SWsoft, Inc.

OpenVZ

User's Guide

Version 2.7.0-8

SWsoft

ISBN: N/A
SWsoft Inc
13755 Sunrise Valley Drive
Suite 325
Herndon, VA 20171
USA
Tel: +1 (703) 815 5670
Fax: +1 (703) 815 5675

Contents

Table of Figures

CHAPTER 1

Preface

In This Chapter

About This Guide

This guide is meant to provide comprehensive information on OpenVZ– high-end server virtualization software for Linux-based computers. The issues discussed in this guide cover the necessary theoretical conceptions as well as practical aspects of working with OpenVZ. The guide will familiarize you with the way to create and administer *Virtual Private Servers* (sometimes also called *Virtual Environments*, or *VEs*) on OpenVZ-based Hardware Nodes and to employ the command line interface for performing various tasks.

Familiarity with Red Hat Linux Operating System and certain Linux administrator's skills are desirable for a person reading the guide. You can obtain some useful information regarding OS installation issues from http://www.redhat.com/docs/manuals/linux/.

Who Should Read This Guide

The primary audience for this book is anyone responsible for administering one or more systems running OpenVZ. To fully understand the guide, you should have strong Linux system administration habits. Attending Linux system administration training courses might be helpful. Still, no more than superficial knowledge of Linux OS is required in order to comprehend the major OpenVZ notions and learn to perform the basic administrative operations.

Organization of This Guide

Chapter 2, OpenVZ Philosophy, is a must-read chapter that helps you grasp the general principles of OpenVZ operation. It provides an outline of OpenVZ architecture, of the way OpenVZ stores and uses configuration information, of the things you as administrator are supposed to perform, and the common way to perform them.

Chapter 3, Installation and Preliminary Operations, dwells on all those things that must be done before you are able to begin the administration proper of OpenVZ. Among these things are a customized installation of Linux on a dedicated computer (Hardware Node, in OpenVZ terminology), OpenVZ installation, preparation of the Hardware Node for creating Virtual Private Servers on it, etc.

Chapter 4, Operations on Virtual Private Servers, covers those operations that you may perform on a VPS as on a single entity: creating and deleting Virtual Private Servers, starting and stopping them, etc.

Chapter 5, Managing Templates, shows you the way to handle OpenVZ templates properly – create and install templates and template updates on the Hardware Node, add them to and remove from Virtual Private Servers, etc.

Chapter 6, Managing Resources, zeroes in on configuring and monitoring the resource control parameters for different VPSs. These parameters comprise disk quotas, CPU and system resources. Common ways of optimizing your VPSs configurations are suggested at the end of the chapter.

Chapter 7, Advanced Tasks, enumerates those tasks that are intended for advanced system administrators who would like to obtain deeper knowledge about OpenVZ capabilities.

Chapter 8, Troubleshooting, suggests ways to resolve common inconveniences should they occur during your work with the OpenVZ software.

Chapter 9, Reference, is a complete reference on all OpenVZ configuration files and Hardware Node command-line utilities. You should read this chapter if you do not understand a file format or looking for an explanation of a particular configuration option, if you need help for a particular command or looking for a command to perform a certain task.

Documentation Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it. For information on specialized terms used in the documentation, see the Glossary at the end of this document.

Typographical Conventions

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the **QoS** tab.
	Titles of chapters, sections, and subsections.	Read the **Basic Administration** chapter.
Italics	Used to emphasize the importance of a point or to introduce a term.	Such servers are called *Hardware Nodes*.
`Monospace`	The names of commands, files, and directories.	Use `vzctl start` to start a VPS.
`Monospace Italics`	Used to designate a command line or a file name placeholder, which is to be replaced with a real value.	Type `vzctl destroy` *`vpsid`*.
`Preformatted`	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	`Saved parameters for VPS 101`
`Monospace Bold`	What you type, contrasted with on-screen computer output.	**`# rpm -q vzctl`**
CAPITALS	Names of keys on the keyboard.	SHIFT, CTRL, ALT
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another.	CTRL+P, ALT+F4

Shell Prompts in Command Examples

Command line examples throughout this guide presume that you are using the Bourne-again shell (bash). Whenever a command can be run as a regular user, we will display it with a dollar sign prompt. When a command is meant to be run as root, we will display it with a hash mark prompt:

Bourne-again shell prompt	$
Bourne-again shell root prompt	#

General Conventions

Be aware of the following conventions used in this book.

- Chapters in this guide are divided into sections, which, in turn, are subdivided into subsections. For example, Documentation Conventions is a section, and General Conventions is a subsection.
- When following steps or using examples, be sure to type double-quotes (`"`), left single-quotes (`` ` ``), and right single-quotes (`'`) exactly as shown.
- The key referred to as RETURN is labeled ENTER on some keyboards.

The root path usually includes the `/bin`, `/sbin`, `/usr/bin` and `/usr/sbin` directories, so the steps in this book show the commands in these directories without absolute path names. Steps that use commands in other, less common, directories show the absolute paths in the examples.

Feedback

If you spot a typo in this guide, or if you have thought of a way to make this guide better, we would love to hear from you!

If you have a suggestion for improving the documentation (or any other relevant comments), try to be as specific as possible when formulating it. If you have found an error, please include the chapter/section/subsection name and some of the surrounding text so we can find it easily.

Please submit a report by e-mail to userdocs@openvz.org.

CHAPTER 2

OpenVZ Philosophy

In This Chapter

About OpenVZ Software

What is OpenVZ

OpenVZ is a complete server automation and virtualization solution developed by SWsoft. OpenVZ creates multiple isolated Virtual Private Servers (VPSs) on a single physical server to share hardware and management effort with maximum efficiency. Each VPS performs and executes exactly like a stand-alone server for its users and applications as it can be rebooted independently and has its own `root` access, users, IP addresses, memory, processes, files, applications, system libraries, and configuration files. Light overhead and efficient design of OpenVZ makes it the right virtualization choice for production servers with live applications and real-life data.

The basic OpenVZ VPS capabilities are:

- *Dynamic Real-time Partitioning* – Partition a physical server into tens of VPSs, each with full dedicated server functionality.
- *Resource Management* – Assign and control VPS resource parameters and re-allocate resources in real-time.
- *Mass Management* - Manage a multitude of physical servers and Virtual Private Servers in a unified way.

OpenVZ Applications

OpenVZ provides a comprehensive solution for Hosting Service Providers allowing them to:

- Have hundreds of customers with their individual full-featured virtual private servers (Virtual Private Servers) sharing a single physical server;
- Provide each customer with a guaranteed Quality of Service;
- Transparently move customers and their environments between servers, without any manual reconfiguration.

If you administer a number of Linux dedicated servers within an enterprise, each of which runs a specific service, you can use OpenVZ to consolidate all these servers onto a single computer without losing a bit of valuable information and without compromising performance. Virtual Private Servers behave just like an isolated stand-alone server:

- Each VPS has its own processes, users, files and provides full root shell access;
- Each VPS has its own IP addresses, port numbers, filtering and routing rules;
- Each VPS can have its own configuration for the system and application software, as well as its own versions of system libraries. It is possible to install or customize software packages inside a VPS independently from other VPSs or the host system. Multiple distributions of a package can be run on one and the same Linux box.

In fact, hundreds of servers may be grouped together in this way. Besides the evident advantages of such consolidation (increased facility of administration and the like), there are some you might not even have thought of, say, cutting down electricity bills by times!

OpenVZ proves invaluable for IT educational institutions that can now provide every student with a personal Linux server, which can be monitored and managed remotely. Software development companies may use virtual environments for testing purposes and the like.

Thus, OpenVZ can be efficiently applied in a wide range of areas: web hosting, enterprise server consolidation, software development and testing, user training, and so on.

Distinctive Features of OpenVZ

The concept of OpenVZ Virtual Private Servers is distinct from the concept of traditional virtual machines in the respect that Virtual Private Servers (VPSs) always run the same OS kernel as the host system (Linux on Linux, Windows on Windows, etc.). This single-kernel implementation technology allows to run Virtual Private Servers with a near-zero overhead. Thus, OpenVZ VPSs offer an order of magnitude higher efficiency and manageability than traditional virtualization technologies.

OS Virtualization

From the point of view of applications and Virtual Private Server users, each VPS is an independent system. This independency is provided by a virtualization layer in the kernel of the host OS. Note that only an infinitesimal part of the CPU resources is spent on virtualization (around 1-2%). The main features of the virtualization layer implemented in OpenVZ are the following:

- VPS looks like a normal Linux system. It has standard startup scripts, software from vendors can run inside VPS without OpenVZ-specific modifications or adjustment;
- A user can change any configuration file and install additional software;
- Virtual Private Servers are fully isolated from each other (file system, processes, Inter Process Communication (IPC), `sysctl` variables);
- Processes belonging to a VPS are scheduled for execution on all available CPUs. Consequently, VPSs are not bound to only one CPU and can use all available CPU power.

Network Virtualization

The OpenVZ network virtualization layer is designed to isolate VPSs from each other and from the physical network:

- Each VPS has its own IP address; multiple IP addresses per VPS are allowed;
- Network traffic of a VPS is isolated from the other VPSs. In other words, Virtual Private Servers are protected from each other in the way that makes traffic snooping impossible;
- Firewalling may be used inside a VPS (the user can create rules limiting access to some services using the canonical `iptables` tool inside the VPS). In other words, it is possible to set up firewall rules from inside a VPS;
- Routing table manipulations are allowed to benefit from advanced routing features. For example, setting different maximum transmission units (MTUs) for different destinations, specifying different source addresses for different destinations, and so on.

Templates

An OS template in OpenVZ is basically a set of packages from some Linux distribution used to populate one or more VPSs. With OpenVZ, different distributions can co-exist on the same hardware box, so multiple OS templates are available. An OS template consists of system programs, libraries, and scripts needed to boot up and run the system (VPS), as well as some very basic applications and utilities. Applications like a compiler and an SQL server are usually not included into an OS template.

For detailed information on OpenVZ templates, see the Understanding Templates section.

Resource Management

OpenVZ Resource Management controls the amount of resources available to Virtual Private Servers. The controlled resources include such parameters as CPU power, disk space, a set of memory-related parameters. Resource management allows OpenVZ to:

- Effectively share available Hardware Node resources among VPSs;
- Guarantee Quality-of-Service (QoS) 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 Private Servers, etc.

Resource Management is much more important for OpenVZ than for a standalone computer since computer resource utilization in an OpenVZ-based system is considerably higher than that in a typical system.

Main Principles of OpenVZ Operation

Basics of OpenVZ Technology

In this section we will try to let you form a more or less precise idea of the way the OpenVZ software operates on your computer. Please see the figure below:

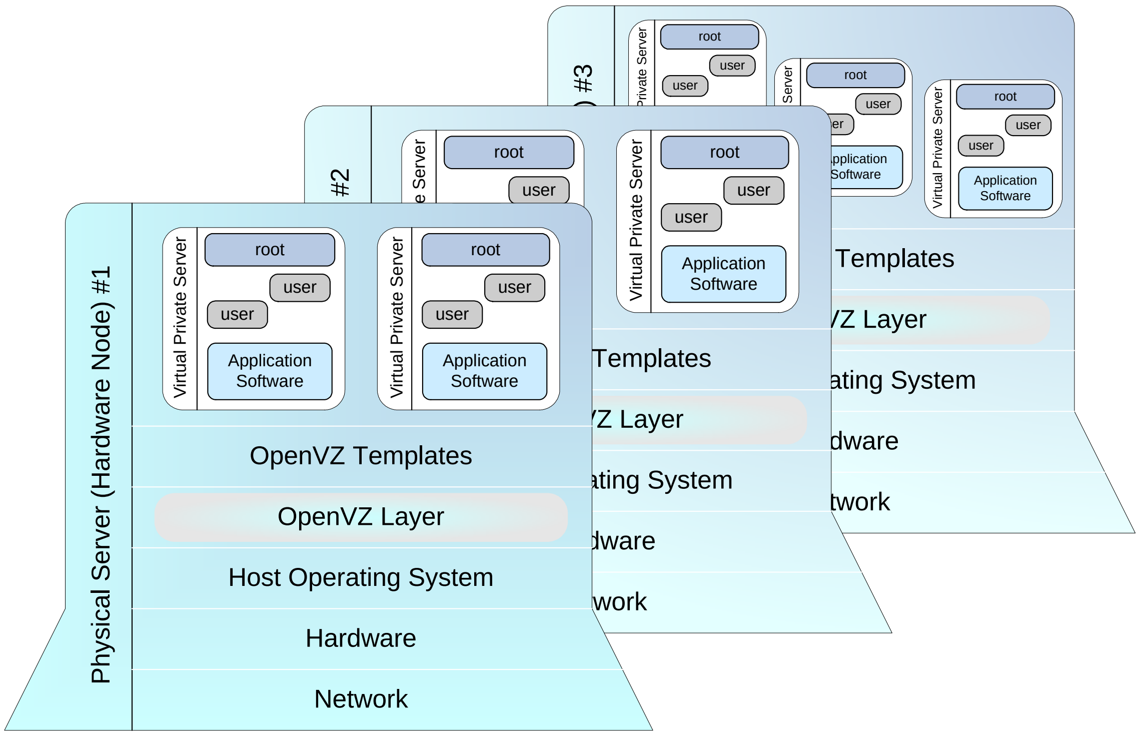


Figure 1: OpenVZ Technology

This figure presumes that you have a number of physical servers united into a network. In fact, you may have only one dedicated server to effectively use OpenVZ for the needs of your network. If you have more than one OpenVZ-based physical server, each one of the servers will have a similar architecture. In OpenVZ terminology, such servers are called *Hardware Nodes* (or *HN*, or just *Nodes*), because they represent hardware units within a network.

OpenVZ is installed on Fedora Core 3 or 4 or Red Hat Enterprise Linux 4 configured in a certain way. For example, such customized configuration shall include the creation of a `/vz` partition, which is the basic partition for hosting Virtual Private Servers and which must be way larger than the root partition. This and similar configuration issues are most easily resolved during Linux installation on the Hardware Node. Detailed instructions on installing Linux (called *Host Operating System*, or *Root Operating System* in Figure 1) on the Hardware Node are provided in the next chapter.

OpenVZ is installed in such a way that you will be able to boot your computer either with OpenVZ support or without it. This support is presented as “OpenVZ” in your boot loader and shown as *OpenVZ Layer* in the figure above.

However, at this point you are not yet able to create Virtual Private Servers. A *Virtual Private Server* is functionally identical to an isolated standalone server, having its own IP addresses, processes, files, users, its own configuration files, its own applications, system libraries, and so on. Virtual private servers share the same *Hardware Node* and the same OS kernel. However, they are isolated from each other. A Virtual Private Server is a kind of ‘sandbox’ for processes and users.

Different Virtual Private Servers can run different versions of Linux (for example, SuSE 9.2 or Fedora Core 4 and many others). Each VPS can run its own version of Linux. In this case we say that a VPS is based on a certain OS template. OS templates are packages shipped with OpenVZ. Before you are able to create a Virtual Private Server, you should install the corresponding OS template in OpenVZ. This is displayed as *OpenVZ Templates* in the scheme above.

After you have installed at least one OS template, you can create any number of VPSs with the help of standard OpenVZ utilities, configure their network and/or other settings, and work with these VPSs as with fully functional Linux servers.

Understanding Templates

A template is a VPS building block. An OS template is a set of packages needed to operate a VPS. Templates are usually created right on your Hardware Node; all you need is template tools (`vzpkg`) and template metadata.

Template metadata

Template metadata are information *about* a particular OS template. It contains:

- a list of packages included in this template (in the form of package names);
- location of (network) package repositories;
- distribution-specific scripts needed to be executed on various stages of template installation;
- public GPG key(s) needed to check signatures of packages;

All this information is contained in a few files installed into the `/vz/template/`*`osname`*`/`*`osrelease`*`/config/` directory. For example, the metadata for the Fedora Core 4 template are installed into the `/vz/template/fedora-core/4/config/` directory.

Along with template metadata, a few OpenVZ-specific packages are usually provided; they are installed into the `/vz/template/`*`osname`*`/`*`osversion`*`/vz-addons/` directory.

Template cache

Template metadata provide enough information to create an OS template. During the OS template creation, the needed package files are downloaded from the network repository to the Hardware Node and installed into a temporary VPS, which is then packed into a gzipped tarball called the *template cache*.

The template cache is used for fast VPS provisioning – basically, it is a pre-created VPS, so all that is needed to create a VPS is to untar this file. The template cache files are stored in the `/vz/template/cache/` directory.

Any template cache becomes obsolete with time as new updates are released for the given distribution. Naturally, there is a way to quickly update the template cache as well as all the previously created VPSs with the newest updates.

While you are able to perform all kinds of tasks within a Virtual Private Server including building `rpm` packages and installing them, OpenVZ provides an easy and far more efficient way of installing the applications you need on VPSs. The same way as you install an OS template on the OpenVZ system in order to create any number of Virtual Private Servers on its basis and share its resources, you can install applications in OpenVZ in order to share package files among any number of VPSs. You can then add these applications to any number of Virtual Private Servers.

It goes without saying that in case you want to install an application on only one VPS, there is no need in working with templates: you can as well work inside the corresponding VPS.

Understanding Licenses

The OpenVZ software consists of the OpenVZ kernel and user-level tools, which are licensed by means of two different open source licenses.

- The OpenVZ kernel is based on the Linux kernel, distributed under the GPL terms, and is licensed under GNU GPL version 2. The license text can be found at http://openvz.org/documentation/licenses/gnu-gpl.
- The user-level tools (`vzctl`, `vzquota`, and `vzpkg`) are licensed under the terms of the QPL license. The license text can be found at http://openvz.org/documentation/licenses/qpl.

OpenVZ Configuration

OpenVZ allows you to flexibly configure various settings for the OpenVZ system in general as well as for each and every Virtual Private Server. Among these settings are disk and user quota, network parameters, default file locations and configuration sample files, and others.

OpenVZ stores the configuration information in two types of files: the global configuration file `/etc/sysconfig/vz` and VPS configuration files `/etc/sysconfig/vz-scripts/`*`vpsid`*`.conf`. The global configuration file defines global and default parameters for VPS operation, for example, logging settings, enabling and disabling disk quota for VPSs, the default configuration file and OS template on the basis of which a new VPS is created, and so on. On the other hand, a VPS configuration file defines the parameters for a given particular VPS, such as disk quota and allocated resources limits, IP address and host name, and so on. In case a parameter is configured both in the global OpenVZ configuration file, and in the VPS configuration file, the VPS configuration file takes precedence. For a list of parameters constituting the global configuration file and the VPS configuration files, turn to the Reference chapter.

The configuration files are read when OpenVZ and/or VPSs are started. However, OpenVZ standard utilities, for example, `vzctl`, allow you to change many configuration settings "on-the-fly", either without modifying the corresponding configuration files or with their modification (if you want the changes to apply the next time OpenVZ and/or VPSs are started).

Hardware Node Availability Considerations

Hardware Node availability is more critical than the availability of a typical PC server. Since it runs multiple Virtual Private Servers providing a number of critical services, Hardware Node outage might be very costly. Hardware Node outage can be as disastrous as the simultaneous outage of a number of servers running critical services.

In order to increase Hardware Node availability, we suggest you follow the recommendations below:

- Use RAID storage for critical VPS private areas. Do prefer hardware RAID, but software mirroring RAID might suit too as a last resort.
- Do not run software on the Hardware Node itself. Create special Virtual Private Servers where you can host necessary services such as BIND, FTPD, HTTPD, and so on. On the Hardware Node itself, 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 Hardware Node itself. You can create as many users as you need in any Virtual Private Server. Remember, compromising the Hardware Node means compromising all Virtual Private Servers as well.

CHAPTER 3

Installation and Preliminary Operations

The current chapter provides exhaustive information on the process of installing and deploying your OpenVZ system including the pre-requisites and the stages you shall pass.

In This Chapter

Installation Requirements

After deciding on the structure of your OpenVZ system, you should make sure that all the Hardware Nodes where you are going to deploy OpenVZ for Linux meet the following system (hardware and software) and network requirements.

System Requirements

This section focuses on the hardware and software requirements for the OpenVZ for Linux software product.

Hardware Compatibility

The Hardware Node requirements for the standard 32-bit edition of OpenVZ are the following:

- IBM PC-compatible computer;
- Intel Celeron, Pentium II, Pentium III, Pentium 4, Xeon, or AMD Athlon CPU;
- At least 128 MB of RAM;
- Hard drive(s) with at least 4 GB of free disk space;
- Network card (either Intel EtherExpress100 (i82557-, i82558- or i82559-based) or 3Com (3c905 or 3c905B or 3c595) or RTL8139-based are recommended).

The computer should satisfy the Red Hat Enterprise Linux or Fedora Core hardware requirements (please, see the hardware compatibility lists at www.redhat.com).

The exact computer configuration depends on how many Virtual Private Servers you are going to run on the computer and what load these VPSs are going to produce. Thus, in order to choose the right configuration, please follow the recommendations below:

- CPUs. The more Virtual Private Servers you plan to run simultaneously, the more CPUs you need.
- Memory. The more memory you have, the more Virtual Private Servers you can run. The exact figure depends on the number and nature of applications you are planning to run in your Virtual Private Servers. However, on the average, at least 1 GB of RAM is recommended for every 20-30 Virtual Private Servers;
- Disk space. Each Virtual Private Server occupies 400–600 MB of hard disk space for system files in addition to the user data inside the Virtual Private Server (for example, web site content). You should consider it when planning disk partitioning and the number of Virtual Private Servers to run.

A typical 2–way Dell PowerEdge 1650 1u–mountable server with 1 GB of RAM and 36 GB of hard drives is suitable for hosting 30 Virtual Private Servers.

Software Compatibility

The Hardware Node should run either Red Hat Enterprise Linux 3 or 4, or Fedora Core 3 or 4, or CentOS 3.4 or 4. The detailed instructions on installing these operating systems for the best performance of OpenVZ are provided in the next sections.

This requirement does not restrict the ability of OpenVZ to provide other Linux versions as an operating system for Virtual Private Servers. The Linux distribution installed in a Virtual Private Server may differ from that of the host OS.

