32 can be migrated to Virtuozzo 7.
- OpenVZ based on kernel 3.10 can be upgraded to Virtuozzo 7.

7.1.1. Migrating Containers from OpenVZ Based on Kernels 2.6.18 and 2.6.32 to Virtuozzo 7

You can migrate containers from a server running OpenVZ based on kernels 2.6.18 and 2.6.32 to a Virtuozzo 7 server by means of the `ovztransfer.sh` script freely available at https://src.openvz.org/scm/ovzl/ovztransfer.git. Do the following:

1. Install the SSH key on the destination server for the root user. To do this, on the source server generate a key with `ssh-keygen -t rsa`, then transfer the key to the destination server with `ssh-copy-id root@<dest_server>`.
2. Clone the repository with the script to the source OpenVZ server with `git clone https://src.openvz.org/scm/ovzl/ovztransfer.git`.
3. Change to the `/ovztransfer` directory and make the script executable with `chmod 755 ovztransfer.sh`.
4. Run the script on the source OpenVZ server as follows:

   ```
   # ./ovztransfer.sh <dest_server> <source_CT1_ID>[:<new_CT1_name>]\
   [ ... <source_CTn_ID>[:<new_CTn_name>]]
   ```

   where `<source_CT_ID>` (the source container ID) and `<new_CT_name>` (the new container name) must both be specified in the old numerical ID format. For example:

   ```
   # ./ovztransfer.sh 192.168.0.10 100:200
   ```

   So, in the example above, `200` will be the name of the resulting ploop-based container on the Virtuozzo 7 server, even though said name looks like an old numerical ID.

7.1.2. Upgrading from OpenVZ Based on Kernel 3.10 to Virtuozzo 7

To upgrdade from OpenVZ based on kernel 3.10 to Virtuozzo 7, run

```
# do-upgrade-vz7
```

In the process, the commercial Virtuozzo 7 packages will be downloaded and installed on your server and a Virtuozzo 7 trial license will be activated.

Warning: This procedure cannot be reverted.

7.2. Configuring Capabilities

Capabilities are sets of bits that permit of splitting the privileges typically held by the root user into a larger set of more specific privileges. The POSIX capabilities are defined by a draft IEEE standard (IEEE Std 1003.1e); they are not unique to Linux or OpenVZ. When the Linux or OpenVZ documentation says "requires root privileges", in nearly all cases it really means "requires a specific capability".

This section documents the tasks that can be achieved using per-container capabilities in OpenVZ and all configurable capabilities.

7.2.1. Available Capabilities for Containers

This section lists all the capabilities that can be set with the `vzctl set` command. The capabilities are divided into two tables: the capabilities defined by the POSIX draft standard and Linux-specific capabilities. For each capability, its description is given together with the default value for a container.

Please note that it is easy to create a non-working container or compromise your hardware node security by setting capabilities incorrectly. Do not change any capability for a container without a full understanding of what this capability can lead to.

7.2.1.1. Capabilities Defined by POSIX Draft

Name	Description	Default
`chown`	If a process has this capability set on, it can change ownership on the files not belonging to it or belonging to another user. You have to set this capability on to allow the container root user to change ownership on files and directories inside the container.	on
`dac_override`	This capability allows to access files even if the permission is set to disable access. Normally leave this on to let the container root access files even if the permission does not allow it.	on
`dac_read_search`	Overrides restrictions on reading and searching for files and directories. The explanation is almost the same as above with the sole exclusion that this capability does not override executable restrictions.	on
`fowner`	Overrides restrictions on setting the `S_ISUID` and `S_ISGID` bits on a file requiring that the effective user ID and effective group ID of the process shall match the file owner ID.	on
`fsetid`	Used to decide between falling back on the old `suser()` or `fsuser()`.	on

Name	Description	Default
kill	Allows sending signals to processes owned by other users.	on
setgid	Allows group ID manipulation and forged group IDs on socket credentials passing.	on
setuid	Allows user ID manipulation and forged user IDs on socket credentials passing.	on