Network Requirements

The network pre-requisites enlisted in this subsection will help you avoid delays and problems with making OpenVZ for Linux up and running. You should take care in advance of the following:

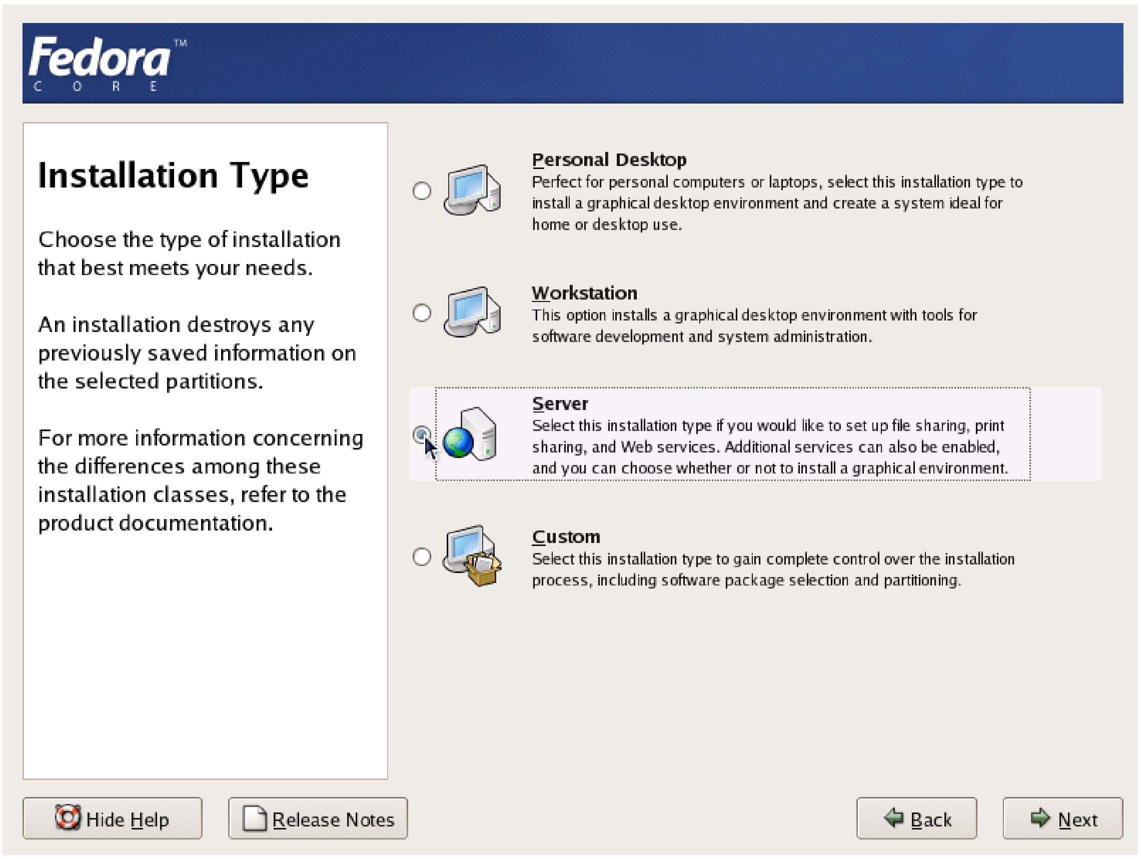
- Local Area Network (LAN) for the Hardware Node;
- Internet connection for the Hardware Node;
- Valid IP address for the Hardware Node as well as other IP parameters (default gateway, network mask, DNS configuration);
- At least one valid IP address for each Virtual Private Server. The total number of addresses should be no less than the planned number of Virtual Private Servers. The addresses may be allocated in different IP networks;
- If a firewall is deployed, check that IP addresses allocated for Virtual Private Servers are open for access from the outside.

Installing and Configuring Host Operating System on Hardware Node

This section explains how to install Fedora Core 4 on the Hardware Node and how to configure it for OpenVZ. If you are using another distribution, please consult the corresponding installation guides about the installation specifics.

Choosing System Type

Please follow the instructions from your Installation Guide when installing the OS on your Hardware Node. After the first several screens, you will be presented with a screen specifying the installation type. OpenVZ requires Server System to be installed, therefore select “Server” at the dialog shown in the figure below.



Installation Type

Choose the type of installation that best meets your needs.

An installation destroys any previously saved information on the selected partitions.

For more information concerning the differences among these installation classes, refer to the product documentation.

Personal Desktop
Perfect for personal computers or laptops, select this installation type to install a graphical desktop environment and create a system ideal for home or desktop use.

Workstation
This option installs a graphical desktop environment with tools for software development and system administration.

Server
Select this installation type if you would like to set up file sharing, print sharing, and Web services. Additional services can also be enabled, and you can choose whether or not to install a graphical environment.

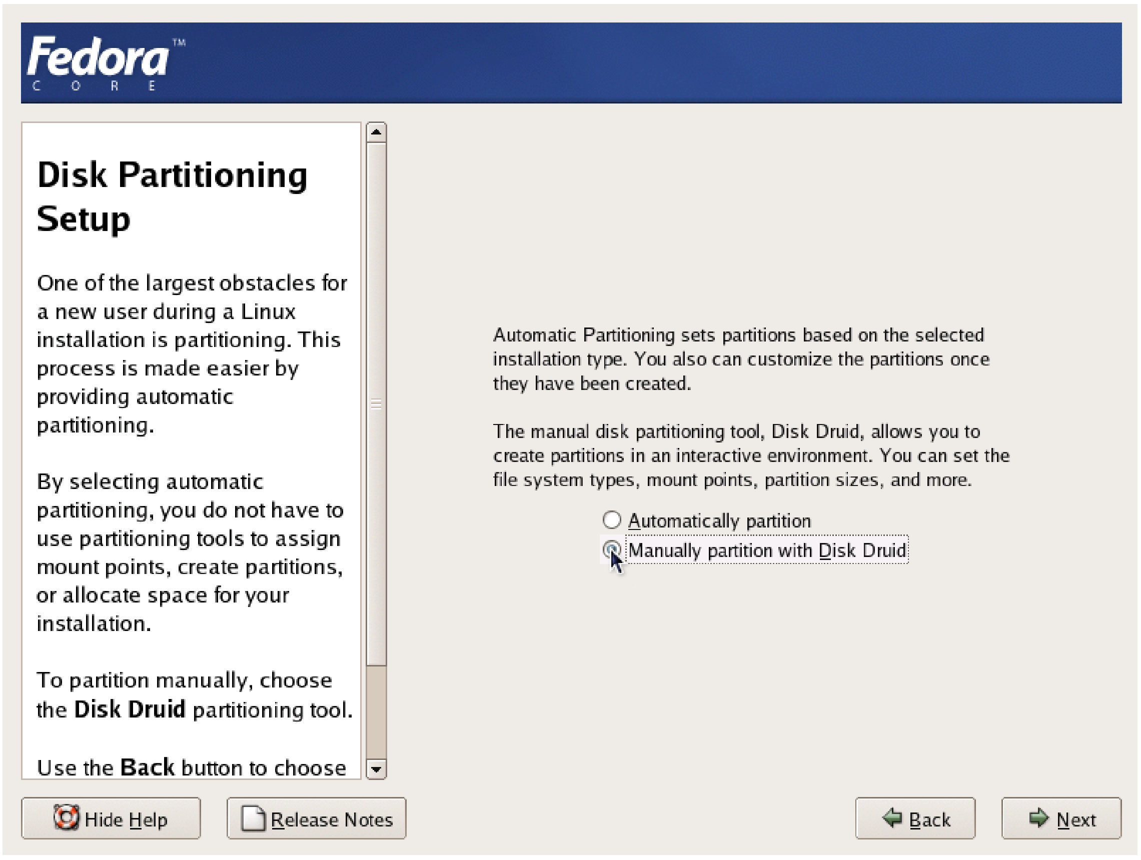
Custom
Select this installation type to gain complete control over the installation process, including software package selection and partitioning.

Figure 2: Fedora Core Installation - Choosing System Type

It is not recommended to install extra packages on the Hardware Node itself due to the all-importance of Hardware Node availability (see the Hardware Node Availability Considerations subsection in this chapter). You will be able to run any necessary services inside dedicated Virtual Private Servers.

Disk Partitioning

On the Disk Partitioning Setup screen, select Manual partition with Disk Druid. Do not choose automatic partitioning since this type of partitioning will create a disk layout intended for systems running multiple services. In case of OpenVZ, all your services shall run inside Virtual Private Servers.



Disk Partitioning Setup

One of the largest obstacles for a new user during a Linux installation is partitioning. This process is made easier by providing automatic partitioning.

By selecting automatic partitioning, you do not have to use partitioning tools to assign mount points, create partitions, or allocate space for your installation.

To partition manually, choose the **Disk Druid** partitioning tool.

Use the **Back** button to choose

Automatic Partitioning sets partitions based on the selected installation type. You also can customize the partitions once they have been created.

The manual disk partitioning tool, Disk Druid, allows you to create partitions in an interactive environment. You can set the file system types, mount points, partition sizes, and more.

Automatically partition [✓ Manually partition with Disk Druid]

Figure 3: Fedora Core Installation - Choosing Manual Partitioning

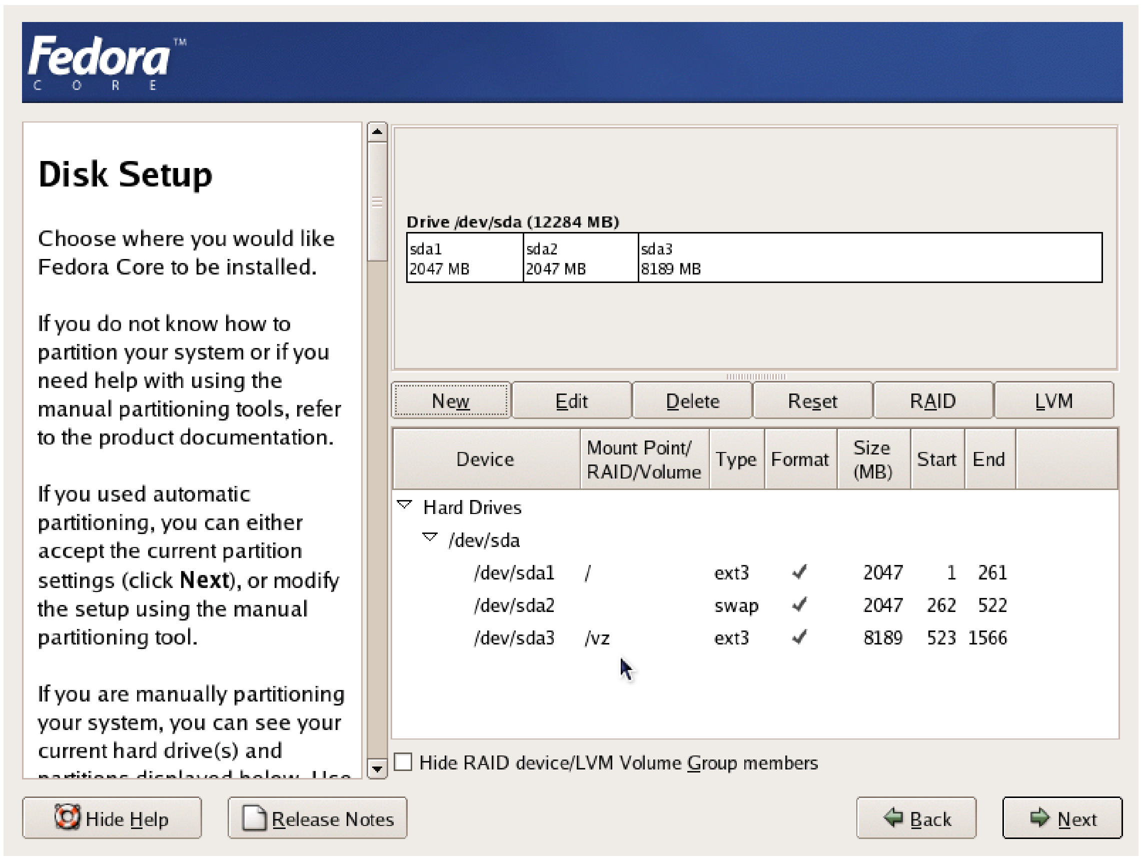
Create the following partitions on the Hardware Node:

Partition	Description	Typical size
`/`	Root partition containing all Hardware Node operating system files	2-4 Gb
`swap`	Paging partition for the Linux operating system	2 times RAM
`/vz`	Partition to host OpenVZ templates and Virtual Private Servers	all the remaining space on the hard disk

It is suggested to use the `ext3` file system for the `/vz` partition. This partition is used for holding all data of the Virtual Private Servers existing on the Hardware Node. Allocate as much disk space as possible to this partition. It is not recommended to use the `reiserfs` file system as it is proved to be less stable than the `ext3`, and stability is of paramount importance for OpenVZ-based computers.

The root partition will host the operating system files. The server set of Fedora Core 4 occupies approximately 1 GB of disk space, so 1 GB is the minimal size of the root partition. The size of the swap partition shall be two times the size of physical RAM installed on the Hardware Node.

The figure below presents a system with a 12 GB SCSI hard drive.



Disk Setup

Choose where you would like Fedora Core to be installed.

If you do not know how to partition your system or if you need help with using the manual partitioning tools, refer to the product documentation.

If you used automatic partitioning, you can either accept the current partition settings (click **Next**), or modify the setup using the manual partitioning tool.

If you are manually partitioning your system, you can see your current hard drive(s) and

Drive /dev/sda (12284 MB)

sda1	sda2	sda3
2047 MB	2047 MB	8189 MB

Device	Mount Point/ RAID/Volume	Type	Format	Size (MB)	Start	End
Hard Drives						
/dev/sda						
/dev/sda1	/	ext3	✓	2047	1	261
/dev/sda2		swap	✓	2047	262	522
/dev/sda3	/vz	ext3	✓	8189	523	1566

☐ Hide RAID device/LVM Volume Group members

Figure 4: Fedora Core Installation - Disk Druid

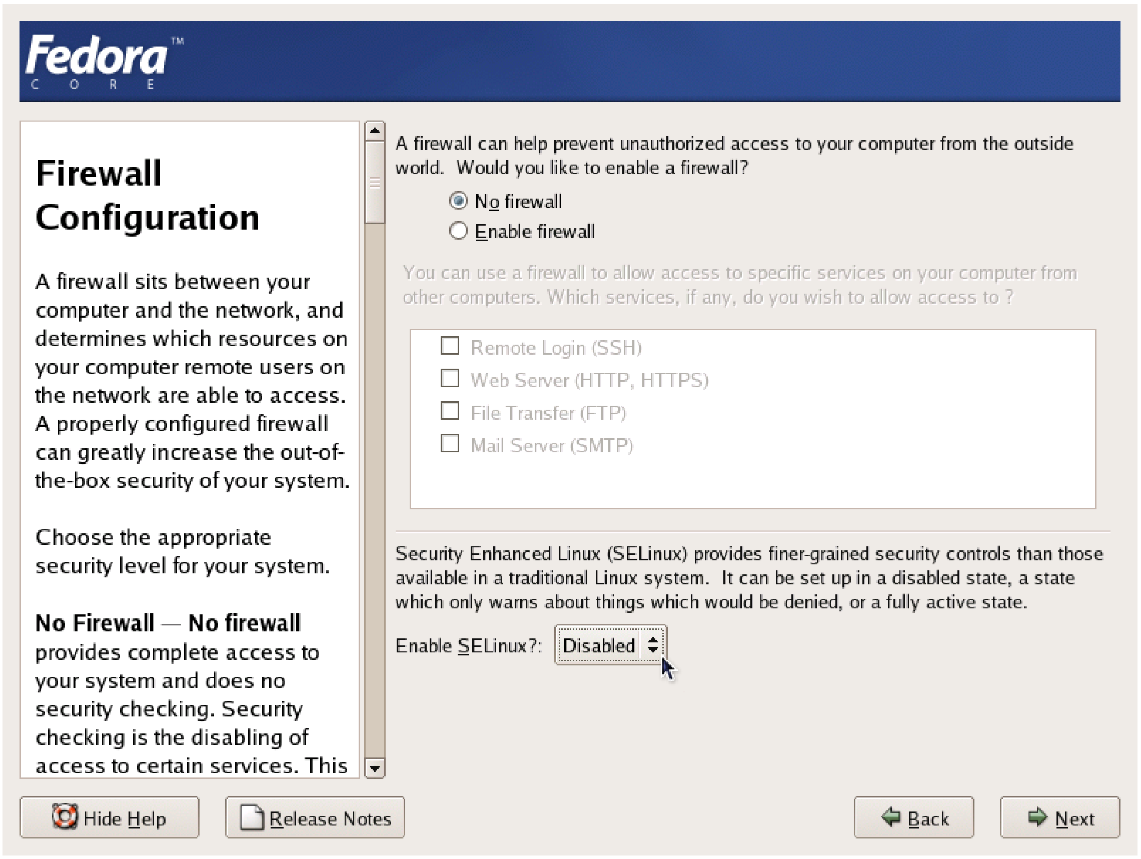
Please keep in mind that Virtual Private Server private areas, containing all data of the Virtual Private Servers shall reside on this single `/vz` disk partition together with all the templates installed.

Finishing OS Installation

After the proper partitioning of your hard drive(s), proceed in accordance with your OS Installation Guide.

While on the **Network Configuration** screen, you should ensure the correctness of the Hardware Node’s IP address, host name, DNS, and default gateway information. If you are using DHCP, make sure that it is properly configured. If necessary, consult your network administrator.

On the **Firewall Configuration** screen, choose **No firewall**. Option **Enable SELinux** should be set to **Disabled**.



Firewall Configuration

A firewall sits between your computer and the network, and determines which resources on your computer remote users on the network are able to access. A properly configured firewall can greatly increase the out-of-the-box security of your system.

Choose the appropriate security level for your system.

No Firewall — **No firewall** provides complete access to your system and does no security checking. Security checking is the disabling of access to certain services. This

A firewall can help prevent unauthorized access to your computer from the outside world. Would you like to enable a firewall?

[✓ No firewall]
Enable firewall

You can use a firewall to allow access to specific services on your computer from other computers. Which services, if any, do you wish to allow access to ?

- ☐ Remote Login (SSH)
- ☐ Web Server (HTTP, HTTPS)
- ☐ File Transfer (FTP)
- ☐ Mail Server (SMTP)

Security Enhanced Linux (SELinux) provides finer-grained security controls than those available in a traditional Linux system. It can be set up in a disabled state, a state which only warns about things which would be denied, or a fully active state.

Enable SELinux?: Disabled

Figure 5: Fedora Core Installation - Disabling Firewall and SELinux

After finishing the installation and rebooting your computer, you are ready to install OpenVZ on your system.

Installing OpenVZ Software

Downloading and Installing OpenVZ Kernel

First of all, you should download the kernel binary RPM from http://openvz.org/download/kernel/. You need only one kernel RPM, so please choose the appropriate kernel binary depending on your hardware:

- If there is more than one CPU available on your Hardware Node (or a CPU with hyperthreading), select the `vzkernel-smp` RPM.
- If there is more than 4 Gb of RAM available, select the `vzkernel-enterprise` RPM.
- Otherwise, select the uniprocessor kernel RPM (`vzkernel-`*`version`*).

Next, you shall install the kernel RPM of your choice on your Hardware Node by issuing the following command:

```
# rpm -ihv vzkernel-name*.rpm
```

Note: You should not use the `rpm –U` command (where `-U` stands for "upgrade"); otherwise, all the kernels currently installed on the Node will be removed.

Configuring Boot Loader

In case you use the GRUB loader, it will be configured automatically. You should only make sure that the lines below are present in the `/boot/grub/grub.conf` file on the Node:

```
title Fedora Core (2.6.8-022stab029.1)
      root (hd0,0)
      kernel /vmlinuz-2.6.8-022stab029.1 ro root=/dev/sda5 quiet rhgb
      initrd /initrd-2.6.8-022stab029.1.img
```

However, we recommend that you configure this file in the following way:

- Change `Fedora Core` to `OpenVZ` (just for clarity, so the OpenVZ kernels will not be mixed up with non OpenVZ ones).
- Remove all extra arguments from the `kernel` line, leaving only the `root=`... parameter.

At the end, the modified `grub.conf` file should look as follows:

```
title OpenVZ (2.6.8-022stab029.1)
      root (hd0,0)
      kernel /vmlinuz-2.6.8-022stab029.1 ro root=/dev/sda5
      initrd /initrd-2.6.8-022stab029.1.img
```

Setting sysctl parameters

There are a number of kernel limits that should be set for OpenVZ to work correctly. OpenVZ is shipped with a tuned `/etc/sysctl.conf` file. Below are the contents of the relevant part of `/etc/sysctl.conf`:

```
# On Hardware Node we generally need
# packet forwarding enabled and proxy arp disabled
net.ipv4.ip_forward = 1
net.ipv4.conf.default.proxy_arp = 0
# Enables source route verification
net.ipv4.conf.all.rp_filter = 1
# Enables the magic-sysrq key
kernel.sysrq = 1
# TCP Explict Congestion Notification
#net.ipv4.tcp_ecn = 0
# we do not want all our interfaces to send redirects
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
```

Please edit the file as described. To apply the changes issue the following command:

```
# sysctl -p
```

Alternatively, the changes will be applied upon the following reboot.

It is also worth mentioning that normally you should have forwarding (`net.ipv4.ip_forward`) turned on since the Hardware Node forwards the packets destined to or originating from the Virtual Private Servers.

After that, you should reboot your computer and choose "OpenVZ" on the boot loader menu.

Downloading and Installing OpenVZ Packages

After you have successfully installed and booted the OpenVZ kernel, you can proceed with installing the user-level tools for OpenVZ.

You should install the following OpenVZ packages:

- `vzctl`: this package is used to perform different tasks on the OpenVZ Virtual Private Servers (create, destroy, start, stop, set parameters etc.).
- `vzquota`: this package is used to manage the VPS quotas.
- `vzpkg`: this package is used to work with OpenVZ templates.

You can download the corresponding binary RPMs from http://openvz.org/download/utils/.

On the next step, you should install these utilities by using the following command:

```
# rpm –Uhv vzctl*.rpm vzquota*.rpm vzpkg*.rpm
```

Note: During the packages installation, you may be presented with a message telling you that `rpm` has found unresolved dependencies. In this case you have to resolve these dependencies first and then repeat the installation.

Now you can launch OpenVZ. To this effect, execute the following command:

```
# /etc/init.d/vz start
```

This will load all the needed OpenVZ kernel modules. During the next reboot, this script will be executed automatically.

Installing OS Templates

Template (or *package set*) is a set of package files to be installed into a VPS. Operating system templates are used to create new Virtual Private Servers with a pre-installed operating system. Therefore, you are bound to download at least one OS template from http://openvz.org/download/template/ and install it.

OS *template metadata* contain the information needed to create a template cache. You have to specify an OS template on the VPS creation, so you need to install the metadata for at least one OS template and prepare the template cache.