7.2.1.2. Linux-specific Capabilities

Name	Description	Default
setpcap	Transfer any capability in your permitted set to any process ID; remove any capability in your permitted set from any process ID.	off
linux_immutable	Allows the modification of the S_IMMUTABLE and S_APPEND file attributes. These attributes are implemented only for the EXT2FS and EXT3FS Linux file systems. However, if you bind mount a directory located on the EXT2FS or EXT3FS file system into a container and revoke this capability, the root user inside the container will not be able to delete or truncate files with these attributes on.	on
net_bind_service	Allows to bind to sockets with numbers below 1024.	on
net_broadcast	Allows network broadcasting and multicast access.	on
net_admin	Allows the administration of IP firewalls and accounting.	off
net_raw	Allows to use the RAW and PACKET sockets.	on
ipc_lock	Allows to lock shared memory segments and mlock/mlockall calls.	on
ipc_owner	Overrides IPC ownership checks.	on
sys_module	Insert and remove kernel modules. Be very careful with setting this capability on for a container; if a user has the permission of inserting kernel modules, this user has essentially full control over the hardware node.	off
sys_chroot	Allows to use chroot().	on
sys_ptrace	Allows to trace any process.	on
sys_pacct	Allows to configure process accounting.	on
sys_admin	In charge of many system administrator tasks such as swapping, administering APM BIOS, and so on. Shall be set to off for containers.	off
sys_boot	This capability currently has no effect on the container behaviour.	on
sys_nice	Allows to raise priority and to set priority for other processes.	on
sys_resource	Override resource limits (not to be confused with user beancounters).	on
sys_time	Allows to change the system time.	off

Name	Description	Default
sys_tty_config	Allows to configure TTY devices.	on
mknod	Allows the privileged aspects of mknod().	on
lease	Allows to take leases of files.	on

7.3. 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.

OpenVZ 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.

7.3.1. 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
```

7.3.1.1. Disabling 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 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 OpenVZ 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.

7.3.2. 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 Section 7.3.1, “Using Golden Image Functionality” on page 104.

7.3.2.1. 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.

7.3.3. 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.

7.4. Enabling VNC Access to Virtual Machines and Containers

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. Enable VNC access in the desired virtual machine or container.
2. Connect to the virtual machine or container with a VNC client.

The sections below describe both steps in details.

7.4.1. 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.

7.4.2. 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.

7.4.3. 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.

7.5. Managing iptables Modules

This section describes how to manage `iptables` modules for both physical servers and containers.

7.5.1. Using iptables Modules in OpenVZ

Filtering network packets on hardware nodes running OpenVZ 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 OpenVZ kernel.

When you enable connection tracking for virtual machines and containers (e.g., for NAT), consider disabling it for the hardware node itself. This way the node will still be reachable in case of a DoS attack. To disable connection tracking for the hardware node itself:

1. specify `options nf_conntrack ip_conntrack_disable_ve0=1` in the file `/etc/modprobe.d/vz.conf` or `/etc/modprobe.d/openvz.conf`,
2. reload the `nf_conntrack` module or restart the hardware node.

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.

7.5.2. Using iptables Modules in Containers

Using `iptables` modules in containers requires additional configuration on your part.

7.5.2.1. 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 `stateless`, which allows all modules except `conntrack` and NAT-related.

7.5.2.2. Using conntrack Rules and NAT Tables

By default, the NAT table and `conntrack` rules are disabled and not allowed for use in containers even if they are loaded on the server. To allow their use in containers, run the `prlctl set --netfilter full` command. For example, for the container `MyCT`:

```
# prlctl set MyCT --netfilter full
```

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`.

7.6. 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 OpenVZ 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:

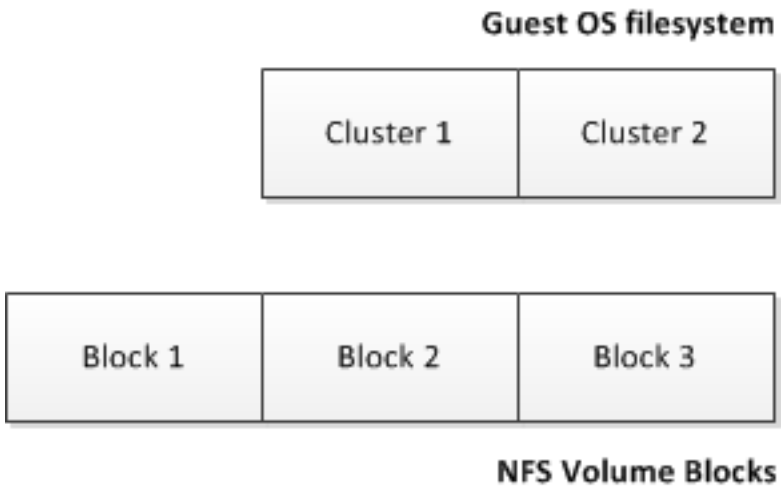
   ```
   # cp fedora.conf centos-7.config
   ```

   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.

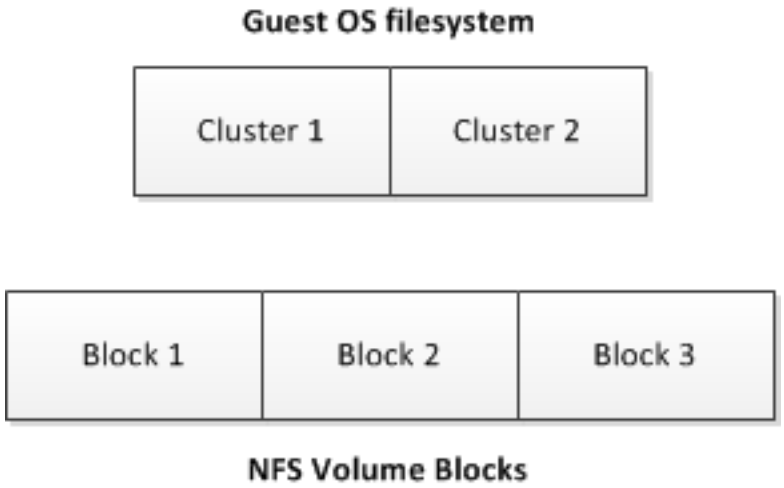
7.7. Aligning Disks and Partitions in Virtual Machines

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.

7.7.1. 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.

7.7.2. 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.

7.7.2.1. 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

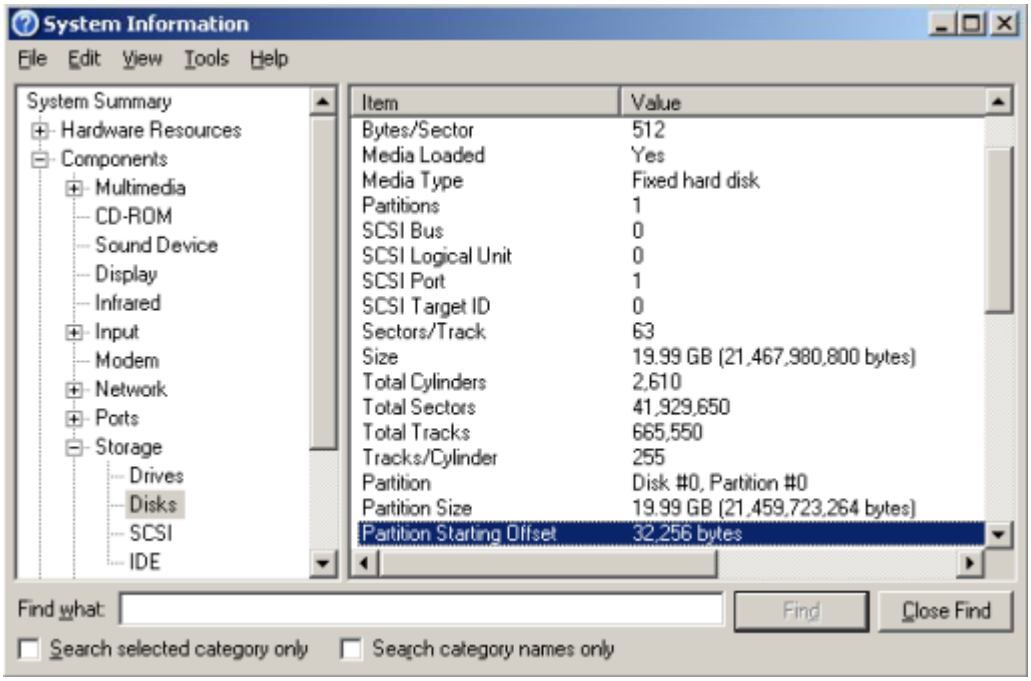
- the `/dev/sdc1` partition by 1 sector so that it starts at 64. In this case, 64 sectors each containing 512 bytes make up 32768 that is a multiple of 4096.
- the `/dev/sdc2` partition by 3 sectors so that it starts at 208848. In this case, 208848 sectors each containing 512 bytes make up 106930176 that is a multiple of 4096.

7.7.2.2. 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.



To find out if the partition is aligned properly, use the method described above for Linux virtual machines.

7.7.3. 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.

7.7.4. 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.

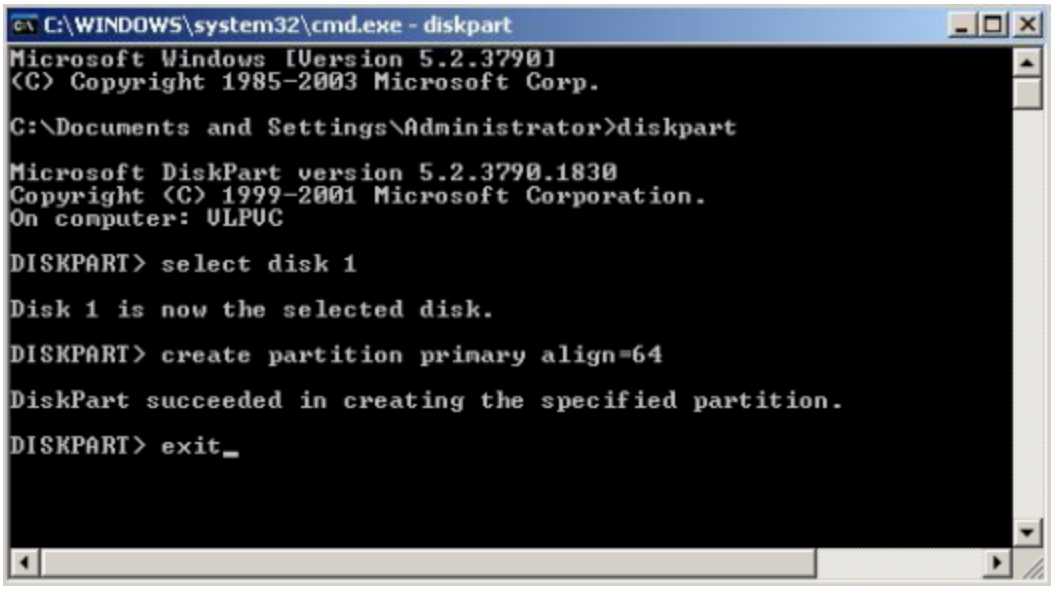
2. Open the command-line prompt, and run the `diskpart` or `diskpar` utility.

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:



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.

7.7.5. 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 Section 7.7.4, “Aligning Partitions for Windows Virtual Machines” on page 114, 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.

7.8. Installing Optional OpenVZ Packages

OpenVZ comes with everything you may need already installed. However, you can also install optional OpenVZ packages from remote repositories by means of the `yum` command.