For example, this is how the template preparation for Fedora Core 3 will look like:

```
# rpm –ihv vztmpl-fedora-core-3-1.0-2.noarch.rpm
Preparing...        ########################################### [100%]
   1: vztmpl-fedora-########################################### [100%]
# vzpkgcache
Creating cache for fedora-core-3 OS template
Setting up install process
<…some output skipped for clarity…>
Packing cache file fedora-core-3.tar.gz ...
Cache file fedora-core-3.tar.gz [130M] created.
```

The first command installs the template metadata, while the second one creates the template cache. Note that this operation can take a considerable time (tens of minutes).

You can also use one of the already pre-cached OS templates available at http://openvz.org/download/template/cache/ for the VPS creation. To this effect, you should download the corresponding OS template and place it to the `/vz/template/cache` directory on the Node.

C H A P T E R 4

Operations on Virtual Private Servers

This chapter describes how to perform day-to-day operations on separate Virtual Private Servers taken in their wholeness.

Note: We assume that you have successfully installed, configured, and deployed your OpenVZ system. In case you have not, please turn to Chapter 3 providing detailed information on all these operations.

In This Chapter

Creating and Configuring New Virtual Private Server

This section guides you through the process of creating a Virtual Private Server. We assume that you have successfully installed OpenVZ and at least one OS template. If there are no OS templates installed on the Hardware Node, turn to the Managing Templates chapter first.

Before you Begin

Before you start creating a Virtual Private Server, you should:

- Check that the Hardware Node is visible on your network. You should be able to connect to/from other hosts. Otherwise, your Virtual Private Servers will not be accessible from other computers.
- Check that you have at least one IP address per Virtual Private Server and the addresses belong to the same network as the Hardware Node or routing to the Virtual Private Servers has been set up via the Hardware Node.

To create a new Virtual Private Server, you have to:

- choose the new Virtual Private Server ID;
- choose the OS template to use for the Virtual Private Server;
- create the Virtual Private Server itself.

Choosing Virtual Private Server ID

Every Virtual Private Server has a numeric ID, also known as VPS ID, associated with it. The ID is a 32-bit integer number beginning with zero and unique for a given Hardware Node. When choosing an ID for your Virtual Private Server, please follow the simple guidelines below:

- ID 0 is used for the Hardware Node itself. You cannot and should not try to create a Virtual Private Server with ID 0.
- OpenVZ reserves the IDs ranging from 0 to 100. Though OpenVZ uses only ID 0, different versions might use additional Virtual Private Servers IDs for internal needs. *To facilitate upgrading, please do not create Virtual Private Servers with IDs below 101.*

The only strict requirement for a VPS ID is to be unique for a particular Hardware Node. However, if you are going to have several computers running OpenVZ, we recommend assigning different VPS ID ranges to them. For example, on Hardware Node 1 you create Virtual Private Servers within the range of IDs from 101 to 1000; on Hardware Node 2 you use the range from 1001 to 2000, and so on. This approach makes it easier to remember on which Hardware Node a Virtual Private Server has been created, and eliminates the possibility of VPS ID conflicts when a Virtual Private Server migrates from one Hardware Node to another.

Another approach to assigning VPS IDs is to follow some pattern of VPS IP addresses. Thus, for example, if you have a subnet with the 10.0.x.x address range, you may want to assign the 17015 ID to the VPS with the 10.0.17.15 IP address, the 39108 ID to the VPS with the 10.0.39.108 IP address, and so on. This makes it much easier to run a number of OpenVZ utilities eliminating the necessity to check up the VPS IP address by its ID and similar tasks. You can also think of your own patterns for assigning VPS IDs depending on the configuration of your network and your specific needs.

Before you decide on a new VPS ID, you may want to make sure that no VPS with this ID has yet been created on the Hardware Node. The easiest way to check whether the VPS with the given ID exists is to issue the following command:

```
# vzlist -a 101
VPS not found
```

This output shows that Virtual Private Server 101 does not exist on the particular Hardware Node; otherwise it would be present in the list.

Choosing OS Template

Next, you shall decide on which OS template you want to base the new VPS. There might be several OS templates installed on the Hardware Node; use the `vzpkgls` command to find out the templates installed on your system:

```
# vzpkgls
fedora-core-3
fedora-core-4
centos-4
```

Creating Virtual Private Server

After the VPS ID and the installed OS template have been chosen, you can create the VPS private area with the `vzctl create` command. The private area is the directory containing the private files of the given VPS. The private area is mounted to the `/vz/root/`*`vpsid/`* directory on the Hardware Node and provides VPS users with a complete Linux file system tree.

The `vzctl create` command requires only the VPS ID and the name of the OS template as arguments; however, in order to avoid setting all the VPS resource control parameters after creating the private area, you can specify a sample configuration to be used for your new Virtual Private Server. The sample configuration files are residing in the `/etc/sysconfig/vz-scripts` directory and have names with the following mask: `ve-`*`config_name`*`.conf-sample`. The most commonly used sample is the `ve-vps.basic.conf-sample` file; this sample file has resource control parameters suitable for most web site Virtual Private Servers.

Thus, for example, you can create a new VPS by typing the following string:

```
# vzctl create 101 --ostemplate fedora-core-4 --config vps.basic
Creating VPS private area
VPS private area was created
```

In this case, OpenVZ will create a Virtual Private Server with ID 101, the private area based on the `fedora-core-4` OS template, and configuration parameters taken from the `ve-vps.basic.conf-sample` sample configuration file.

If you specify neither an OS template nor a sample configuration, `vzctl` will try to take the corresponding values from the global OpenVZ configuration file `/etc/sysconfig/vz`. So you can set the default values in this file using your favorite text file editor, for example:

```
DEF_OSTEMPLATE="fedora-core-4"
CONFIGFILE="vps.basic"
```

and do without specifying these parameters each time you create a new VPS.

Now you can create a VPS with ID 101 with the following command:

```
# vzctl create 101
Creating VPS private area: /vz/private/101
VPS is mounted
Postcreate action done
VPS is unmounted
VPS private area was created
```

In principle, now you are ready to start your newly created Virtual Private Server. However, typically you need to set its network IP address, host name, DNS server address and `root` password before starting the Virtual Private Server for the first time. Please see the next subsection for information on how to perform these tasks.

Configuring Virtual Private Server

Configuring a Virtual Private Server consists of several tasks:

- Setting Virtual Private Server startup parameters;
- Setting Virtual Private Server network parameters;
- Setting Virtual Private Server user passwords;
- Configuring Quality of Service (Service Level) parameters.

For all these tasks, the `vzctl set` command is used. Using this command for setting VPS startup parameters, network parameters, and user passwords is explained later in this subsection. Service Level Management configuration topics are dwelled upon in the Managing Resources chapter.

Setting Startup Parameters

The following options of the `vzctl set` command define the VPS startup parameters: `onboot` and `capability`. To make the Virtual Private Server 101 automatically boot at Hardware Node startup, issue the following command:

```
# vzctl set 101 --onboot yes --save
Saved parameters for VPS 101
```

Setting Network Parameters

In order to be accessible from the network, a Virtual Private Server shall be assigned a correct IP address and host name; DNS server addresses shall also be configured. The session below illustrates setting the Virtual Private Server 101 network parameters:

```
# vzctl set 101 --hostname test101.my.org --save
Hostname for VPS set: test101.my.org
Saved parameters for VPS 101
# vzctl set 101 --ipadd 10.0.186.1 --save
Adding IP address(es): 10.0.186.1
Saved parameters for VPS 101
# vzctl set 101 --nameserver 192.168.1.165 --save
File resolv.conf was modified
Saved parameters for VPS 101
```

This command will assign VPS 101 the IP address of 10.0.186.1, the host name of test101.my.org, and set the DNS server address to 192.168.1.165. The `--save` flag saves all the parameters to the VPS configuration file.

You can issue the above commands when the Virtual Private Server is running. In this case, if you do not want the applied values to persist, you can omit the `--save` option and the applied values will be valid only until the Virtual Private Server shutdown.

To check whether SSH is running inside the Virtual Private Server, use `vzctl exec`, which allows executing any commands in the Virtual Private Server context.

```
# vzctl start 101
[This command starts VPS 101, if it is not started yet]
# vzctl exec 101 service sshd status
sshd is stopped
# vzctl exec 101 service sshd start
Starting sshd: [  OK  ]
# vzctl exec 101 service sshd status
sshd (pid 16036) is running...
```

The above example assumes that VPS 101 is created on the Fedora Core template. For other OS templates, please consult the corresponding OS documentation.

For more information on running commands inside a VPS from the Hardware Node, see the Running Commands in Virtual Private Server subsection.

Setting root Password for VPS

By default, the root account is locked in a newly created VPS, and you cannot log in. In order to log in to the VPS, it is necessary to create a user account inside the Virtual Private Server and set a password for this account or unlock the root account. The easiest way of doing it is to run:

```
# vzctl start 101
[This command starts VPS 101, if it is not started yet]
# vzctl set 101 --userpasswd root:test
```

In this example, we set the root password for VPS 101 to “test”, and you can log in to the Virtual Private Server via SSH as root and administer it in the same way as you administer a standalone Linux computer: install additional software, add users, set up services, and so on. The password will be set inside the VPS in the `/etc/shadow` file in an encrypted form and will not be stored in the VPS configuration file. Therefore, if you forget the password, you have to reset it. Note that `--userpasswd` is the only option of the `vzctl set` command that never requires the `--save` switch, the password is anyway persistently set for the given Virtual Private Server.

While you can create users and set passwords for them using the `vzctl exec` or `vzctl set` commands, it is suggested that you delegate user management to the Virtual Private Server administrator advising him/her of the VPS root account password.

Starting, Stopping, Restarting, and Querying Status of Virtual Private Server

When a Virtual Private Server is created, it may be started up and shut down like an ordinary computer. To start Virtual Private Server 101, use the following command:

```
# vzctl start 101
Starting VPS ...
VPS is mounted
Adding IP address(es): 10.0.186.101
Hostname for VPS 101 set: test.my.org
VPS start in progress...
```

To check the status of a VPS, use the `vzctl status` *`vpsid`* command:

```
# vzctl status 101
VPS 101 exist mounted running
```

Its output shows the following information:

- Whether the VPS private area exists;
- Whether this private area is mounted;
- Whether the Virtual Private Server is running.

In our case, `vzctl` reports that VPS 101 exists, its private area is mounted, and the VPS is running. Alternatively, you can make use of the `vzlist` utility:

```
# vzlist 101
VPSID     NPROC STATUS  IP_ADDR         HOSTNAME
 101         20 running 10.0.186.101    test.my.org
```

Still another way of getting the VPS status is checking the `/proc/vz/veinfo` file. This file lists all the Virtual Private Servers currently running on the Hardware Node. Each line presents a running Virtual Private Server in the `<VPS_ID> <reserved> <number_of_processes> <IP_address>` format:

```
# cat /proc/vz/veinfo
       101     0    20   10.0.186.1
         0     0    48
```

This output shows that VPS 101 is running, there are 20 running processes inside the VPS, and its IP address is 192.168.1.1. Note that second field is reserved; it has no special meaning and should always be zero.

The last line corresponds to the VPS with ID 0, which is the Hardware Node itself.

The following command is used to stop a Virtual Private Server:

```
# vzctl stop 101
Stopping VPS ...
VPS was stopped
VPS is unmounted
# vzctl status 101
VPS 101 exist unmounted down
```

`vzctl` has a two-minute timeout for the VPS shutdown scripts to be executed. If the VPS is not stopped in two minutes, the system forcibly kills all the processes in the Virtual Private Server. The Virtual Private Server will be stopped in any case, even if it is seriously damaged. To avoid waiting for two minutes in case of a Virtual Private Server that is known to be corrupt, you may use the `--fast` switch:

```
# vzctl stop 101 --fast
Stopping VPS ...
VPS was stopped
VPS is unmounted
```

Make sure that you do not use the `--fast` switch with healthy VPSs, unless necessary, as the forcible killing of VPS processes may be potentially dangerous.

The `vzctl start` and `vzctl stop` commands initiate the normal Linux OS startup or shutdown sequences inside the Virtual Private Server. In case of a Red Hat-like distribution, System V initialization scripts will be executed just like on an ordinary computer. You can customize startup scripts inside the Virtual Private Server as needed.

To restart a Virtual Private Server, you may as well use the `vzctl restart` command:

```
# vzctl restart 101
Restarting VPS
Stopping VPS ...
VPS was stopped
VPS is unmounted
Starting VPS ...
VPS is mounted
Adding IP address(es): 10.0.186.101
VPS start in progress...
```

Listing Virtual Private Servers

Very often you may want to get an overview of the Virtual Private Servers existing on the given Hardware Node and to get additional information about them - their IP addresses, hostnames, current resource consumption, etc. In the most general case, you may get a list of all VPSs by issuing the following command:

```
# vzlist -a
     VPSID      NPROC STATUS  IP_ADDR         HOSTNAME
       101          8 running 10.101.66.1     vps101.my.org
       102          7 running 10.101.66.159   vps102.my.org
       103          - stopped 10.101.66.103   vps103.my.org
```

The `-a` switch tells the `vzlist` utility to output both running and stopped VPSs. By default, only running VPSs are shown. The default columns inform you of the VPS IDs, the number of running processes inside VPSs, their status, IP addresses, and hostnames. This output may be customized as desired by using `vzlist` command line switches. For example:

```
# vzlist -o veid,diskinodes.s -s diskinodes.s
     VPSID DQINODES.S
         1     400000
       101     200000
       102     200000
```

This shows only running VPSs with the information about their IDs and soft limit on disk inodes (see the Managing Resources chapter for more information), with the list sorted by this soft limit. The full list of the `vzlist` command line switches and output and sorting options is available in the vzlist subsection of the Reference chapter.

Deleting Virtual Private Server

You can delete a Virtual Private Server that is not needed anymore with the `vzctl destroy` *`VPS_ID`* command. This command removes the Virtual Private Server private area completely and renames the VPS configuration file and action scripts by appending the `.destroyed` suffix to them.

A running VPS cannot be destroyed with the `vzctl destroy` command. The example below illustrates destroying VPS 101:

```
# vzctl destroy 101
VPS is currently mounted (umount first)
# vzctl stop 101
Stopping VPS ...
VPS was stopped
VPS is unmounted
# vzctl destroy 101
Destroying VPS private area: /vz/private/101
VPS private area was destroyed
# ls /etc/sysconfig/vz-scripts/101.*
/etc/sysconfig/vz-scripts/101.conf.destroyed
/etc/sysconfig/vz-scripts/101.mount.destroyed
/etc/sysconfig/vz-scripts/101.umount.destroyed
# vzctl status 101
VPS 101 deleted unmounted down
```

If you do not need the backup copy of the VPS configuration files (with the `.destroyed` suffix), you may delete them manually.

Running Commands in Virtual Private Server

Usually, a Virtual Private Server administrator logs in to the VPS via network and executes any commands in the VPS as on any other Linux box. However, you might need to execute commands inside Virtual Private Servers bypassing the normal login sequence. This can happen if:

- You do not know the Virtual Private Server login information, and you need to run some diagnosis commands inside the VPS in order to verify that it is operational.
- Network access is absent for a Virtual Private Server. For example, the VPS administrator might have accidentally applied incorrect firewalling rules or stopped SSH daemon.

OpenVZ allows you to execute commands in a Virtual Private Server in these cases. Use the `vzctl exec` *`VPS_ID`* command for running a command inside the VPS with the given ID. The session below illustrates the situation when SSH daemon is not started:

```
# vzctl exec 101 /etc/init.d/sshd status
sshd is stopped
# vzctl exec 101 /etc/init.d/sshd start
Starting sshd:[  OK  ]
# vzctl exec 101 /etc/init.d/sshd status
sshd (pid 26187) is running...
```

Now VPS users can log in to the VPS via SSH.

When executing commands inside a Virtual Private Server from shell scripts, use the `vzctl exec2` command. It has the same syntax as `vzctl exec` but returns the exit code of the command being executed instead of the exit code of `vzctl` itself. You can check the exit code to find out whether the command has completed successfully.

If you wish to execute a command in all running VPSs, you can use the following script:

```
# for i in `vzlist –o veid -H`; do \
echo "VPS $i"; vzctl exec $i <command>; done
```

where <*`command`*> is the command to be executed in all the running VPSs. For example:

```
# for i in `vzlist –o veid -H`; do\
echo "VPS $i"; vzctl exec $i uptime; done
VPS 101
  2:26pm  up 6 days,  1:28,  0 users,  load average: 0.00, 0.00, 0.00
VPS 102
  2:26pm  up 6 days,  1:39,  0 users,  load average: 0.00, 0.00, 0.00
[The rest of the output is skipped...]
```

CHAPTER 5

Managing Templates

A *template* is basically a set of packages from some Linux distribution used to populate a VPS. An OS template consists of system programs, libraries, and scripts needed to boot up and run the system (VPS), as well as some very basic applications and utilities. Applications like a compiler and an SQL server are usually not included into an OS template.

In This Chapter

Template Lifecycle

A *template cache* is an OS template installed into a VPS and then packed into a gzipped tar archive. This allows to speed up the creation of a new Virtual Private Server: instead of installing all the packages comprising a Linux distribution, `vzctl` just unpacks the archive.

Template metadata are a set of files containing the information needed to recreate the template cache. It contains the following information:

- List of packages this template comprises
- Locations of (network) package repositories
- Scripts needed to be executed on various stages of template installation
- Public GPG key(s) needed to check signatures of packages
- Additional OpenVZ-specific packages

In order to operate with a template, you should first create its metadata (available from http://openvz.org/download/template/metadata/). The the `vzpkgcache` utility should be run in order to actually create the template cache. It downloads all the packages this template comprises from the network repositories for the given distribution and installs these packages to a temporary VPS, which is then packed into a tar archive to be used later during the creation of new VPSs.

Since this process involves downloading a lot of files (about 400 files, up to 200 Mb in total for a typical distribution) from the Internet, it *might* be sped up using a snapshot of an already fetched repository for a given distribution. Such snapshots are available from http://openvz.org/download/template/repocache/, they are to be unpacked into the `/vz/template` directory. Please note that this step is optional.

In case a template cache (i.e. a tar archive) already exists, `vzpkgcache` tries to bring it up to date by applying the latest updates available from a distribution repository. Since nowadays Linux distributions are updated quite frequently, it makes sense to run this utility from time to time, or at least before doing mass VPS creation.

If there is no need to process all templates, template names can be specified after `vzpkgcache` in the command line, e.g. the following command creates or updates the cache for the Fedora Core 4 template only:

```
# vzpkgcache fedora-core-4
```

When the template cache is ready, it can be installed into a VPS or, in other words, a VPS can be created on the basis of a template. This is performed by the `vzctl create --ostemplate` ***`name`*** command, where ***`name`*** is template name (e.g. `fedora-core-4`).

Listing Templates

The `vzpkgls` utility allows you to list the templates installed on the Hardware Node. They may be already used or not used by certain VPSs:

```
# vzpkgls
fedora-core-4
centos-4
```

As you see, the `fedora-core-4` and `centos 4` templates are available on the Hardware Node. Note that some of them might not be cached yet. To see only those templates that are cached (and thus are ready to be used for creating a VPS), use the `--cached` flag with `vzpkgls`:

```
# vzpkgls ––cached
fedora-core-4
```

Considering the previous output, this means that the `centos-4` template is just installed and is not cached yet.

Specifying a VPS number as a parameter, this command prints the template used by the specified VPS:

```
# vzpkgls 101
fedora-core-4
```

Working with VPS

If you need to update an already existing VPS with the newer packages available from distribution repositories or install some packages that are not part of the template, use the `vzyum` command, which is a simple `yum` wrapper. For example, to update the VPS with ID 123, run:

```
# vzyum 123 update
```

This will find, download, and install all the available updates.

As you may have noticed, a lot of applications are not installed with an OS template. They are to be installed separately, using the `vzyum` utility. For example, if you need the MySQL server inside VPS 123, use the following command:

```
# vzyum 123 install mysql-server
```

Here, `vzyum` will call the `yum` package manager and provide it with all the paths to the repositories suitable for the distribution installed into the VPS. `Yum` will calculate the dependencies, present you with a list of packages to install/update/remove based on what you have asked for and, if confirmed, run a transaction to actually perform all the needed steps. For more information, see the `yum` manual page (`man 8 yum`).

If the package you want to install is already available on Hardware Node, you can use the `vzrpm` utility to install it into a VPS, e.g.:

```
# vzrpm 123 –ihv mypackage-1.0-2.i386.rpm
```

This will install the `mypackage` RPM to VPS 123.

And of course you can do all the usual operations right from inside any VPS: build, install, upgrade, and remove software.

CHAPTER 6

Managing Resources

The main goal of resource control in OpenVZ is to provide Service Level Management or Quality of Service (QoS) for Virtual Private Servers. Correctly configured resource control settings prevent serious impacts resulting from the resource over-usage (accidental or malicious) of any Virtual Private Server on the other Virtual Private Servers. Using resource control parameters for Quality of Service management also allows to enforce fairness of resource usage among Virtual Private Servers and better service quality for preferred VPSs, if necessary.

In This Chapter

What are Resource Control Parameters?

The system administrator controls the resources available to a Virtual Private Server through a set of resource management parameters. All these parameters are defined either in the OpenVZ global configuration file (`/etc/sysconfig/vz`), or in the respective VPS configuration files (`/etc/sysconfig/vz-scripts/`*`VPSID`*`.conf`), or in both. You can set them by manually editing the corresponding configuration files, or by using the OpenVZ command-line utilities. These parameters can be divided into the disk, network, CPU, and system categories. The table below summarizes these groups:

Group	Description	Parameter names	Explained in
Disk	This group of parameters determines disk quota in OpenVZ. The OpenVZ disk quota is realized on two levels: the per-VPS level and the per-user/group level. You can turn on/off disk quota on any level and configure its settings.	`DISK_QUOTA, DISKSPACE, DISKINODES, QUOTATIME, QUOTAUGIDLIMIT`	Managing Disk Quotas
CPU	This group of parameters defines the CPU time different VPSs are guaranteed to receive.	`VE0CPUUNITS, CPUUNITS`	Managing CPU Share

System	This group of parameters defines various aspects of using system memory, TCP sockets, IP packets and like parameters by different VPSs.	`avnumproc, numproc, numtcpsock, numothersock, vmguarpages, kmemsize, tcpsndbuf, tcprcvbuf, othersockbuf, dgramrcvbuf, oomguarpages, lockedpages, shmpages, privvmpages, physpages, numfile, numflock, numpty, numsiginfo, dcachesize, numiptent`	Managing System Parameters

Managing Disk Quotas

This section explains what disk quotas are, defines disk quota parameters, and describes how to perform disk quota related operations:

- Turning on and off per-VPS (first-level) disk quotas;
- Setting up first-level disk quota parameters for a Virtual Private Server;
- Turning on and off per-user and per-group (second-level) disk quotas inside a Virtual Private Server;
- Setting up second-level quotas for a user or for a group;
- Checking disk quota statistics;
- Cleaning up Virtual Private Servers in certain cases.

What are Disk Quotas?

Disk quotas enable system administrators to control the size of Linux file systems by limiting the amount of disk space and the number of inodes a Virtual Private Server can use. These quotas are known as per-VPS quotas or first-level quotas in OpenVZ. In addition, OpenVZ enables the Virtual Private Sever administrator to limit disk space and the number of inodes that individual users and groups in that VPS can use. These quotas are called per-user and per-group quotas or second-level quotas in OpenVZ.

By default, OpenVZ has first-level quotas enabled (which is defined in the OpenVZ global configuration file), whereas second-level quotas must be turned on for each Virtual Private Server separately (in the corresponding VPS configuration files). It is impossible to turn on second-level disk quotas for a Virtual Private Server if first-level disk quotas are off for that Virtual Private Server.

The disk quota block size in OpenVZ is always 1024 bytes. It may differ from the block size of the underlying file system.

OpenVZ keeps quota usage statistics and limits in `/var/vzquota/quota.`*`vpsid`* - a special quota file. The quota file has a special flag indicating whether the file is “dirty”. The file becomes dirty when its contents become inconsistent with the real VPS usage. This means that when the disk space or inodes usage changes during the VPS operation, these statistics are not automatically synchronized with the quota file, the file just gets the “dirty” flag. They are synchronized only when the VPS is stopped or when the HN is shut down. After synchronization, the “dirty” flag is removed. If the Hardware Node has been incorrectly brought down (for example, the power switch was hit), the file remains “dirty”, and the quota is re-initialized on the next VPS startup. This operation may noticeably increase the Node startup time. Thus, it is highly recommended to shut down the Hardware Node properly.

Disk Quota Parameters

The table below summarizes the disk quota parameters that you can control. The File column indicates whether the parameter is defined in the OpenVZ global configuration file (G), in the VPS configuration files (V), or it is defined in the global configuration file but can be overridden in a separate VPS configuration file (GV).

Parameter	Description	File
`disk_quota`	Indicates whether first-level quotas are on or off for all VPSs or for a separate VPS.	GV
`diskspace`	Total size of disk space the VPS may consume, in 1-Kb blocks.	V
`diskinodes`	Total number of disk inodes (files, directories, and symbolic links) the Virtual Private Server can allocate.	V
`quotatime`	The grace period for the disk quota overusage defined in seconds. The Virtual Private Server is allowed to temporarily exceed its quota soft limits for no more than the `QUOTATIME` period.	V
`quotaugidlimit`	Number of user/group IDs allowed for the VPS internal disk quota. If set to 0, the UID/GID quota will not be enabled.	V

Turning On and Off Per-VPS Disk Quotas

The parameter that defines whether to use first-level disk quotas is `DISK_QUOTA` in the OpenVZ global configuration file (`/etc/sysconfig/vz`). By setting it to “no”, you will disable OpenVZ quotas completely.

This parameter can be specified in the Virtual Private Server configuration file (`/etc/sysconfig/vz-scripts/`*`vpsid`*`.conf`) as well. In this case its value will take precedence of the one specified in the global configuration file. If you intend to have a mixture of Virtual Private Servers with quotas turned on and off, it is recommended to set the `DISK_QUOTA` value to “yes” in the global configuration file and to “no” in the configuration file of that VPS which does not need quotas.

The session below illustrates a scenario when first-level quotas are on by default and are turned off for Virtual Private Server 101:

```
[checking that quota is on]
# grep DISK_QUOTA /etc/sysconfig/vz
DISK_QUOTA=yes

[checking available space on /vz partition]
# df /vz
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/sda2              8957295   1421982   7023242  17% /vz

[editing VPS configuration file to add DISK_QUOTA=no]
# vi /etc/sysconfig/vz-scripts/101.conf

[checking that quota is off for VPS 101]
# grep DISK_QUOTA /etc/sysconfig/vz-scripts/101.conf
DISK_QUOTA=no

# vzctl start 101
Starting VPS ...
VPS is mounted
Adding IP address(es): 192.168.1.101
Hostname for VPS set: vps101.my.org
VPS start in progress...
# vzctl exec 101 df
Filesystem           1k-blocks      Used Available Use% Mounted on
simfs                  8282373    747060   7023242  10% /
```

As the above example shows, the only disk space limit a Virtual Private Server with the quotas turned off has is the available space and inodes on the partition where the VPS private area resides.

Note: You must change the `DISK_QUOTA` parameter in the global OpenVZ configuration file only when all Virtual Private Servers are stopped, and in the VPS configuration file – only when the corresponding VPS is stopped. Otherwise, the configuration may prove inconsistent with the real quota usage, and this can interfere with the normal Hardware Node operation.

Setting Up Per-VPS Disk Quota Parameters

Three parameters determine how much disk space and inodes a Virtual Private Server can use. These parameters are specified in the Virtual Private Server configuration file:

`DISKSPACE`	Total size of disk space that can be consumed by the Virtual Private Server in 1-Kb blocks. When the space used by the Virtual Private Server hits the soft limit, the VPS can allocate additional disk space up to the hard limit during the grace period specified by the `QUOTATIME` parameter.
`DISKINODES`	Total number of disk inodes (files, directories, and symbolic links) the Virtual Private Server can allocate. When the number of inodes used by the Virtual Private Server hits the soft limit, the VPS can create additional file entries up to the hard limit during the grace period specified by the `QUOTATIME` parameter.
`QUOTATIME`	The grace period of the disk quota specified in seconds. The Virtual Private Server is allowed to temporarily exceed the soft limit values for the disk space and disk inodes quotas for no more than the period specified by this parameter.

The first two parameters have both soft and hard limits (or, simply, barriers and limits). The hard limit is the limit that cannot be exceeded under any circumstances. The soft limit can be exceeded up to the hard limit, but as soon as the grace period expires, the additional disk space or inodes allocations will fail. Barriers and limits are separated by colons (“:”) in Virtual Private Server configuration files and in the command line.

The following session sets the disk space available to Virtual Private Server 101 to approximately 1Gb and allows the VPS to allocate up to 90,000 inodes. The grace period for the quotas is set to ten minutes:

```
# vzctl set 101 --diskspace 1000000:1100000 --save
Saved parameters for VPS 101
# vzctl set 101 --diskinodes 90000:91000 --save
Saved parameters for VPS 101
# vzctl set 101 --quotatime 600 --save
Saved parameters for VPS 101
# vzctl exec 101 df
Filesystem           1k-blocks      Used Available Use% Mounted on
simfs                  1000000    747066    252934  75% /
# vzctl exec 101 stat -f /
  File: "/"
    ID: 0        Namelen: 255    Type: ext2/ext3
Blocks: Total: 1000000   Free: 252934   Available: 252934   Size: 1024
Inodes: Total: 90000     Free: 9594
```

It is possible to change the first-level disk quota parameters for a running Virtual Private Server. The changes will take effect immediately. If you do not want your changes to persist till the next Virtual Private Server startup, do not use the `--save` switch.

Turning On and Off Second-Level Quotas for Virtual Private Server

The parameter that controls the second-level disk quotas is `QUOTAUGIDLIMIT` in the VPS configuration file. By default, the value of this parameter is zero and this corresponds to disabled per-user/group quotas.

If you assign a non-zero value to the `QUOTAUGIDLIMIT` parameter, this action brings about the two following results:

1. Second-level (per-user and per-group) disk quotas are enabled for the given Virtual Private Server;
2. The value that you assign to this parameter will be the limit for the number of file owners and groups of this VPS, including Linux system users. Note that you will theoretically be able to create extra users of this VPS, but if the number of file owners inside the VPS has already reached the limit, these users will not be able to own files.

Enabling per-user/group quotas for a Virtual Private Server requires restarting the VPS. The value for it should be carefully chosen; the bigger value you set, the bigger kernel memory overhead this Virtual Private Server creates. This value must be greater than or equal to the number of entries in the VPS `/etc/passwd` and `/etc/group` files. Taking into account that a newly created Red Hat Linux-based VPS has about 80 entries in total, the typical value would be 100. However, for Virtual Private Servers with a large number of users this value may be increased.

The session below turns on second-level quotas for Virtual Private Server 101:

```
# vzctl set 101 --quotaugidlimit 100 --save
Unable to apply new quota values: ugid quota not initialized
Saved parameters for VPS 101
# vzctl stop 101; vzctl start 101
Stopping VPS ...
VPS was stopped
VPS is unmounted
Starting VPS ...
VPS is mounted
Adding IP address(es): 192.168.1.101
Hostname for VPS set: vps101.my.org
VPS start in progress...
```

Setting Up Second-Level Disk Quota Parameters

In order to work with disk quotas inside a VPS, you should have standard quota tools installed:

```
# vzctl exec 101 rpm -q quota
quota-3.12-5
```

This command shows that the `quota` package is installed into the Virtual Private Server. Use the utilities from this package (as is prescribed in your Linux manual) to set OpenVZ second-level quotas for the given VPS. For example:

```
# ssh ve101
root@ve101's password:
Last login: Sat Jul 5 00:37:07 2003 from 10.100.40.18
[root@ve101 root]# edquota root
Disk quotas for user root (uid 0):
  Filesystem   blocks      soft      hard     inodes     soft    hard
  /dev/simfs   38216       50000     60000    45454      70000   70000
[root@ve101 root]# repquota -a
*** Report for user quotas on device /dev/simfs
Block grace time: 00:00; Inode grace time: 00:00
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   38218   50000   60000          45453 70000 70000
[the rest of repquota output is skipped]

[root@ve101 root]# dd if=/dev/zero of=test
dd: writing to `test': Disk quota exceeded
23473+0 records in
23472+0 records out
[root@ve101 root]# repquota -a
*** Report for user quotas on device /dev/simfs
Block grace time: 00:00; Inode grace time: 00:00
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      +-   50001   50000   60000   none   45454 70000 70000
[the rest of repquota output is skipped]
```

The above example shows the session when the `root` user has the disk space quota set to the hard limit of 60,000 1Kb blocks and to the soft limit of 50,000 1Kb blocks; both hard and soft limits for the number of inodes are set to 70,000.

It is also possible to set the grace period separately for block limits and inodes limits with the help of the `/usr/sbin/setquota` command. For more information on using the utilities from the `quota` package, please consult the system administration guide shipped with your Linux distribution or manual pages included in the package.

Checking Quota Status

As the Hardware Node system administrator, you can check the quota status for any Virtual Private Server with the `vzquota stat` and `vzquota show` commands. The first command reports the status from the kernel and shall be used for running Virtual Private Servers. The second command reports the status from the quota file (located at `/var/vzquota/quota.`*`vpsid`*) and shall be used for stopped Virtual Private Servers. Both commands have the same output format.

The session below shows a partial output of VPS 101 quota statistics:

```
# vzquota stat 101 –t
   resource          usage       softlimit      hardlimit    grace
  1k-blocks          38281         1000000        1100000
     inodes          45703           90000          91000
User/group quota: on,active
Ugids: loaded 34, total 34, limit 100
Ugid limit was exceeded: no

User/group grace times and quotafile flags:
 type block_exp_time inode_exp_time  dqi_flags
 user                                       0h
group                                       0h

User/group objects:
ID    type   resource    usage   softlimit   hardlimit    grace status
0     user  1k-blocks    38220       50000       60000          loaded
0     user     inodes    45453       70000       70000          loaded
[the rest is skipped]
```

The first three lines of the output show the status of first-level disk quotas for the Virtual Private Server. The rest of the output displays statistics for user/group quotas and has separate lines for each user and group ID existing in the system.

If you do not need the second-level quota statistics, you can omit the `–t` switch from the `vzquota` command line.

Managing CPU Share

The current section explains the CPU resource parameters (CPU share) that you can configure and monitor for each Virtual Private Server.

The table below provides the name and the description for the CPU parameters. The **File** column indicates whether the parameter is defined in the OpenVZ global configuration file (G) or in the VPS configuration files (V).

Parameter	Description	File
`ve0cpuunits`	This is a positive integer number that determines the minimal guaranteed share of the CPU time Virtual Private Server 0 (the Hardware Node itself) will receive. It is recommended to set the value of this parameter to be 5-10% of the power of the Hardware Node.	G

`cpuunits`	This is a positive integer number that determines the minimal guaranteed share of the CPU time the corresponding Virtual Private Server will receive.	V
`cpulimit`	This is a positive number indicating the CPU time in per cent the corresponding VPS is not allowed to exceed.	V

The OpenVZ CPU resource control utilities allow you to guarantee any Virtual Private Server the amount of CPU time this Virtual Private Server receives. The Virtual Private Server can consume more than the guaranteed value if there are no other Virtual Private Servers competing for the CPU and the `cpulimit` parameter is not defined.

To get a view of the optimal share to be assigned to a Virtual Private Server, check the current Hardware Node CPU utilization:

```
# vzcpucheck
Current CPU utilization: 5166
Power of the node: 73072.5
```

The output of this command displays the total number of the so-called CPU units consumed by all running Virtual Private Servers and Hardware Node processes. This number is calculated by OpenVZ with the help of a special algorithm. The above example illustrates the situation when the Hardware Node is underused. In other words, the running Virtual Private Servers receive more CPU time than was guaranteed to them.

In the following example, Virtual Private Server 102 is guaranteed to receive about 2% of the CPU time even if the Hardware Node is fully used, or in other words, if the current CPU utilization equals the power of the Node. Besides, VPS 102 will not receive more than 4% of the CPU time even if the CPU is not fully loaded:

```
# vzctl set 102 --cpuunits 1500 --cpulimit 4 --save
Saved parameters for VPS 102
# vzctl start 102
Starting VPS ...
VPS is mounted
Adding IP address(es): 192.168.1.102
VPS start in progress...
# vzcpucheck
Current CPU utilization: 6667
Power of the node: 73072.5
```

Virtual Private Server 102 will receive from 2 to 4% of the Hardware Node CPU time unless the Hardware Node is overcommitted, i.e. the running Virtual Private Servers have been promised more CPU units than the power of the Hardware Node. In this case the VPS might get less than 2 per cent.

Managing System Parameters

The resources a Virtual Private Server may allocate are defined by the system resource control parameters. These parameters can be subdivided into the following categories: primary, secondary, and auxiliary parameters. The primary parameters are the start point for creating a Virtual Private Server configuration from scratch. The secondary parameters are dependent on the primary ones and are calculated from them according to a set of constraints. The auxiliary parameters help improve fault isolation among applications in one and the same Virtual Private Server and the way applications handle errors and consume resources. They also help enforce administrative policies on Virtual Private Servers by limiting the resources required by an application and preventing the application to run in the Virtual Private Server.

Listed below are all the system resource control parameters. The parameters starting with "num" are measured in integers. The parameters ending in "buf" or "size" are measured in bytes. The parameters containing "pages" in their names are measured in 4096-byte pages (IA32 architecture). The File column indicates that all the system parameters are defined in the corresponding VPS configuration files (V).

Primary parameters

Parameter	Description	File
`avnumproc`	The average number of processes and threads.	V
`numproc`	The maximal number of processes and threads the VPS may create.	V
`numtcpsock`	The number of TCP sockets (PF_INET family, SOCK_STREAM type). This parameter limits the number of TCP connections and, thus, the number of clients the server application can handle in parallel.	V
`numothersock`	The number of sockets other than TCP ones. Local (UNIX-domain) sockets are used for communications inside the system. UDP sockets are used, for example, for Domain Name Service (DNS) queries. UDP and other sockets may also be used in some very specialized applications (SNMP agents and others).	V
`vmguarpages`	The memory allocation guarantee, in pages (one page is 4 Kb). VPS applications are guaranteed to be able to allocate additional memory so long as the amount of memory accounted as `privvmpages` (see the auxiliary parameters) does not exceed the configured barrier of the `vmguarpages` parameter. Above the barrier, additional memory allocation is not guaranteed and may fail in case of overall memory shortage.	V

Secondary parameters

Parameter	Description	File
`kmemsize`	The size of unswappable kernel memory allocated for the internal kernel structures for the processes of a particular VPS.	V
`tcpsndbuf`	The total size of send buffers for TCP sockets, i.e. the amount of kernel memory allocated for the data sent from an application to a TCP socket, but not acknowledged by the remote side yet.	V

`tcprcvbuf`	The total size of receive buffers for TCP sockets, i.e. the amount of kernel memory allocated for the data received from the remote side, but not read by the local application yet.	V
`othersockbuf`	The total size of UNIX-domain socket buffers, UDP, and other datagram protocol send buffers.	V
`dgramrcvbuf`	The total size of receive buffers of UDP and other datagram protocols.	V
`oomguarpages`	The out-of-memory guarantee, in pages (one page is 4 Kb). Any VPS process will not be killed even in case of heavy memory shortage if the current memory consumption (including both physical memory and swap) does not reach the `oomguarpages` barrier.	V

Auxiliary parameters

Parameter	Description	File
`lockedpages`	The memory not allowed to be swapped out (locked with the `mlock()` system call), in pages.	V
`shmpages`	The total size of shared memory (including IPC, shared anonymous mappings and `tmpfs` objects) allocated by the processes of a particular VPS, in pages.	V
`privvmpages`	The size of private (or potentially private) memory allocated by an application. The memory that is always shared among different applications is not included in this resource parameter.	V
`numfile`	The number of files opened by all VPS processes.	V
`numflock`	The number of file locks created by all VPS processes.	V
`numpty`	The number of pseudo-terminals, such as an `ssh` session, the `screen` or `xterm` applications, etc.	V
`numsiginfo`	The number of `siginfo` structures (essentially, this parameter limits the size of the signal delivery queue).	V
`dcachesize`	The total size of `dentry` and `inode` structures locked in the memory.	V
`physpages`	The total size of RAM used by the VPS processes. This is an accounting-only parameter currently. It shows the usage of RAM by the VPS. For the memory pages used by several different VPSs (mappings of shared libraries, for example), only the corresponding fraction of a page is charged to each VPS. The sum of the `physpages` usage for all VPSs corresponds to the total number of pages used in the system by all the accounted users.	V
`numiptent`	The number of IP packet filtering entries.	V

You can edit any of these parameters in the `/etc/sysconfig/vz-scripts/`*`vpsid`*`.conf` file of the corresponding VPS by means of your favorite text editor (for example, `vi` or `emacs`), or by running the `vzctl set` command. For example:

```
# vzctl set 101 --kmemsize 2211840:2359296 --save
Saved parameters for VPS 101
```

Monitoring System Resources Consumption

It is possible to check the system resource control parameters statistics from within a Virtual Private Server. The primary use of these statistics is to understand what particular resource has limits preventing an application to start. Moreover, these statistics report the current and maximal resources consumption for the running Virtual Private Server. This information can be obtained from the `/proc/user_beancounters` file.

The output below illustrates a typical session:

```
# vzctl exec 101 cat /proc/user_beancounters
Version: 2.5
       uid  resource           held    maxheld    barrier      limit    failcnt
       101: kmemsize         803866    1246758    2457600    2621440          0
            lockedpages           0          0         32         32          0
            privvmpages        5611       7709      22528      24576          0
            shmpages             39        695       8192       8192          0
            dummy                 0          0          0          0          0
            numproc              16         27         65         65          0
            physpages          1011       3113          0 2147483647          0
            vmguarpages           0          0       6144 2147483647          0
            oomguarpages       2025       3113       6144 2147483647          0
            numtcpsock            3          4         80         80          0
            numflock              2          4        100        110          0
            numpty                0          1         16         16          0
            numsiginfo            0          2        256        256          0
            tcpsndbuf             0       6684     319488     524288          0
            tcprcvbuf             0       4456     319488     524288          0
            othersockbuf       2228       9688     132096     336896          0
            dgramrcvbuf           0       4276     132096     132096          0
            numothersock          4         17         80         80          0
            dcachesize        78952     108488     524288     548864          0
            numfile             194        306       1280       1280          0
            dummy                 0          0          0          0          0
            dummy                 0          0          0          0          0
            dummy                 0          0          0          0          0
            numiptent             0          0        128        128          0
```

The failcnt column displays the number of unsuccessful attempts to allocate a particular resource. If this value increases after an application fails to start, then the corresponding resource limit is in effect lower than is needed by the application.

The held column displays the current resource usage, and the maxheld column – the maximal value of the resource consumption for the last accounting period. The meaning of the barrier and limit columns depends on the parameter and is explained in the OpenVZ Management of System Resources guide.

Inside a VPS, the `/proc/user_beancounters` file displays the information on the given VPS only, whereas from the Hardware Node this file displays the information on all the VPSs.

To check the UBC usage for a HN (summary for all running VPSs), you can use the following scripts:

(for any resource accounted in pages)

```
# for res in lockedpages totvmpages ipcshmpages anonshpages rsspages;\
do echo;echo "$res usage for all VEs, in MB:";cat \
/proc/user_beancounters |grep $res|awk 'BEGIN{ cur=max=lim=0; } \
{ cur+=$2; max+=$3;lim+=$5 } END {print "held:",cur*4/1024, "max:", \
max*4/1024, "limit:", lim*4/1024}'; done
```

(for kmemsize and other resources accounted in bytes)

```
# for res in tcpsendbuf tcprcvbuf unixsockbuf sockrcvbuf kmemsize; \
do echo;echo "$res usage for all VEs, in MB:";cat \
/proc/user_beancounters |grep $res|sed "s/[[:digit:]]\+://g" \
|awk 'BEGIN{ cur=max=lim=0; } { cur+=$2; max+=$3;lim+=$5 } \
END {print "held:",cur/1024/1024, "max:", max/1024/1024, \
"limit:", lim/1024/1024}'; done
```

Monitoring Memory Consumption

You can monitor a number of memory parameters for the whole Hardware Node and for particular Virtual Private Servers with the help of the `vzmemcheck` utility. For example:

```
# vzmemcheck -v
Output values in %
veid    LowMem  LowMem     RAM MemSwap MemSwap   Alloc   Alloc   Alloc
          util  commit    util    util  commit    util  commit   limit
101       0.19    1.93    1.23    0.34    1.38    0.42    1.38    4.94
1         0.27    8.69    1.94    0.49    7.19    1.59    2.05   56.54
--------------------------------------------------------------------
Summary:  0.46   10.62    3.17    0.83    8.57    2.02    3.43   61.48
```

The `–v` option is used to display the memory information for each Virtual Private Server and not for the Hardware Node in general. It is also possible to show the absolute values in Megabytes by using the `–A` switch. The monitored parameters are (from left to right in the output above) low memory utilization, low memory commitment, RAM utilization, memory+swap utilization, memory+swap commitment, allocated memory utilization, allocated memory commitment, allocated memory limit.