Note: For more information on using `yum` in OpenVZ, see Section 6.1, “Updating OpenVZ” on page 97 and the `yum` manual page.

7.9. Integrating OpenVZ with OpenStack

OpenVZ supports OpenStack as a cloud management solution. You can install OpenStack with OpenVZ nodes with the help of Devstack tools. For instructions on how to integrate OpenVZ with OpenStack, see https://openvz.org/Setup_OpenStack_with_OpenVZ_7.

Chapter 8. Troubleshooting

This chapter provides the information about those problems that may occur during your work with OpenVZ and suggests the ways to solve them.

8.1. 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 OpenVZ (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

8.2. Kernel Troubleshooting

8.2.1. 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.

8.2.2. Saving Kernel Faults (OOPS)

You can use the following command to check for the kernel messages that should be reported to OpenVZ 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 [1] 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 [2] __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.

8.2.3. 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.

8.3. Container Management Issues

This section includes recommendations on how to solve certain container issues.

8.3.1. Failure to Start a Container

An attempt to start a container fails.

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.

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.

Solution 3

The container might have used all its disk quota (disk space). Check the quota (see Section 3.2, "Managing Disk Quotas" on page 51) and adjust its parameters if needed.

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.

8.3.2. Failure to Access a Container from Network

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
```

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.

8.3.3. Failure to Log In to a Container

The container starts successfully, but you cannot log in.

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
```

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
```