To understand these parameters, let us first draw the distinction between utilization and commitment levels. *Utilization* level is the amount of resources consumed by VPSs at the given time. In general, low utilization values mean that the system is under-utilized. Often, it means that the system is capable of supporting more Virtual Private Servers if the existing VPSs continue to maintain the same load and resource consumption level. High utilization values (in general, more than 1, or 100%) mean that the system is overloaded and the service level of the Virtual Private Servers is degraded. *Commitment* level shows how much resources are "promised" to the existing Virtual Private Servers. Low commitment levels mean that the system is capable of supporting more Virtual Private Servers. Commitment levels more than 1 mean that the Virtual Private Servers are promised more resources than the system has, and the system is said to be *overcommitted*. If the system runs a lot of VPSs, it is usually acceptable to have some overcommitment because it is unlikely that all Virtual Private Servers will request resources at one and the same time. However, very high commitment levels will cause VPSs to fail to allocate and use the resources promised to them and may hurt system stability.

There follows an overview of resources checked up by the `vzmemcheck` utility. Their complete description is provided in the OpenVZ Management of System Resources guide.

The *low memory* is the most important RAM area representing the part of memory residing at lower addresses and directly accessible by the kernel. In OpenVZ, the size of the "low" memory area is limited to 832 MB in the UP (uniprocessor) and SMP versions of the kernel, and to 3.6 GB in the Enterprise version of the kernel. If the total size of the computer RAM is less than the limit (832 MB or 3.6 GB, respectively), then the actual size of the "low" memory area is equal to the total memory size.

The union of *RAM and swap* space is the main computer resource determining the amount of memory available to applications. If the total size of memory used by applications exceeds the RAM size, the Linux kernel moves some data to swap and loads it back when the application needs it. More frequently used data tends to stay in RAM, less frequently used data spends more time in swap. Swap-in and swap-out activity reduces the system performance to some extent. However, if this activity is not excessive, the performance decrease is not very noticeable. On the other hand, the benefits of using swap space are quite big, allowing to increase the number of Virtual Private Servers in the system by 2 times. Swap space is essential for handling system load bursts. A system with enough swap space just slows down at high load bursts, whereas a system without swap space reacts to high load bursts by refusing memory allocations (causing applications to refuse to accept clients or terminate) and directly killing some applications. Additionally, the presence of swap space helps the system better balance memory and move data between the low memory area and the rest of the RAM.

Allocated memory is a more "virtual" system resource than the RAM or RAM plus swap space. Applications may allocate memory but start to use it only later, and only then will the amount of free physical memory really decrease. The sum of the sizes of memory allocated in all Virtual Private Servers is only the estimation of how much physical memory will be used if all applications claim the allocated memory. The memory available for allocation can be not only used (the Alloc util column) or promised (the Alloc commit column), but also limited (applications will not be able to allocate more resources than is indicated in the Alloc limit column).

Managing VPS Resources Configuration

Any VPS is configured by means of its own configuration file. You can manage your VPS configurations in a number of ways:

1 Using configuration sample files shipped with OpenVZ. These files are used when a new Virtual Private Server is being created (for details, see the Creating and Configuring New Virtual Private Server section on page 31). They are stored in the same directory as VPS configuration files (`/etc/sysconfig/vz-scripts/`) and have the `ve-`*`name`*`.conf-sample` mask. Currently, the following configuration sample files are provided:

   - `light` – to be used for creating "light" VPSs having restrictions on the upper limit of quality of service parameters;
   - `vps.basic` – to be used for common VPSs.

   Note: Configuration sample files cannot contain spaces in their names.

   Any sample configuration file may also be applied to a Virtual Private Server after it has been created. You would do this if, for example, you want to upgrade or downgrade the overall resources configuration of a particular VPS:

```
# vzctl set 101 --applyconfig light --save
```

   This command applies all the parameters from the `ve-light.conf-sample` file to the given VPS, except for the `OSTEMPLATE`, `VE_ROOT`, and `VE_PRIVATE` parameters, should they exist in the sample configuration file.

2 Using OpenVZ specialized 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 configuration file (`/etc/sysconfig/vz-scripts/`*`VPS_ID`*`.conf`). This can be performed either with the help of any text editor. The instructions on how to edit VPS 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.

Splitting Hardware Node Into Equal Pieces

It is possible to create a Virtual Private Server configuration roughly representing a given fraction of the Hardware Node. If you want to create such a configuration that up to 20 fully loaded Virtual Private Servers would be able to be simultaneously running on the given Hardware Node, you can do it as is illustrated below:

```
# cd /etc/sysconfig/vz-scripts/
# vzsplit -n 20 -f vps.mytest
Config /etc/sysconfig/vz-scripts/ve-vps.mytest.conf-sample was created
# vzcfgvalidate ve-vps.mytest.conf-sample
Recommendation: kmemsize.lim-kmemsize.bar should be > 253952 \
(currently, 126391)
Recommendation: dgramrcvbuf.bar should be > 132096 (currently, 93622)
```

Note that the configuration produced depends on the given Hardware Node resources. Therefore, it is important to validate the resulted configuration file before trying to use it, which is done with the help of the `vzcfgvalidate` utility.

The number of Virtual Private Servers you can run on the Hardware Node is actually several times greater than the value specified in the command line because Virtual Private Servers normally do not consume all the resources that are guaranteed to them. To illustrate this idea, let us look at the Virtual Private Server created from the configuration produced above:

```
# vzctl create 101 --ostemplate fedora-core-4 --config vps.mytest
Creating VPS private area: /vz/private/101
VPS private area was created
# vzctl set 101 --ipadd 192.168.1.101 --save
Saved parameters for VPS 101
# vzctl start 101
Starting VPS ...
VPS is mounted
Adding IP address(es): 192.168.1.101
VPS start in progress...
# vzcalc 101
Resource     Current(%)  Promised(%)  Max(%)
Memory           0.53       1.90       6.44
```

As is seen, if Virtual Private Servers use all the resources guaranteed to them, then around 20 VPSs can be simultaneously running. However, taking into account the Promised column output, it is safe to run 40-50 such Virtual Private Servers on this Hardware Node.

Validating Virtual Private Server Configuration

The system resource control parameters have complex interdependencies. Violation of these interdependencies can be catastrophic for the Virtual Private Server. In order to ensure that a Virtual Private Server does not break them, it is important to validate the VPS configuration file before creating VPSs on its basis.

The typical validation scenario is shown below:

```
# vzcfgvalidate /etc/sysconfig/vz-scripts/101.conf
Error: kmemsize.bar should be > 1835008 (currently, 25000)
Recommendation: dgramrcvbuf.bar should be > 132096 (currently, 65536)
Recommendation: othersockbuf.bar should be > 132096 (currently,
122880)
# vzctl set 101 --kmemsize 2211840:2359296 --save
Saved parameters for VPS 101
# vzcfgvalidate  /etc/sysconfig/vz-scripts/101.conf
Recommendation: kmemsize.lim-kmemsize.bar should be > 163840
(currently, 147456)
Recommendation: dgramrcvbuf.bar should be > 132096 (currently, 65536)
Recommendation: othersockbuf.bar should ba > 132096 (currently,
122880)
Validation completed: success
```

The utility checks constraints on the resource management parameters and displays all the constraint violations found. There can be three levels of violation severity:

Recommendation	This is a suggestion, which is not critical for Virtual Private Server or Hardware Node operations. The configuration is valid in general; however, if the system has enough memory, it is better to increase the settings as advised.
Warning	A constraint is not satisfied, and the configuration is invalid. The Virtual Private Server applications may not have optimal performance or may fail in an ungraceful way.
Error	An important constraint is not satisfied, and the configuration is invalid. The Virtual Private Server applications have increased chances to fail unexpectedly, to be terminated, or to hang.

In the scenario above, the first run of the `vzcfgvalidate` utility found a critical error for the `kmemsize` parameter value. After setting reasonable values for `kmemsize`, the resulting configuration produced only recommendations, and the Virtual Private Server can be safely run with this configuration.

CHAPTER 7

Advanced Tasks

In This Chapter

Determining VPS ID by Process ID

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 Virtual Private Server ID the process with the given id belongs to. Multiple process IDs can be specified as arguments. In this case the utility will print the Virtual Private Server number for each of the processes.

The typical output of the `vzpid` utility is shown below:

```
[root@ts23 root]# vzpid 12
Pid     VPS    Name
12      4      init
```

In our example the process with the identifier 12 has the name 'init' and is running in the Virtual Private Server with ID = 4.

Changing System Time from VPS

Normally it is impossible to change the system time from a Virtual Private Server. Otherwise, different Virtual Private Servers could interfere with each other and could even break applications depending on the system time accuracy.

Normally only the Hardware Node system administrator can change the system time. However, if you want to synchronize the time via Network Time Protocol (NTP), you have to run NTP software, which will connect to external NTP servers and update the system time. It is not advisable to run application software on the Hardware Node itself, since flaws in the software can lead to compromising all Virtual Private Servers on the Hardware Node. Thus, if you plan to use NTP, you shall create a special Virtual Private Server for it and configure it to have the `sys_time` capability. The example below illustrates configuring such a Virtual Private Server:

```
# vzctl set 101 --capability sys_time:on --save
Unable to set capability on running VPS
Saved parameters for VPS 101
```

The output of the above command warns you that `vzctl` cannot apply changes in the capabilities to a running Virtual Private Server. The VPS has to be restarted before changes take effect:

```
# vzctl restart 101
Restarting VPS
Stopping VPS ...
VPS was stopped
VPS is unmounted
Starting VPS ...
VPS is mounted
Adding IP address(es): 192.168.1.101
Hostname for VPS set: vps101.my.org
VPS start in progress...
# ssh root@vps101
```

```
root@vps101's password:
Last login: Mon Oct 28 23:25:58 2002 from 10.100.40.18
[root@vps101 root]# date
Mon Oct 28 23:31:57 EST 2002
[root@vps101 root]# date 10291300
Tue Oct 29 13:00:00 EST 2002
[root@vps101 root]# date
Tue Oct 29 13:00:02 EST 2002
[root@vps101 root]# logout
Connection to ve101 closed.
# date
Tue Oct 29 13:01:31 EST 2002
```

The command session above shows the way to change the system time from Virtual Private Server 101. The changes will affect all the Virtual Private Servers and the Hardware Node itself. It is not advisable to have more than one Virtual Private Server with the `sys_time` capability set on.

NTP is described in Internet Standard RFC 1305; more information including client software can be obtained from the NTP web server (http://www.ntp.org/).

Accessing Devices from Inside Virtual Private Server

It is possible to grant a Virtual Private Server read, write, or read/write access to a character or block device. This might be necessary, for example, for Oracle database software if you want to employ its ability to work with raw disk partitions.

In most cases, providing access to the file system hierarchy for a Virtual Private Server is achieved by using bind mounts. However, bind mounts do not allow you to create new partitions, format them with a file system, or mount them inside a Virtual Private Server. If you intend to delegate disk management to a Virtual Private Server administrator, you shall use either the `–-devices` or the `--devnodes` option of the `vzctl set` command.

The example session below illustrates the following situation: you want to allow the root user of Virtual Private Server 101 to take responsibility for administering the `/dev/sdb`, `/dev/sdb1` and `/dev/sdb2` devices. In other words, you allow the VPS 101 system administrator to repartition the `/dev/sdb` device and create file systems on the first two partitions (or use them with any software capable of working with raw block devices, such as Oracle database software).

First, we are going to grant the Virtual Private Server the permissions to work with the needed block devices:

```
# vzctl set 101 --devices b:8:16:rw --devices b:8:17:rw --devices
 b:8:18:rw --save
Setting devperms
Saved parameters for VPS 101
```

This command sets the read/write permissions for block devices with major number 8 and minor numbers 16, 17 and 18 (corresponding to `/dev/sdb`, `/dev/sdb1`, and `/dev/sdb2`). If you are not sure which major and minor numbers correspond to the necessary block devices, you may issue the following command:

```
# ls -l /dev/sdb{,1,2}
brw-rw----    1 root     disk       8,  16 Jan 30 13:24 /dev/sdb
brw-rw----    1 root     disk       8,  17 Jan 30 13:24 /dev/sdb1
brw-rw----    1 root     disk       8,  18 Jan 30 13:24 /dev/sdb2
```

Now let us create a 100-Mb Linux partition in addition to an already existing 2 GB partition on `/dev/sdb1` from VPS 101.

```
[root@vps101 root]# fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 255 heads, 63 sectors, 2231 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sdb1   *         1       255   2048256   83  Linux

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (256-2231, default 256):
Using default value 256
Last cylinder or +size or +sizeM or +sizeK \
(256-2231, default 2231): +100M

Command (m for help): p

Disk /dev/sdb: 255 heads, 63 sectors, 2231 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sdb1   *         1       255   2048256   83  Linux
/dev/sdb2           256       268    104422+  83  Linux

Command (m for help): w
```

After the new partition table has been written, you can format it and mount inside the Virtual Private Server:

```
[root@vps101 root]# mke2fs /dev/sdb2
[Output of mke2fs is skipped…]
[root@vps101 root]# mount /dev/sdb2 /mnt
[root@vps101 root]# df
Filesystem           1k-blocks      Used Available Use% Mounted on
simfs                  1048576    149916    898660  15% /
ext2                    101107        13     95873   1% /mnt
```

Remember that you have to specify all minors for the devices you want to delegate authority for; allowing to access `/dev/sdb` grants the permission to create, modify and delete partitions on it, but explicit permissions shall be given for partitions you allow the Virtual Private Server to work with.

Moving Network Adapter to Virtual Private Server

By default, all the VPSs on a Node are connected among themselves and with the Node by means of a virtual network adapter called `venet0`. Still, there is a possibility for a VPS to directly access a physical network adapter (for example, `eth1`). In this case the adapter becomes inaccessible to the Hardware Node itself. This is done with the help of the `vzctl` command:

```
# vzctl set 101 --netdev_add eth1 --save
Add network device: eth1
Saved parameters for VPS 101
```

Mind that the network device added to a VPS in such a way has the following limitations:

- This network device will be accessible only to the VPS whereto it has been moved, but not to the Hardware Node (VPS 0) and not to all the other VPSs on the Node.
- If such a device is removed from the VPS (by means of the `vzctl set --netdev_del` command) and added to another VPS instead, all the network settings of this device are purged. To work around this problem, you should store all the device settings in the `ifcfg-`*`dev`* file and have this file available in the `/etc/sysconfig/network-scripts` directory inside all the VPSs that may have access to this device (including VPS 0). After the device has been added to a VPS, it will be enough to issue the `ifup` *`dev`* command inside the VPS to read the settings from the file mentioned above. Mind though that this will still not restore advanced network configuration settings, such as packet filtering rules.
- The physical device inside a VPS has no security restrictions typical for the `venet` virtual device. Inside the VPS it will be possible to assign any IP address to this device and use it, to sniff network traffic in the promiscuous mode, and so on.

Enabling VPN for VPS

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 VPS is possible via the TUN/TAP device. To allow a particular VPS to use this device, the following steps are required:

- Make sure the `tun` module is already loaded before OpenVZ is started:

```
# lsmod | grep tun
```

In case it is not loaded, load it with the following command:

```
# modprobe tun
```

- Allow the VPS to use the TUN/TAP device:

```
# vzctl set 101 --devices c:10:200:rw --save
```

- Create the corresponding device inside the VPS and set the proper permissions:

```
# vzctl exec 101 mkdir -p /dev/net
# vzctl exec 101 mknod /dev/net/tun c 10 200
# vzctl exec 101 chmod 600 /dev/net/tun
```

Configuring the VPN proper is carried out as 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 (http://vtun.sourceforge.net/) and OpenVPN (http://openvpn.sourceforge.net/).

Loading iptables Modules

The OpenVZ kernel provides support for additional `iptables` modules that are not loaded automatically. If you want any of these modules to be loaded either to the Hardware Node or, additionally, to any particular VPSs, you should do some manual operations.

Loading iptables Modules to Hardware Node

To have certain `iptables` modules loaded on the Hardware Node startup, you should provide their names as the value of the `IPTABLES_MODULES` parameter in the `/etc/sysconfig/iptables-config` file. The default value of this parameter is the following:

```
IPTABLES_MODULES="ip_tables ipt_REJECT ipt_tos ipt_limit ipt_multiport
                  iptable_filter iptable_mangle ipt_TCPMSS ipt_tcpmss
                  ipt_ttl ipt_length"
```

You may modify this value to add any of the following modules:

```
ip_conntrack
ip_conntrack_ftp
ip_conntrack_irc
ipt_LOG
ipt_conntrack
ipt_helper
ipt_state
iptable_nat
ip_nat_ftp
ip_nat_irc
ipt_TOS
```

All the modules indicated as the value of this parameter will be loaded on the Node startup after you reboot the Hardware Node. However, if you want this set of modules to be loaded by default to the VPSs hosted on this Node or you wish to restrict loading any of these modules to all or particular VPSs, you should perform some additional steps.

Loading iptables Modules to Particular VPSs

What `iptables` modules are loaded by default inside the VPSs hosted on the given Node is determined by the value of the `IPTABLES` parameter in the `/etc/sysconfig/vz` file. Naturally, those modules that constitute the value of this parameter will be loaded to VPSs only in case they are also loaded on the Hardware Node itself (see page 70). This parameter can also be redefined both in VPS sample configuration files (`/etc/sysconfig/vz-scripts/ve-`*`sample_name`*`.conf-sample`) and in the configuration files of particular VPSs (`/etc/sysconfig/vz-scripts/`*`vps_id`*`.conf`).

In order to load extra `iptables` modules or not to load certain default modules inside particular VPSs, you should explicitly indicate what modules you *wish* to be loaded to these VPSs either by modifying the `IPTABLES` parameter in the respective VPS configuration files or by using the `vzctl` command. For example:

```
# vzctl set 101 --iptables iptable_filter --iptables ipt_length --
iptables ipt_limit --iptables iptable_mangle --iptables ipt_REJECT --
save
```

This command will tell OpenVZ to load only the following modules to VPS 101: `iptable_filter`, `ipt_length`, `ipt_limit`, `iptable_mangle`, `ipt_REJECT`. This information will also be saved in the VPS configuration file thanks to the `--save` option.

Loading a new set of `iptables` modules does not happen on the fly. You should restart the VPS for the changes to take effect.

Rebooting Virtual Private Server

When you issue the `reboot` command at your Linux box console, the command makes the reboot system call with argument '`restart`', which is passed to the computer BIOS. The Linux kernel then reboots the computer. For obvious reasons this system call is blocked inside Virtual Private Servers: no Virtual Private Server can access BIOS directly; otherwise, a reboot inside a VPS would reboot the whole Hardware Node. That is why the `reboot` command inside a VPS actually works in a different way. On executing the `reboot` command inside a VPS, the VPS is stopped and then started by a special script (`/etc/sysconfig/vz-scripts/vpsreboot`) which is executed periodically (every minute by default) by the cron daemon. Cron configuration to run the script is in the file `/etc/cron.d/vpsreboot`.

If you want a Virtual Private Server to be unable to initiate reboot itself, add the `ALLOWREBOOT=”no”` line to the Virtual Private Server configuration file (`/etc/sysconfig/vz-scripts/`*`vps_id`*`.conf`). If you want to have VPS reboot disabled by default and want to specify explicitly which Virtual Private Servers are allowed to reboot, add the `ALLOWREBOOT=”no”` line to the OpenVZ global configuration file (`/etc/sysconfig/vz`) and explicitly specify `ALLOWREBOOT=”yes”` in the corresponding Virtual Private Server configuration files.

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.

In This Chapter

General Considerations

The general issues to take into consideration when troubleshooting your OpenVZ system are listed below. You should read them carefully before trying to solve more specific problems.

- You should always remember where you are located now in your terminal. Check it periodically using the `pwd`, `hostname`, `ifconfig`, `cat /proc/vz/veinfo` commands. One and the same command executed inside a VPS and at the HN 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 that, 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`, `top` 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`). See Using ALT+SYSRQ Keyboard Sequences section for more information.
- If the Hardware Node was incorrectly brought down, on its next startup all the partitions will be checked and quota recalculated for each VPS, which dramatically increases the startup time.
- Do not run any binary or script that belongs to a VPS directly from the Hardware Node, for example, *do not* ever do that:

```
cd /vz/root/99/etc/init.d
./httpd status
```

Any script inside a VPS could have been changed to whatever the VPS owner chooses: it could have been trojaned, replaced to something like `rm -rf`, etc. You can use only `vzctl exec` or `vzctl enter` to execute programs inside a VPS.

- Do not use init scripts at the Hardware Node. An init script may use `killall` to stop a service, which means that all similar processes will be killed in all VPSs! You can check `/var/run/`*`service`*`.pid` and kill the correspondent process explicitly.
- You must be able to detect any rootkit inside a VPS. It is recommended to use the `chkrootkit` package for detection (you can download the latest version from 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 VPS and make sure it is 100% removed. If you're not sure, create a new VPS for that customer and migrate her data 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`, `vzctl.log` log files, 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 turn to the `messages` log file first and then proceed with the other files from the `/var/log/` directory.
- Subscribe to bug tracking lists, at least for Red Hat. You should keep track of new public DoS tools or remote exploits for the software and install them into VPSs or at Hardware Nodes.
- When using `iptables`, there is a simple rule for Chains usage to help protect both the HN and its VPSs:
  - use INPUT, OUTPUT to filter packets that come in/out the HN;
  - use FORWARD to filter packets that are designated for VPSs.

Kernel Troubleshooting

Using ALT+SYSRQ Keyboard Sequences

Press ALT+SYSRQ+H (3 keys simultaneously) and check what's printed at the HN console, for example:

```
SysRq : HELP : loglevel0-8 reBoot tErm Full kIll saK showMem Nice
powerOff showPc unRaw Sync showTasks Unmount
```

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 press the following 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 all mounted filesystems;

ALT+SYSRQ+E to terminate processes;

ALT+SYSRQ+I to kill all processes

ALT+SYSRQ+U to try to unmount once again;

ALT+SYSRQ+B to reboot.

If the computer is not rebooted after that, you can press the Power button.

Saving Kernel Fault (OOPS)

You can use the following command to check for the kernel messages that should be decoded and reported to OpenVZ developers:

```
grep -E "Call Trace|Code" /var/log/messages*
```

Then you should find these lines in the correspondent log file and figure out what kernel was booted when the oops occurred. Search backward for the "Linux" string, look for strings like that:

```
May 23 16:55:00 ts13 Linux version 2.6.8-022stab026.1 (root@kern26x.build.sw.ru) (gcc
version 3.3.3 20040412 (Red Hat Linux 3.3.3-7)) #1 Fri Jul 8 17:31:10 MSD 2005
```

An oops usually starts with some description of what happened and ends with the Code string. Here is an example:

```
May 24 15:12:07 ts13 Unable to handle kernel paging request at virtual address d0d48b08
May 24 15:12:07 ts13  printing eip:
May 24 15:12:07 ts13 c01b4049
May 24 15:12:07 ts13 *pde = 00044063
May 24 15:12:07 ts13 *pte = 10d48000
May 24 15:12:07 ts13 Oops: 0000 [#1]
May 24 15:12:07 ts13 SMP DEBUG_PAGEALLOC
May 24 15:12:07 ts13 Modules linked in: e100 mii af_packet ip_nat_ftp ip_nat_irc
ipt_helper ip_conntrack_irc ip_conntrack_ftp ipt_TOS ipt_LOG ipt_conntrack ipt_state
iptable_nat ip_conntrack ipt_length ipt_ttl ipt_tcpmss ipt_TCPMSS iptable_mangle
iptable_filter ipt_multiport ipt_limit ipt_tos ipt_REJECT ip_tables
May 24 15:12:07 ts13 CPU:    0, VCPU: 2147483647:0
May 24 15:12:07 ts13 EIP:    0060:[<c01b4049>]    Not tainted
May 24 15:12:07 ts13 EFLAGS: 00010206
May 24 15:12:07 ts13 EIP is at proc_pid_stat+0x289/0x5b0
May 24 15:12:07 ts13 eax: d0d48a70   ebx: 00000000   ecx: 00000000   edx: c0128962
May 24 15:12:07 ts13 esi: 00000000   edi: c599fa70   ebp: d93f2f34   esp: d93f2e04
May 24 15:12:07 ts13 ds: 007b   es: 007b   ss: 0068
May 24 15:12:07 ts13 Process top (pid: 19753, threadinfo=d93f2000 task=d93f1a70)
May 24 15:12:07 ts13 Stack: c599fa70 00000000 c041b980 d93f2e50 c018e7bd cc752f58
c041b980 d8b4df58
May 24 15:12:07 ts13        d93f2e4c c01b0cea c599fa70 c627a00c 00000004 df1c3f58
00000000 d8b4df58
May 24 15:12:07 ts13        cc752f58 00000000 c0d97e94 d93f2e6c c018eb9e 04837000
00000000 d93f2e84
May 24 15:12:07 ts13 Call Trace:
May 24 15:12:07 ts13  [<c010650f>] show_stack+0x7f/0xa0
May 24 15:12:07 ts13  [<c01066df>] show_registers+0x17f/0x220
May 24 15:12:07 ts13  [<c01068c7>] die+0xa7/0x170
May 24 15:12:07 ts13  [<c01188ea>] do_page_fault+0x2fa/0x59e
May 24 15:12:07 ts13  [<c034576f>] error_code+0x2f/0x38
May 24 15:12:07 ts13  [<c01afe71>] proc_info_read+0x51/0x160
May 24 15:12:07 ts13  [<c0171dea>] vfs_read+0xaa/0x130
May 24 15:12:07 ts13  [<c017208b>] sys_read+0x4b/0x80
May 24 15:12:07 ts13  [<c0344cda>] sysenter_past_esp+0x43/0x61
May 24 15:12:07 ts13 Code: 8b 80 98 00 00 00 89 45 b0 f0 ff 05 c0 b7 41 c0 8b 87 88 01
```

All you need is to put the oops into a file.

Finding Kernel Function That Caused 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 you should 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.

Problems with VPS Management

This section includes recommendations on how to settle some problems with your VPSs.

Failure to Create VPS

An attempt to create a new Virtual Private Server fails. There is a message on the system console: `Cached os template /vz/template/cache/`*`XXX`*`.tar.gz not found`.

Solution

The necessary OS template might be absent from the Hardware Node. Copy the template to the Hardware Node, install it, cache it, and try to create a VPS once again.

Failure to Start VPS

An attempt to start a Virtual Private Server fails.

Solution 1

If there is a message on the system console: `parameters missing`, and the list of missed parameters follows the message, set these parameters using the `vzctl set --save` command (see Configuring Virtual Private Server on page 34 for instructions). Try to start the VPS once again.

Solution 2

If there is a message on the system console: `Address already in use`, issue the `cat /proc/vz/veinfo` command. The information about the VPS numeric identifier, VPS class, number of VPS's processes and VPS IP address shall be displayed for each running VPS. This shall also demonstrate that your VPS is up, i.e. it must be running without any IP address assigned. Set its IP address using the command:

```
vzctl set vps_id --ipadd addr --save
```

where *`vps_id`* represents the VPS numeric identifier and *`addr`* represents an actual IP address.

Solution 3

Poor UBC parameters might prevent the VPS from starting. Try to validate the VPS configuration (see Validating Virtual Private Server Configuration on page 62). See what configuration parameters have caused the error and set appropriate values using the `vzctl set --save` command.

Solution 4

The VPS might have used all its disk quota (either disk space or disk inodes). Check the VPS disk quota (see the Managing Disk Quotas section and Chapter 6 for details) and increase the quota parameters if needed (see Setting Up Per-VPS Disk Quota Parameters on page 50).

Failure to Access VPS From Network

Solution 1

The IP address assigned to this Virtual Private Server might be already in use in your network. Make sure it is not. The problem VPS address can be checked by issuing the following command:

```
# grep IP_ADDRESS /etc/sysconfig/vz-scripts/VPS_ID.conf
IP_ADDRESS="10.0.186.101"
```

The IP addresses of other VPSs, which are running, can be checked by running

```
cat /proc/vz/veinfo
```

Solution 2

Make sure the routing to the Virtual Private Server is properly configured. Virtual Private Servers can use the default router for your network, or you may configure the Hardware Node as rooter for its VPSs.

Failure to Log In to VPS

The Virtual Private Server 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 `vzctl set --save --userpasswd` command. For example, for Virtual Private Server 101 you might issue the following command:

```
# vzctl set 101 --save --userpasswd root:secret
```

Solution 2

Check forwarding setting by issuing the command:

```
# cat /proc/sys/ipv4/conf/venet0/forwarding
```

If it is 0 then change it to 1 by issuing the command:

```
# echo 1 > /proc/sys/ipv4/conf/venet0/forwarding
```

Problems with VPS Operation

Timeout When Accessing Remote Hosts

A host is unreachable by the OpenVZ Hardware Node or its Private Servers, though it can be reached from other computers.

Solution

Often these timeouts occur due to the fact that the Explicit Congestion Notification (ECN) mechanism of the TCP/IP protocol is on by default in OpenVZ and off in some other systems, which leads to their incompatibility. ECN is used to avoid unnecessary packet drops and for some other enhancements. If OpenVZ cannot connect to a host, turn off this mechanism:

```
# sysctl –w net.ipv4.tcp_ecn=0
net.ipv4.tcp_ecn = 0
```

CHAPTER 9

Reference

In order to make OpenVZ successfully accomplish its tasks you need to understand how to configure OpenVZ correctly. This section explains what configuration parameters OpenVZ has and how they affect its behavior.

In This Chapter

Configuring OpenVZ

In order to make OpenVZ successfully accomplish its tasks you need to understand how to configure OpenVZ correctly. This chapter explains what configuration parameters OpenVZ has and how they affect its behavior.

Matrix of OpenVZ Configuration Files

There are a number of files responsible for the OpenVZ system configuration. These files are located in the `/etc` directory on the Hardware Node; a list of these files is given below:

`/etc/sysconfig/vz`	OpenVZ global configuration file. This file keeps system-wide settings, affecting VPS and OpenVZ template default location, global network settings and so on.
`/etc/sysconfig/vz-scripts/`*`vpsid`*`.conf`	Private configuration file owned by VPS numbered *`vpsid`*. File keeps VPS specific settings – its resource management parameters, location of private area, IP address and so on.
`/etc/sysconfig/vz-scripts/ve-`*`name`*`.conf-sample`	Sample files, containing a number of default VPS configurations, which may be used as a reference for VPS creation. Following samples are shipped with OpenVZ: `light`, `vps.basic`. Also, you may create your new samples customized for your own needs.
`/etc/sysctl.conf`	Kernel parameters. You should adjust a number of kernel `sysctl` parameters stored in the `/etc/sysctl.conf` file.
`/etc/cron.d/vpsreboot`	Configuration file for the `cron` daemon. Using this file, OpenVZ emulates the "`reboot`" command working in VPS.
`/etc/sysconfig/vz-scripts/dists/`*`dist_name`*`.conf`	Configuration files used to determine what scripts are to be run on performing some operations in the VPS context (e.g. on adding a new IP address to the VPS). These scripts are different from OpenVZ action scripts and depend on the Linux version the given VPS is running.

Global OpenVZ Configuration File

OpenVZ keeps its system wide configuration parameters in the `/etc/sysconfig/vz` configuration file. This file is in shell format. Keep in mind that OpenVZ scripts source this file – thus, shell commands in this file will cause system to execute them under root account. Parameters in this file are presented in the form `PARAMETER=”value”`. Logically all the parameters belong to the following groups: global parameters, logging, disk quota, template, Virtual Private Servers, and supplementary parameters. Below is the description of all the parameters defined in this version of OpenVZ.

Global parameters

Parameter	Description	Default value
`VIRTUOZZO`	This can be either “yes” or “no”. OpenVZ System V startup script checks this parameter. If set to “no”, then OpenVZ modules are not loaded. You might set it to “no” if you want to perform system maintenance and do not want to bring up all VPSs on the Hardware Node.	`yes`
`LOCKDIR`	Actions on a Virtual Private Server should be serialized, since two simultaneous operations on the same Virtual Private Server may break its consistency. OpenVZ keeps lock files in this directory in order to serialize access to one Virtual Private Server.	`/vz/lock`
`VE0CPUUNITS`	CPU weight designated for the Hardware Node itself.	`1000`

Logging parameters affect the `vzctl` utility logging behavior.

Parameter	Description	Default value
`LOGGING`	This parameter defines whether `vzctl` should log its actions.	`Yes`
`LOGFILE`	File where `vzctl` logs its actions.	`/var/log/vzctl.log`
`LOG_LEVEL`	There are three levels of logging defined in the current version of OpenVZ.	`0`

The table below describes the possible values of the `LOG_LEVEL` parameter and their meanings:

Log level	Information to be logged
0	Actions of `vzctl` on Virtual Private Servers like `start`, `stop`, `create`, `destroy`, `mount`, `umount`.
1	Level 1 logs events, calls to `vzctl` helper scripts located in `/etc/sysconfig/vz-scripts` (such as `vz-start` and `vz-stop`) and situations when the init process of the VPS is killed on VPS stop after timeout.
2	Level 0 and level 1 logging events, plus template version used for VPS creation and calls to mount and quota operations with parameters.

Disk quota parameters allow you to control the disk usage by the Virtual Private Servers:

Parameter	Description	Default value
DISK_QUOTA	DISK_QUOTA defines whether to turn on disk quota for Virtual Private Servers. If set to “no” then disk space and inodes accounting will be disabled.	yes
VZFASTBOOT	If set to "no", disk quota is reinitialized for each VPS when the Hardware Node is booted after an incorrect shutdown, which results in a very long booting time. If set to "yes", all the VPSs are started without quota reinitialization, and then the VPSs are consecutively stopped and restarted one by one to reinitialize their quotas.	no

Template parameters allow to configure the template area location.

Parameter	Description	Default value
TEMPLATE	This is the directory where to find templates. It is not recommended to redefine this option since all the templates built by SWsoft use the default directory.	/vz/template

Virtual Private Server default parameters either affect new VPS creation, or represent VPS parameters that can be overridden in VPS configuration file:

Parameter	Description	Default value
VE_ROOT	This is a path to the VPS root directory where private area is mounted.	/vz/root/$VEID
VE_PRIVATE	This is a path to the VPS private area. OpenVZ implementation requires VE_PRIVATE reside within a single physical partition.	/vz/private/$VEID
CONFIGFILE	Default configuration file sample for VPS creation, may be overridden with the `--config` option of the `vzctl create` command.	vps.basic
DEF_OSTEMPLATE	Default OS template for VPS creation, may be overridden with the `--ostemplate` command line option for `vzctl create`.	fedora-core-4
IPTABLES	Only those `iptables` modules will be loaded to the VPSs hosted on the Node which are indicated as the value of this parameter and only if they are loaded on the Node itself as well. Here follows a list of possible modules: `ip_conntrack`, `ip_conntrack_ftp`, `ip_conntrack_irc`, `iptable_filter`, `ipt_length`, `ipt_limit`, `ipt_LOG`, `iptable_mangle`, `ipt_conntrack`, `ipt_helper`, `ipt_state`, `ipt_tcpmss`, `ipt_tos`, `ipt_multiport`, `iptable_nat`, `ip_nat_ftp`, `ip_nat_irc`, `ipt_REJECT`, `ipt_TCPMSS`, `ipt_TOS`, `ipt_ttl`.	ipt_REJECT ipt_tos ipt_limit ipt_multiport iptable_filter iptable_mangle ipt_TCPMSS ipt_tcpmss ipt_ttl ipt_length

Supplementary parameters define other OpenVZ settings:

Parameter	Description	Default value
`VZWDOG`	Defines whether the `vzwdog` module is loaded on OpenVZ startup. This module is responsible for catching messages from the kernel. It is needed in case you configure the serial Monitor Node for OpenVZ.	`no`

VPS Configuration File

Each Virtual Private Server has its own configuration file, which is stored in the `/etc/sysconfig/vz-scripts` directory and has a name like `*vpsid*.conf`. This file has the same format as the global configuration file. The settings specified in this file can be subdivided into the following categories: miscellaneous, networking, and resource management parameters.

Miscellaneous parameters:

`VERSION`	Specifies the OpenVZ version the configuration file applies to.
`ONBOOT`	Specifies whether the VPS should be started automatically on system startup. OpenVZ automatically starts all Virtual Private Servers that have this parameter set to “yes” upon startup.
`ALLOWREBOOT`	Specifies whether VPS may be restarted with “reboot” command inside. If omitted or set to “`yes`”, reboot is allowed.
`CAPABILITY`	Specifies capabilities inside of VPS. Setting of following capabilities is allowed: `CHOWN`, `AC_OVERRIDE`, `AC_READ_SEARCH`, `FOWNER`, `FSETID`, `KILL`, `SETGID`, `SETUID`, `SETPCAP`, `LINUX_IMMUTABLE`, `NET_BIND_SERVICE`, `NET_BROADCAST`, `NET_ADMIN`, `NET_RAW`, `IPC_LOCK`, `IPC_OWNER`, `SYS_MODULE`, `SYS_RAWIO`, `SYS_CHROOT`, `SYS_PTRACE`, `SYS_PACCT`, `SYS_ADMIN`, `SYS_BOOT`, `SYS_NICE`, `SYS_RESOURCE`, `SYS_TIME`, `SYS_TTY_CONFIG`, `MKNOD`, `LEASE`.
`OSTEMPLATE`	This is the name of the OS template that has been used for creating a VPS. You do not have to change this parameter; `vzctl` will set it for you upon calling the `vzctl create` *`vpsid`* `--ostemplate` *`template`* command (or using the defaults from the global configuration file).
`VE_ROOT`	Overrides the `VE_ROOT` parameter from the global configuration file.
`VE_PRIVATE`	Overrides the `VE_PRIVATE` parameter from the global configuration file.

Resource management parameters control the amount of resources a VPS can consume. They are described in the Managing Resources chapter in detail; here is only a list of parameters allowed in VPS configuration file.

All resource management parameters can be subdivided into the general, disk, and system categories for your convenience. Any parameter can be set with the `vzctl set` command and the corresponding option name (in the lower case, e.g. `--kmemsize` for `KMEMSIZE`, etc.). See the OpenVZ Command Line Interface section for more details. The Typical value column specifies a range of reasonable parameter values for different applications, from light to huge heavy loaded VPS (consuming 1/8 of hardware node with 2 GB memory). If barrier and limit fields are in use, ranges for both thresholds are given.

General and disk parameters:

Parameter	Description	Typical value
`ORIGIN_SAMPLE`	The configuration sample the VPS was based on when created.	
`CONFIG_CUSTOMIZED`	Indicates whether any of the VPS configuration parameters have been modified as regards its original configuration sample. If this parameter is omitted, its value is considered as "no".	
`CPUUNITS`	Guaranteed CPU power. This is a positive integer number, which determines the minimal guaranteed share of the CPU the Virtual Private Server receives. The total CPU power in `CPUUNITS` is its Bogomips number multiplied by 25. OpenVZ reporting tools consider one 1 GHz PIII Intel processor to be approximately equivalent to 50,000 CPU units.	`250…1000`
`CPULIMIT`	Allowed CPU power. This is a positive number indicating the share of the CPU time in per cent the VPS may never exceed. You may estimate this share as (allowed VPS CPUUNITS/CPU power)*100%.	`1…4`
`DISKSPACE`	Total size of disk space consumed by VPS, in 1 Kb blocks.	`204800…10485760 - 204800…11534340`
`DISKINODES`	Total number of disk inodes (files, directories, symbolic links) a Virtual Private Server can allocate.	`80000…400000- 88000…440000`
`QUOTATIME`	The grace period of the disk quota. It is defined in seconds. Virtual Private Server is allowed to temporarily exceed its quota soft limits for not more than the `QUOTATIME` period.	`0…604800`
`QUOTAUGIDLIMIT`	Number of user/group IDs allowed for VPS internal disk quota. If set to 0, UID/GID quota will not be enabled.	`0…500`

System parameters:

`NUMPROC`	Number of processes and threads allowed. Upon hitting this limit, VPS will not be able to start new process or thread.	`40…400`

AVNUMPROC	Number of processes expected to run in the Virtual Private Server on average. This is informational parameter used by utilities like `vzcfgvalidate` in order to ensure configuration correctness.	0…NUMPROC
NUMTCPSOCK	Number of TCP sockets (`PF_INET` family, `SOCK_STREAM` type). This parameter limits the number of TCP connections and, thus, the number of clients the server application can handle in parallel.	40…500
NUMOTHERSOCK	Number of socket other than TCP. Local (UNIX-domain) sockets are used for communications inside the system. UDP sockets are used for Domain Name Service (DNS) queries, as example. UDP and other sockets may also be used in some very special applications (SNMP agents and others)	40…500
VMGUARPAGES	Memory allocation guarantee, in pages. Applications are guaranteed to be able to allocate memory while the amount of memory accounted as `privvmpages` does not exceed the configured barrier of the `vmguarpages` parameter. Above the barrier, memory allocation is not guaranteed and may fail in case of overall memory shortage.	1725…107520
KMEMSIZE	Size of unswappable kernel memory, allocated for internal kernel structures for the processes of a particular VPS. Typical amounts of kernel memory is 16…50 Kb per process.	798720…13148160 - 851968…14024704
TCPSNDBUF	Total size of send buffers for TCP sockets – amount of kernel memory, allocated for data sent from application to TCP socket, but not acknowledged by remote side yet.	159744…5365760-262144…10458760
TCPRCVBUF	Total size of receive buffers for TCP sockets. Amount of kernel memory, received from remote side but not read by local application yet.	159744…5365760-262144…10458760
OTHERSOCKBUF	Total size of UNIX-domain socket buffers, UDP and other datagram protocols send buffers.	61440…1503232-163840…4063232
DGRAMRCVBUF	Total size of receive buffers of UDP and other datagram protocols.	32768…262144
OOMGUARPAGES	Out-of-memory guarantee, in pages. Any VPS process will not be killed even in case of heavy memory shortage if current memory consumption (including both physical memory and swap) until `oomguarpages` barrier is not reached.	1725…107520
LOCKEDPAGES	Memory not allowed to be swapped out (locked with the `mlock()` system call), in pages (one page is 4 Kb).	4…4096
SHMPAGES	Total size of shared memory (including IPC, shared anonymous mappings and `tmpfs` objects), allocated by processes of particular VPS, in pages.	512…16384
PRIVVMPAGES	Size of private (or potentially private) memory, allocated by an application. Memory that is always shared among different applications is not included in this resource parameter.	3072…151200-3450…1612800

NUMFILE	Number of files opened by all VPS processes.	512…8192
NUMFLOCK	Number of file locks created by all VPS processes.	50…200 – 60…220
NUMPTY	Number of pseudo-terminals. For example, ssh session, screen, xterm application consumes pseudo-terminal resource.	4…64
NUMSIGINFO	Number of siginfo structures (essentially this parameter limits size of signal delivery queue).	256…512
DCACHESIZE	Total size of dentry and inode structures locked in memory. As example, application, first opening the /etc/passwd file, locks entries corresponding to etc and passwd inodes. If a second application opens the /etc/shadow file – only entry corresponding to shadow is charged, because etc is charged already.	184320…3932160-196608…4194304
PHYSPAGES	Total size of RAM used by processes. This is accounting-only parameter currently. It shows the usage of RAM by VPS. For memory pages used by several different VPSs (mappings of shared libraries, for example), only a fraction of a page is charged to each VPS. The sum of the physpages usage for all VPSs corresponds to the total number of pages used in the system by all accounted users.	Not limited
NUMIPTENT	The number of IP packet filtering entries.	12…128

Network-related parameters allow you to set bandwidth management parameters, hostname and IP addresses that Virtual Private Server can use as well as to indicate those iptables modules that can be loaded to the VPS:

HOSTNAME	If this parameter is specified, then vzctl will set the hostname to its value upon the next VPS start. This parameter can be omitted. In this case, VPS administrator should configure the hostname manually.
IP_ADDRESS	This is the list of IP addresses, which can be used on VPS network interfaces. This list is an argument of the VPS start call and it is impossible to assign IP address from inside the VPS if the address is not in the list. Any IP address assigned from within VPS will be visible only within the VPS.
NAMESERVER	DNS server IP address for VPS. More than one server may be specified in space-separated format.
SEARCHDOMAIN	DNS search domains for VPS. More than one domain may be specified.
NETDEV	The names of physical network adapters that have been moved from the Hardware Node to the given VPS.
IPTABLES	Overrides the IPTABLES parameter from the global configuration file.

Managing OpenVZ Scripts

This section provides information on OpenVZ scripts used to automate and perform some operations and procedures within your system.

Overview

Along with OpenVZ configuration files responsible for the OpenVZ system configuration, there are a number of OpenVZ scripts allowing you to customize the VPS behaviour in different ways. These are the following scripts:

Script Name	Description
`/etc/sysconfig/vz-scripts/`*`VPS_ID.action`*	VPS private action scripts. These scripts allow to run user-defined actions on particular events. Currently defined actions are `start`, `stop`, `mount`, `umount`.
`/var/lib/vzctl/scripts/…`	Scripts to be executed on performing certain VPS-related operations (e.g. on adding a new IP address to the VPS). These operations should be specified in the corresponding distribution configuration file.
`/etc/rc.d/init.d/vz`	OpenVZ start/stop System V script. This script is responsible for proper OpenVZ startup and shutdown procedures, including OpenVZ modules loading and VPS start/stop procedures.

OpenVZ Action Scripts

There might be situations when you need to do additional actions when a particular VPS is started or stopped. For example, if you want to be able to access the Host OS file system (or part of it) from VPS 101, then you can bind mount it inside the VPS manually from the Host OS. However, after you restart the VPS, your mount disappears, and you should manually type the `mount` command again.

OpenVZ allows you to automate procedures like the above by using *OpenVZ action scripts*. There are six action scripts defined in this version of OpenVZ:

global `mount`	This script runs immediately after `vzctl` mounts the VPS private area. The VPS itself is not yet running and the script is running in the Host OS context.
`mount`	This script runs immediately after the global mount script. The VPS is still not running, and the scripts is called in the Host OS context.
`start`	After `vzctl` has started a VPS, it runs the VPS `start` script. The script is running already in the VPS context.
`stop`	This script runs before the VPS is stopped, in the VPS context.
`umount`	After the VPS has been already stopped, the `umount` script is executed, and the script runs in the Host OS context.
global `umount`	This script runs when `vzctl` is about to dismount the VPS private area. It also runs in the Host OS context.

The normal order of executing action scripts is shown in the figure below. The `mount` and `umount` scripts run in the context of the Host OS rather than in the Virtual Private Server.

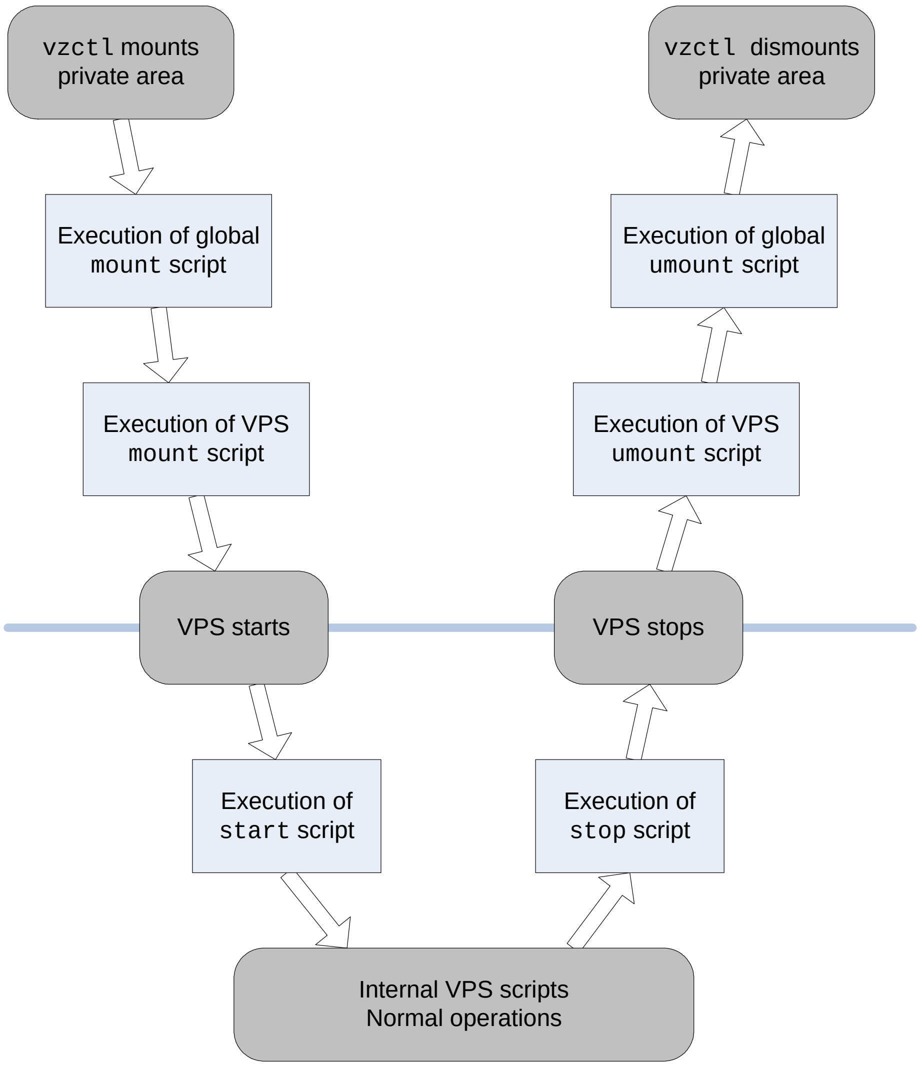

vzctl mounts private area
Execution of global mount script
Execution of VPS mount script
VPS starts
Execution of start script
Internal VPS scripts Normal operations
Execution of stop script
VPS stops
Execution of VPS umount script
Execution of global umount script
vzctl dismounts private area

Figure 6: Sequence of Executing Action Scripts

It is important to understand how `vzctl` handles exit codes of action scripts. If exit code is non-zero, then `vzctl` will try to undo the action for the `mount` and `start` scripts. In other words, if the `start` script returns an error, then `vzctl` will stop VPS, and if one of the `mount` scripts fails, then `vzctl` will dismount the VPS private area. Please note that in this case `vzctl` will not execute the `stop` and `umount` scripts at all.

Caution: When executing `vzctl start`, both `mount` and `start` scripts run. However, if the `start` script fails then neither `stop` nor `umount` scripts will run. As a result, `vzctl` might be unable to dismount the VPS private area, if you set up additional mounts in the `mount` scripts and dismount them in the `umount` scripts.

The situation with the `umount` and `stop` scripts is similar. If a script returns an error, then the action will not be taken. Be careful since this allows to create Virtual Private Servers that are not stoppable by `vzctl`.

Action scripts are located in the same directory as VPS configuration files and have names like *`vpsid.action`*. The global scripts are named `vps.mount` and `vps.umount`, and the other scripts have the corresponding VPS ID as part of their name. As the names of the global scripts are fixed, they are called when any VPS is started or stopped. That is why, in these scripts you should perform those commands that are common for all VPSs, and leave VPS-specific commands for the scripts belonging to a particular VPS. Thus, for example, for VPS 101 the action scripts will have names:

- `/etc/sysconfig/vz-scripts/vps.mount`
- `/etc/sysconfig/vz-scripts/101.mount`
- `/etc/sysconfig/vz-scripts/101.start`
- `/etc/sysconfig/vz-scripts/101.stop`
- `/etc/sysconfig/vz-scripts/101.umount`
- `/etc/sysconfig/vz-scripts/vps.umount`

For the `mount` and `umount` scripts, the environment passed is the standard environment of the parent (i.e. `vzctl`) with two additional variables: `$VEID` and `$VE_CONFFILE`. The first one holds the ID of the Virtual Private Server being mounted (started, stopped, dismounted), and the second one holds the full path to the VPS configuration file. It is probably a bit redundant. SWsoft introduced both variables for convenience. You can use the following fragment of the code in bash scripts to get access to additional VPS information like `$VE_PRIVATE` or `$VE_ROOT` locations:

```
#!/bin/bash
#
# This script source VPS configuration files in the same
# order as vzctl does

# if one of these files does not exist then something is
# really broken
[ -f /etc/sysconfig/vz ] || exit 1
[ -f $VE_CONFFILE ] || exit 1

# source both files. Note the order, it is important
. /etc/sysconfig/vz
. $VE_CONFFILE
```

The `start` and `stop` scripts are performed in the VPS context. If these scripts call any external commands, these commands are taken from VPS itself. Also note that the `start` script runs before any VPS tasks (including `init`), thus the `/proc` file system is not mounted in VPS at this moment – therefore, applications using an information from `/proc` may be not functional.

OpenVZ Command Line Interface

OpenVZ is shipped with a number of command line tools. This chapter documents the utilities, which are supported in OpenVZ. For every utility, all available command-line options and switches are described.

Matrix of OpenVZ Command Line Utilities

The table below contains the full list of OpenVZ command-line utilities.

General utilities are intended for performing day-to-day maintenance tasks:

`vzctl`	Utility to control Virtual Private Servers.
`vzlist`	Utility to view a list of VPSs existing on the Node with additional information.
`vzquota`	Utility to control OpenVZ disk quotas.

Template management tools allow template creation, maintenance and installation of applications into VPS:

`vzpkgls`	Utility to get a list of templates available on the Hardware Node and in VPSs.
`vzpkgcache`	Create/update a set of template caches.
`vzrpm`	Simple rpm wrapper to use rpm with a particular VPS.
`vzyum`	Yum wrapper to use yum with a particular VPS.

Supplementary tools perform a number of tasks and are used by other OpenVZ utilities:

`vzdqcheck`	Print file space current usage from quota's point of view.
`vzdqdump` and `vzdqload`	Utilities to dump the VPS user/group quota limits and grace times from the kernel or the quota file or for loading them to a quota file.
`vzcpucheck`	Utility for checking CPU utilization by Virtual Private Servers.
`vzmemcheck`	Utility for checking the HN and VPS current memory parameters.
`vzcalc`	Utility to calculate resource usage by a Virtual Private Server.
`vzpid`	Utility that prints Virtual Private Server id the process belongs to.
`vzsplit`	Utility to generate VPS configuration file sample, "splitting" the Hardware Node into equal parts.
`vzcfgvalidate`	Utility to validate Virtual Private Server configuration file correctness.

vzctl

`vzctl` is the primary tool for Virtual Private Server management. To use it, you have to log in to the Hardware Node as the root user. The syntax of `vzctl` is:

```
vzctl [verbosity-options] command vpsid [command-specific-options]
```

Where `command` can be one of the following:

`create`	Used to create Virtual Private Servers and base for Shared Virtual Private Servers
`destroy`	Used to destroy a Virtual Private Server
`mount`	Allows mounting Virtual Private Server private area and executing VPS mount script
`umount`	Allows dismounting Virtual Private Server private area and executing umount script
`start`	Starts a Virtual Private Server
`stop`	Stops a Virtual Private Server
`restart`	Restarts a Virtual Private Server
`status`	Displays a Virtual Private Server status
`set`	Used to set Virtual Private Server parameters, including resource control settings, location of private area, VPS hostname, IP addresses and VPS root user password
`enter`	Provides a way for hardware node administrator to “enter” a Virtual Private Server without knowing VPS root password. Use this command with caution and never run it on un-trusted Virtual Private Servers.
`exec, exec2`	These two commands allow running arbitrary commands inside a VPS without logging in to the corresponding VPS. The difference between two is the returned status.

Verbosity options can be used with any of the above commands and they are:

`--verbose`	Overrides the `LOG_LEVEL` setting from the OpenVZ global configuration file `/etc/sysconfig/vz` and sets log level to maximum possible value for this `vzctl` session.
`--quiet`	Disables logging to screen and to the log file.

vzctl create

This command is used to create a new Virtual Private Server. It has the following syntax:

```
vzctl create vpsid [--ostemplate name] [--config name]
      [--private path]  [--root path]
```

With this command, you can create Virtual Private Servers. Virtual Private Server ID `vpsid` is required for this command and shall be unique for the Hardware Node.

Note: Virtual Private Server IDs from 1 to 100 are reserved for internal OpenVZ needs. Do not use IDs from 1 to 100 for your Virtual Private Servers.

Command arguments are as follows:

`--ostemplate name`	Denotes package set (OS template) to use when creating the Virtual Private Server. If omitted, this value is taken from the global OpenVZ configuration file (`DEF_OSTEMPLATE` parameter).
`--config name`	Optional. If this option is given, `vzctl` copies the values from the sample VPS configuration file located in `/etc/sysconfig/vz-scripts` and having the name in the form of `ve-name.conf-sample`. The sample configuration files usually have a number of resource control limits for the VPS. If you skip this option and the default configuration file name is not specified in the global OpenVZ configuration file, you will have to set resource control parameters for the VPS by using the `vzctl set` command before you are able to start the VPS.
`--private path`	Optional. When used specifies path to the Virtual Private Server private area. This option is used to override default path to private area from the `/etc/sysconfig/vz` configuration file (`VE_PRIVATE` variable). The argument can contain `$VEID` string which will be replaced by numeric VPS ID value.
`--root path`	Optional. When used specifies path to the mount point of the Virtual Private Server root directory. This option is used to override default path to VPS root directory from the `/etc/sysconfig/vz` configuration file (`VE_ROOT` variable). The argument can contain `$VEID` string which will be replaced by numeric VPS ID value.

When creating a new Virtual Private Server, you should specify a unique ID for it. There are no restrictions besides uniqueness from the `vzctl` standpoint. However, it is advisable to assign different ID ranges to hardware nodes in multi-node environments. For example, you can use IDs from 101 to 2000 on the first node, IDs from 2001 to 4000 on the second one and so on. This will help you in tracking down the node where VPS was created and will eliminate possibility of VPS IDs conflicts when migrating Virtual Private Servers between Nodes.

vzctl destroy

The syntax of this command is:

```
vzctl destroy vpsid
```

This command is used to delete a Virtual Private Server, which is no longer needed. It physically removes all the files located in VPS private area (specified as `VE_PRIVATE` variable in the VPS configuration file) and renames the VPS configuration file in `/etc/sysconfig/vz-scripts/` from *`vpsid`*`.conf` to *`vpsid`*`.conf.destroyed`. It also renames VPS action scripts if any in a similar manner.

This command does not take any additional arguments and requires the Virtual Private Server to be stopped and its private area to be dismounted.

vzctl start, vzctl stop, vzctl restart, and vzctl status

These four commands have the same syntax and take no obligatory arguments:

```
vzctl start   vpsid
vzctl stop    vpsid [--fast]
vzctl restart vpsid
vzctl status  vpsid
```

The first command is used to start a Virtual Private Server. It will set up all network interfaces, initialize VPS quota, if needed, and start the `init` process inside the Virtual Private Server.

When starting a Virtual Private Server, `vzctl` can execute custom scripts located in the `/etc/sysconfig/vz-scripts` directory, namely (in order of execution):

`vpsid``.mount`	Optional Virtual Private Server mount script. If it exists then it is executed immediately after mounting VPS private area. If it exits with non-zero status then vzctl dismounts VPS private area and returns the error.
`vpsid``.start`	Optional Virtual Private Server start script. If it exists then it is executed in the context of just started VPS.

`vzctl stop` shuts the Virtual Private Server down. If the VPS is not down after a two-minute timeout due to an error in an application, for example, `vzctl` will forcibly kill all the processes inside the VPS. To avoid waiting for two minutes in case of a corrupted Virtual Private Server, you may use the `--fast` option with this command. The normal shutdown sequence of `vzctl stop` is described below in order of execution:

`vpsid``.stop`	Optional Virtual Private Server stop script. If it exists then it is executed in context of the Virtual Private Server prior to any other actions. If it exits with non-zero status then `vzctl` does not stop the VPS.
`vpsid``.umount`	Optional Virtual Private Server umount script. If it exists then it is executed after the VPS was stopped but before its private area is being dismounted.

You should use action scripts (`mount`/`umount` and `start`/`stop`) if you would like to carry out some actions upon VPS startup/shutdown.

The `vzctl restart` *`vpsid`* command consecutively performs the stopping and starting of the corresponding VPS.

The `vzctl status` *`vpsid`* command shows current VPS state. It outputs the following information: whether the VPS private area exists, whether it is mounted and whether the VPS is running as in the example below:

```
# vzctl status 101
VPS 101 exist mounted running
```

vzctl mount and vzctl umount

These commands take no additional arguments:

```
vzctl mount  vpsid
vzctl umount vpsid
```

The first command mounts the VPS private area to the VPS root directory (`/vz/root/`*`vpsid`*`/` on the Hardware Node) without starting it. Normally you do not have to use this command as the `vzctl start` command mounts the VPS private area automatically.

The `vzctl umount` command unmounts the VPS private area. Usually there is no need in using this command either, for `vzctl stop` unmounts the VPS private area automatically.

vzctl set

This command is used for setting VPS parameters. It has the following syntax:

```
vzctl set vpsid --setting_name value […] [ --save ]
```

An optional `--save` switch tells `vzctl` whether to save changes into the VPS configuration file `/etc/sysconfig/vz-scripts/`*`vpsid`*`.conf`. Practically all VPS settings can be changed dynamically without the necessity of VPS reboot. The exceptions are `--onboot`, `--quotaugidlimit`, `--capability`, `--private`, and `--root`.

The settings specified in this file can be subdivided into the following categories: miscellaneous, networking, and resource management parameters.

Miscellaneous settings:

`--onboot yes|no`	This setting requires the `--save` switch. If you set it to "yes" than OpenVZ will automatically start this Virtual Private Server on next system startup.
`--userpasswd` *`user`*`:`*`password`*	This setting creates a new user with the specified password in the VPS, or changes the password of an already existing user. This command modifies not the VPS configuration file, but the `/etc/passwd` and `/etc/shadow` files inside the VPS. In case the VPS root is not mounted, it is automatically mounted to apply the changes and then unmounted.

`--noatime yes|no`	Sets the `noatime` flag (do not update inode access times) on the VPS file system. The default is `yes` for a Class 1 VPS, and `no` otherwise.
`--devnodes` *`device`*`:r|w|rw|none`	Lets the VPS access the specified devices in the specified mode - read-only, write-only, or read-write - or denies any access. E.g.: `--devnodes hda1:rw` The device must be present in the VPS `/dev` directory, otherwise, a new device is automatically created.
`--netdev_add` *`name`*	Moves the specified network device from the Hardware Node to the given VPS. E.g.: `--netdev_add eth0`
`--netdev_del` *`name`*	Moves the specified network device from the given VPS to the Hardware Node.
`--capability` *`cap`*`:on|off`	Specifies capabilities inside of VPS. Setting of following capabilities is allowed: `AC_OVERRIDE`, `AC_READ_SEARCH`, `CHOWN`, `FOWNER`, `FSETID`, `IPC_LOCK`, `IPC_OWNER`, `KILL`, `LEASE`, `LINUX_IMMUTABLE`, `MKNOD`, `NET_ADMIN`, `NET_BIND_SERVICE`, `NET_BROADCAST`, `NET_RAW`, `SETGID`, `SETPCAP`, `SETUID`, `SYS_ADMIN`, `SYS_BOOT`, `SYS_CHROOT`, `SYS_MODULE`, `SYS_NICE`, `SYS_PACCT`, `SYS_PTRACE`, `SYS_RAWIO`, `SYS_RESOURCE`, `SYS_TIME`, `SYS_TTY_CONFIG`.
`--root` *`path`*	This setting does NOT move root mount point of your Virtual Private Server to a new path. It simply overrides the `VE_ROOT` parameter in the VPS configuration file.
`--private` *`path`*	This setting does NOT move the private area of your Virtual Private Server to a new path. It simply overrides the `VE_PRIVATE` parameter in the VPS configuration file. You should use this option only if you have manually moved VPS private area to a new place and want to update VPS configuration file.
`--setmode restart|ignore`	This option tells the utility either to restart or not restart the VPS after applying any parameters requiring that the VPS be rebooted for them to take effect.

Resource management settings control the amount of resources a VPS can consume. If the setting has bar:lim after it than this setting requires specifying both barrier and limit values separated by colons.

`--applyconfig` *`name`*	This option lets you set the resource parameters for the VPS not one by one, but by reading them from the VPS sample configuration file. All VPS sample configuration files are located in the `/etc/sysconfig/vz-scripts` directory and are named according to the following pattern: `ve-`*`name`*`.conf-sample`, so you should specify only the *`name`* part of the corresponding sample name after the `--applyconfig` option. Note that the names of sample configuration files cannot contain spaces. The `--applyconfig` option applies all the parameters from the sample file to the given VPS, except for the `OSTEMPLATE`, `VE_ROOT`, and `VE_PRIVATE` parameters, should they exist in the sample configuration file.
`--numproc` *`bar`*`:`*`lim`*	Number of processes and threads allowed. Upon hitting this limit, VPS will not be able to start new process or thread. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--numtcpsock` *`bar`*`:`*`lim`*	Number of TCP sockets (`PF_INET` family, `SOCK_STREAM` type). This parameter limits the number of TCP connections and, thus, the number of clients the server application can handle in parallel. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--numothersock` *`bar`*`:`*`lim`*	Number of socket other than TCP. Local (UNIX-domain) sockets are used for communications inside the system. UDP sockets are used for Domain Name Service (DNS) queries, for example. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--vmguarpages` *`bar`*`:`*`lim`*	Memory allocation guarantee, in pages (one page is 4 Kb). Applications are guaranteed to be able to allocate memory while the amount of memory accounted as `privvmpages` does not exceed the configured barrier of the `vmguarpages` parameter. Above the barrier, memory allocation may fail in case of overall memory shortage. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--kmemsize` *`bar`*`:`*`lim`*	Size of unswappable kernel memory (in bytes), allocated for internal kernel structures of the processes of a particular VPS. Typical amounts of kernel memory are 16-50 Kb per process.
`--tcpsndbuf` *`bar`*`:`*`lim`*	Total size (in bytes) of send buffers for TCP sockets – amount of kernel memory allocated for data sent from an application to a TCP socket, but not acknowledged by the remote side yet.
`--tcprcvbuf` *`bar`*`:`*`lim`*	Total size (in bytes) of receive buffers for TCP sockets. Amount of kernel memory received from the remote side but not read by the local application yet.

`--othersockbuf` *bar*:*lim*	Total size in bytes of UNIX-domain socket buffers, UDP and other datagram protocol send buffers.
`--dgramrcvbuf` *bar*:*lim*	Total size in bytes of receive buffers of UDP and other datagram protocols.
`--oomguarpages` *bar*:*lim*	Out-of-memory guarantee, in 4 Kb pages. Any VPS process will not be killed even in case of heavy memory shortage if the current memory consumption (including both physical memory and swap) does not reach the `oomguarpages` barrier. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--lockedpages` *bar*:*lim*	Memory not allowed to be swapped out (locked with the `mlock()` system call), in 4-Kb pages.
`--shmpages` *bar*:*lim*	Total size of shared memory (including IPC, shared anonymous mappings and `tmpfs` objects), allocated by processes of particular VPS, in 4 Kb pages.
`--privvmpages` *bar*:*lim*	Size in 4 Kb pages of private (or potentially private) memory, allocated by Virtual Private Server applications. Memory that is always shared among different applications is not included in this resource parameter.
`--numfile` *bar*:*lim*	Number of files opened by all VPS processes. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--numflock` *bar*:*lim*	Number of file locks created by all VPS processes.
`--numpty` *bar*:*lim*	Number of pseudo-terminals. For example, `ssh` session, `screen`, `xterm` application consumes pseudo-terminal resource. In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--numsiginfo` *bar*:*lim*	Number of `siginfo` structures (essentially this parameter limits size of signal delivery queue). In this version of OpenVZ, the limit shall be set to the same value as the barrier.
`--dcache` *bar*:*lim*	Total size in bytes of `dentry` and `inode` structures locked in memory. Exists as a separate parameter to impose a limit causing file operations to sense memory shortage and return an error to applications, protecting from excessive consumption of memory due to intensive file system operations.
`--cpuunits` *units*	Allowed CPU power. This is a positive integer number, which determines the minimal guaranteed share of the CPU the Virtual Private Server will receive. You may estimate this share as ((VPS CPUUNITS)/(Sum of CPU UNITS across all busy Virtual Private Servers))*100%. The total CPU power depends on CPU and OpenVZ reporting tools consider one 1 GHz PIII Intel processor to be equivalent to 50,000 CPU units.
`--cpulimit` *percent*	This is a positive number indicating the CPU time in per cent the corresponding VPS is not allowed to exceed.

`--diskspace` *`bar`*`:`*`lim`*	Total size of disk space consumed by VPS, in 1 Kb blocks. When the space used by a Virtual Private Server hits the barrier, the VPS can allocate additional disk space up to the limit during grace period specified by the `--quotatime` setting.
`--diskinodes` *`bar`*`:`*`lim`*	Total number of disk inodes (files, directories, symbolic links) a Virtual Private Server can allocate. When the number of inodes used by a Virtual Private Server hits the barrier, the VPS can create additional file entries up to the limit during grace period specified by the `--quotatime` setting.
`--quotatime` *`seconds`*	The grace period of the disk quota. It is defined in seconds. Virtual Private Server is allowed to temporary exceed barrier values for disk space and disk inodes limits for not more than the period specified with this setting.
`--quotaugidlimit` *`num`*	Number of user/group IDs allowed for the VPS second level disk quota (quota per user inside the VPS). If set to 0, UID/GID quota will not be enabled.

Network related settings allow you to set the hostname, the domain to search when a not fully qualified domain name is used, the DNS server address and the IP addresses that Virtual Private Server can use as well as to indicate those `iptables` modules that can be loaded to the VPS:

`--hostname` *`name`*	Sets the hostname to the specified name.
`--ipadd` *`addr`*	Adds IP address to the list of IP addresses the Virtual Private Server can use and brings up network interface with this address inside the VPS.
`--ipdel` *`addr`*`|all`	Allows you to revoke IP address from the Virtual Private Server. If “all” is used instead of IP address than all IP addresses will be revoked.
`--nameserver` *`addr`*	DNS server IP address for VPS. More than one server may be specified in space-separated format.
`--searchdomain` *`domain`*	DNS search domains for VPS. More than one domain may be specified.
`--iptables` *`module`*	Only those `iptables` modules will be loaded to the given VPS which are indicated. Here follows a list of possible modules: `ip_conntrack`, `ip_conntrack_ftp`, `ip_conntrack_irc`, `iptable_filter`, `ipt_length`, `ipt_limit`, `ipt_LOG`, `iptable_mangle`, `ipt_conntrack`, `ipt_helper`, `ipt_state`, `ipt_tcpmss`, `ipt_tos`, `ipt_multiport`, `iptable_nat`, `ip_nat_ftp`, `ip_nat_irc`, `ipt_REJECT`, `ipt_TCPMSS`, `ipt_TOS`, `ipt_ttl`.

vzctl exec, vzctl exec2, and vzctl enter

These commands are used to run arbitrary commands inside a Virtual Private Server being authenticated as root on the Hardware Node. The syntax of these commands is as follows:

```
vzctl exec|exec2 vpsid command
vzctl enter vpsid
```

where `command` is a string to be executed in the Virtual Private Server. If `command` is specified as "`-`" then the commands for execution will be read from the standard input until the end of file or "exit" is encountered.

The difference between `exec` and `exec2` is the exit code. `vzctl exec` returns 0 in case `vzctl` has been able to launch the command and does not take into account the exit code of the command itself. `vzctl exec2` returns the exit code of the command executed in the Virtual Private Server.

When using `exec` or `exec2`, you should remember that the shell parses the command line and, if your command has shell meta-characters in it, you should escape or quote them.

`vzctl enter` is similar to `vzctl exec /bin/bash`. The difference between the two is that `vzctl enter` makes the shell interpreter believe that it is connected to a terminal. As such, you receive a shell prompt and are able to execute multiple commands as if you were logged in to the Virtual Private Server.

However, be aware that `vzctl enter` is a potentially dangerous command if you have un-trusted users inside the Virtual Private Server. Your shell will have its file descriptors accessible for the VPS root in the `/proc` filesystem and a malicious user could run `ioctl` calls on it. Never use `vzctl enter` for Virtual Private Servers you do not trust. That is why, `vzctl enter` is only supposed to be an off-duty way of connecting to VPSs, not a complete replacement of `ssh`. Therefore, it has certain limitations, for example, you cannot establish `ssh` connections while being connected to a VPS through `vzctl enter`.

vzlist

The `vzlist` utility is used to list the VPSs existing on the given Hardware Node together with additional information about these VPSs. The output and sorting of this information can be customized as needed. The utility has the following syntax:

```
vzlist [-a] [-S] [-o parameter[.specifier] \
[,parameter[.specifier]...]] [-s [-]parameter[.specifier]] \
[-H] [-h hostname_pattern] [vpsid ...]

vzlist –L|--list
```

Here follows the description of available options:

Option	Description
`-a, --all`	List all the VPSs existing on the Node. By default, only running VPSs are shown.
`-S, --stopped`	List only stopped VPSs.

`-o` *`parameter`*`[.`*`specifier`*`]`	This option is used to display only particular information about the VPSs. The parameters and their specifiers that can be used after the `-o` option are listed in the following subsection. To display a number of parameters in a single output, they should be separated with commas, as is shown in the synopsis above.
`-s, --sort` `[-]`*`parameter`*`[.`*`specifier`*`]`	Sort the VPSs in the list by the specified parameter. If "`-`" is given before the name of the parameter, the sorting order is reversed.
`-h, --hosthame` *`hostname_pattern`*	Display only those VPSs that correspond to the specified hostname pattern. The following wildcards can be used: `*`,`?`, and `[]`. Note: the last wildcard should be escaped, to avoid shell interpretation.
`-H, --no-header`	Do not display column headers.
`vpsid`	Display only the VPS with the specified ID. Several VPS IDs separated with a space can be specified. If `-1` is given as the VPS ID, the utility lists only IDs of the VPSs existing on the Node, with no additional information.
`-L, --list`	List all the parameters available to be used with the `-o` option.

vzlist Output Parameters and Their Specifiers

Almost any parameter that can be used after the `-o` and `-s` switches of the `vzlist` utility can be specified by the "dot+letter" combination following the parameter and denoting one of the following things:

Specifier	Description
`.m`	The maximal registered usage of the corresponding resource by the given VPS.
`.b`	The barrier on using the corresponding resource set for the given VPS.
`.l`	The limit on using the corresponding resource set for the given VPS.
`.f`	The number of times the system has failed to allocate the corresponding resource for the given VPS.
`.s`	The soft limit on using the corresponding resource set for the given VPS.
`.h`	The hard limit on using the corresponding resource set for the given VPS.

To learn more about barriers/limits and soft/hard limits on resources, you may turn to the Managing Resources chapter.

The following parameters are available for using with the utility:

Parameter	Possible Specifiers	Output Column	Description
`vpsid`	none	`VEID`	The VPS ID.
`hostname`	none	`HOSTNAME`	The VPS hostname.
`ip`	none	`IP_ADDR`	The VPS IP address.
`status`	none	`STATUS`	Specifies whether the VPS is running or stopped.

`kmemsize`	`.m, .b, .l, .f`	`KMEMSIZE`	Size of unswappable kernel memory (in bytes), allocated for internal kernel structures of the processes of a particular VPS. Typical amounts of kernel memory are 16…50 Kb per process.
`lockedpages`	`.m, .b, .l, .f`	`LOCKEDP`	Memory not allowed to be swapped out (locked with the `mlock()` system call), in 4-Kb pages.
`privvmpages`	`.m, .b, .l, .f`	`PRIVVMP`	Size in 4 Kb pages of private (or potentially private) memory, allocated by Virtual Private Server applications. Memory that is always shared among different applications is not included in this resource parameter.
`shmpages`	`.m, .b, .l, .f`	`SHMP`	Total size of shared memory (including IPC, shared anonymous mappings and `tmpfs` objects), allocated by processes of particular VPS, in 4 Kb pages.
`numproc`	`.m, .b, .l, .f`	`NPROC`	Number of processes and threads allowed.
`physpages`	`.m, .b, .l, .f`	`PHYSP`	Total size of RAM used by processes. This is accounting-only parameter currently. It shows the usage of RAM by VPS. For memory pages used by several different VPSs (mappings of shared libraries, for example), only a fraction of a page is charged to each VPS. The sum of the `physpages` usage for all VPSs corresponds to the total number of pages used in the system by all accounted users.
`vmguarpages`	`.m, .b, .l, .f`	`VMGUARP`	Memory allocation guarantee, in pages (one page is 4 Kb). Applications are guaranteed to be able to allocate memory while the amount of memory accounted as `privvmpages` does not exceed the configured barrier of the `vmguarpages` parameter. Above the barrier, memory allocation may fail in case of overall memory shortage.
`oomguarpages`	`.m, .b, .l, .f`	`OOMGUARP`	Out-of-memory guarantee, in 4 Kb pages. Any VPS process will not be killed even in case of heavy memory shortage if the current memory consumption (including both physical memory and swap) does not reach the `oomguarpages` barrier.
`numtcpsock`	`.m, .b, .l, .f`	`NTCPSOCK`	Number of TCP sockets (`PF_INET` family, `SOCK_STREAM` type). This parameter limits the number of TCP connections and, thus, the number of clients the server application can handle in parallel.
`numflock`	`.m, .b, .l, .f`	`NFLOCK`	Number of file locks created by all VPS processes.

`numpty`	`.m, .b, .l, .f`	`NPTY`	Number of pseudo-terminals. For example, `ssh` session, `screen`, `xterm` application consumes pseudo-terminal resource.
`numsiginfo`	`.m, .b, .l, .f`	`NSIGINFO`	Number of `siginfo` structures (essentially this parameter limits size of signal delivery queue).
`tcpsndbuf`	`.m, .b, .l, .f`	`TCPSNDB`	Total size (in bytes) of send buffers for TCP sockets – amount of kernel memory allocated for data sent from an application to a TCP socket, but not acknowledged by the remote side yet.
`tcprcvbuf`	`.m, .b, .l, .f`	`TCPRCVB`	Total size (in bytes) of receive buffers for TCP sockets. Amount of kernel memory received from the remote side but not read by the local application yet.
`othersockb`	`.m, .b, .l, .f`	`OTHSOCKB`	Total size in bytes of UNIX-domain socket buffers, UDP and other datagram protocol send buffers.
`dgramrcvbuf`	`.m, .b, .l, .f`	`DGRAMRCVB`	Total size in bytes of receive buffers of UDP and other datagram protocols.
`nothersock`	`.m, .b, .l, .f`	`NOTHSOCK`	Number of socket other than TCP. Local (UNIX-domain) sockets are used for communications inside the system. UDP sockets are used for Domain Name Service (DNS) queries, for example.
`dcachesize`	`.m, .b, .l, .f`	`DCACHESIZE`	Total size in bytes of `dentry` and `inode` structures locked in memory. Exists as a separate parameter to impose a limit causing file operations to sense memory shortage and return an error to applications, protecting from excessive consumption of memory due to intensive file system operations.
`numfile`	`.m, .b, .l, .f`	`NFILE`	Number of files opened by all VPS processes.
`numiptent`	`.m, .b, .l, .f`	`NIPTENT`	The number of IP packet filtering entries.
`diskspace`	`.s, .h`	`DQBLOCKS`	Total size of disk space consumed by VPS, in 1 Kb blocks. When the space used by a Virtual Private Server hits the barrier, the VPS can allocate additional disk space up to the limit during grace period.
`diskinodes`	`.s, .h`	`DQINODES`	Total number of disk inodes (files, directories, symbolic links) a Virtual Private Server can allocate. When the number of inodes used by a Virtual Private Server hits the barrier, the VPS can create additional file entries up to the limit during grace period.
`laverage`	none	`LAVERAGE`	The average number of processes ready to run during the last 1, 5 and 15 minutes.

`cpulimit`	none	`CPULIM`	This is a positive number indicating the CPU time in per cent the corresponding VPS is not allowed to exceed.
`cpuunits`	none	`CPUUNI`	Allowed CPU power. This is a positive integer number, which determines the minimal guaranteed share of the CPU the Virtual Private Server will receive. You may estimate this share as ((VPS CPUUNITS)/(Sum of CPU UNITS across all busy Virtual Private Servers))*100%. The total CPU power depends on CPU and OpenVZ reporting tools consider one 1 GHz PIII Intel processor to be equivalent to 50,000 CPU units.

If a parameter that can be used with a specifier is used without any specifier in the command line, the current usage of the corresponding resource is shown by default.

vzquota

This command is used to configure and see disk quota statistics for Virtual Private Servers. `vzquota` is also used to turn on the possibility of using per-user/group quotas inside the VPS. It allows you to configure per-user or per-group quota inside the Virtual Private Server as well. `vzctl` uses `vzquota` internally to configure quotas and you usually do not have to use `vzquota` except for checking current quota statistics. The syntax of `vzquota` command is as follows:

```
vzquota [options] command vpsid [command-options]
```

General options available to all `vzquota` commands are:

`-v`	Verbose mode. Causes `vzquota` to print debugging messages about its progress. You can give up to two `–v` switches to increase verbosity.
`-q`	Quiet mode. Causes all warning and diagnostic messages to be suppressed. Only fatal errors are displayed.

OpenVZ quota works on a file system sub-tree or area. If this area has additional file systems mounted to its subdirectories quota will not follow this mount points. When you initialize quota, you specify the file system sub-tree starting point for the quota. Quota keeps its current usage and settings for a Virtual Private Server in the `/var/vzquota/quota.`*`vpsid`* file.

Quota file has a special flag, which indicates whether the file is “dirty”. File is dirty when its content can be inconsistent with that of real quota usage. On VPS startup, quota will be re-initialized if the Hardware Node was incorrectly brought down (for example power switch was hit). This operation may noticeably increase node startup time.

For both disk usage and inodes usage OpenVZ allows setting soft and hard limits as well as an expiration time. Upon reaching a soft limit OpenVZ starts expiration time counter. When the time is expired, the quota will block the subsequent disk space or inode allocation requests. Hard limit cannot be exceeded.

`vzquota` understands the following commands:

`init`	Before you can use quota the current disk space and inode usage should be counted. For the `init` command, you must specify all the limits as well as the file tree where you want to initialize the quota.
`drop`	Forget about given quota ID, dropping existent quota file.
`on`	Turns on quota accounting on the specified quota ID.
`off`	Turns off quota accounting on the specified quota ID.
`setlimit`	Allows changing the quota limits for the running quota.
`stat`	Shows quota statistics for the running quota.
`show`	Shows quota usage from quota file.

vzquota init

This command is used for counting current usage of disk space and inodes. It has the following syntax:

```
vzquota [options] init vpsid [command-options]
```

The following options are understood by the `vzquota init` command:

`-s, --sub-quotas 1|0`	Optional. If the value used is 1 than per user/group quota is enabled in the Virtual Private Server. By default user/group quotas are disabled.
`-b, --block-softlimit` *`num`*	Required. Disk quota block soft limit – amount of 1 Kb blocks allowed for the Virtual Private Server to use. This limit can be exceeded by the VPS for the time specified by block expiration time (see below). When expiration time is off, the Virtual Private Server cannot allocate more disk space even if the hard limit is not yet reached.
`-B, --block-hardlimit` *`num`*	Required. Specifies disk quota block hard limit in 1 Kb blocks. This limit cannot be exceeded by the Virtual Private Server.

`-e, --block-exptime` *`time`*	Required. Expiration time for excess of the block soft limit. Time can be specified in two formats: ▪ `dd:hh:mm:ss` For example: 30 - 30 seconds; 12:00 - 12 minutes; 20:15:11:00 - 20 days, 15 hours, 11 minutes ▪ `xx`A, where A - h/H(hour); d/D(day); w/W(week); m/M(month); y/Y(year). For instance: 7D - 7 days; 01w - 1 week; 3m – 3 months
`-i, --inode-softlimit` *`num`*	Required. Inodes soft limit – amount of inodes allowed for the Virtual Private Server to create. This limit can be exceeded by the VPS for the time specified by inode expiration time (see below). When expiration time is off the Virtual Private Server cannot create more inodes even if hard limit is not yet reached.
`-I, --inode-hardlimit` *`num`*	Required. Specifies inodes hard limit. This limit cannot be exceeded by the Virtual Private Server.
`-n, --inode-exptime` *`time`*	Required. Expiration time for excess of the inode soft limit. Time can be specified in two formats: ▪ `dd:hh:mm:ss` For example: 30 - 30 seconds; 12:00 - 12 minutes; 20:15:11:00 - 20 days, 15 hours, 11 minutes ▪ `xx`A, where A - h/H(hour); d/D(day); w/W(week); m/M(month); y/Y(year). For instance: 7D - 7 days; 01w - 1 week; 3m – 3 months
`-p` *`path`*	Required. Specifies the path to the Virtual Private Server private area.
`-c` *`quota_file`*	Optional. Specifies the file to write output of counted disk space and inodes as well as limits. If omitted, the default `/var/vzquota/quota.`*`vpsid`* file is used.

vzquota drop

Removes the quota file. The syntax of this command is:

```
vzquota [options] drop vpsid [-f] [-c quota_file]
```

The command checks whether the quota is running for a given Virtual Private Server and if it is, exits with error. An optional `–f` switch can be given to override this behavior and drop quota even if it is running. You can also override the path to the quota file to be dropped with an optional `–c` switch.

vzquota on and vzquota off

These commands are used to turn quota on and off. Their syntax is as follows:

```
vzquota [options] on vpsid [command-options]
vzquota [options] off vpsid [-f] [-c quota_file]
```

`vzquota off` turns the quota off for the file system tree specified in quota file given with an optional `–c` switch. If this switch is omitted, the default `/var/vzquota/quota.`*`vpsid`* file is used. This command exits with error if for some reason quota file cannot be accessed and usage statistics could be lost. You can override this behavior by giving an optional `–f` switch.

`vzquota on` accepts the following options:

`-s, --sub-quotas 1|0`	Optional. If the value used is 1 then per user/group quota is enabled in the Virtual Private Server. By default user/group quotas are disabled.
`-u, --ugid-limit` *`num`*	Optional. Specifies the maximum number of user and group IDs for which usage statistics will be counted in this VPS. If this value is 0, user/group quota will not be accounted. The default value is 0.
`-p` *`path`*	Required. Specifies the path to the Virtual Private Server private area.
`-f`	This option forces recalculation of quota usage even if the quota file does not have dirty flag set on.
`-c` *`quota_file`*	Optional. Specifies the file to write output of counted disk space and inodes as well as limits. If omitted, the default `/var/vzquota/quota.`*`vpsid`* file is used.

`-b, --block-softlimit` *`num`*
`-B, --block-hardlimit` *`num`*
`-e, --block-exptime` *`time`*
`-i, --inode-softlimit` *`num`*
`-I, --inode-hardlimit` *`num`*
`-n, --inode-exptime` *`time`*
	These options are optional for the `vzquota on` command. They are described in the vzquota init subsection.

vzquota setlimit

This command updates limits for the running quota. It requires at least one limit to be specified. It also updates the corresponding quota file with new settings. The syntax of this command is:

```
vzquota [options] setlimit vpsid [command-options]
```

Command options can be:

`-u, --ugid-limit` *`num`*	Optional. Specifies the maximum number of user and group IDs for which usage statistics will be counted in this VPS. If this value is 0, user/group quota will not be accounted. Default value is 0.

`-b, --block-softlimit` *`num`*
`-B, --block-hardlimit` *`num`*
`-e, --block-exptime` *`time`*
`-i, --inode-softlimit` *`num`*
`-I, --inode-hardlimit` *`num`*
`-n, --inode-exptime` *`time`*
	These options are optional for the `vzquota on` command. However, at least one of these options or `-u`, `--ugid-limit` *`num`* must be specified. These options are described in the vzquota init subsection.

`-c` *`quota_file`*	Optional. Specifies the file to write output of counted disk space and inodes as well as limits. If omitted, the default `/var/vzquota/quota.`*`vpsid`* file is used.

vzquota stat and vzquota show

These commands are used for querying quota statistics. The syntax is as below:

```
vzquota [options] show vpsid [-t] [-f] [-c quota_file]
vzquota [options] stat vpsid [-t] [-c quota_file]
```

The difference between the `vzquota stat` and `vzquota show` commands is that the first one reports usage from the kernel while the second one reports usage as written in the quota file. However, by default `vzquota stat` updates the file with the last kernel statistics. If you do not want to update the quota file, add the `–f` switch to the command.

You can specify an alternative location to the quota file with the `–c` *`quota_file`* switch. Otherwise, the default `/var/vzquota/quota.`*`vpsid`* file will be used.

To add information on user/group quota to the above commands output, use the `–t` command line switch.

A typical output of the `vzquota stat` command is shown below:

```
# vzquota stat 101 -t
   resource          usage       softlimit      hardlimit    grace
  1k-blocks         113856         2097152        2097152
     inodes          42539          200000         220000
User/group quota: on,active
Ugids: loaded 33, total 33, limit 100
Ugid limit was exceeded: no

User/group grace times and flags:
 type block_exp_time inode_exp_time  hex_flags
 user                                        0
group                                        0

User/group objects:
 type    ID  resource     usage   softlimit   hardlimit   grace status
 user     0 1k-blocks    113672           0           0         loaded
 user     0    inodes     42422           0           0         loaded
```

This output is suppressed for the sake of simplicity. As can be seen, Virtual Private Server 101 has the same soft and hard limits for disk space and VPS can occupy up to 2 Gb of disk space. Current usage is 113 Mb. There are 42,539 inodes used by the VPS, it has soft limit of 200,000 inodes and hard limit is set to 220,000. The empty grace column shows that grace period is started neither for inodes nor for disk space.

Per user/group quota is turned on and up to 100 users and groups are counted by the quota. Currently there are 33 users and groups found in the VPS and statistics for root is shown. There are no limits set from within the Virtual Private Server and current usage for root is 42,422 inodes and 113 Mb of disk space.

Template Management Utilities

A *template* is basically a set of packages from some Linux distribution used to populate a VPS. An OS template consists of system programs, libraries, and scripts needed to boot up and run the system (VPS), as well as some very basic applications and utilities. Applications like a compiler and an SQL server are usually not included into an OS template.

A *template cache* is an OS template installed into a VPS and then packed into a gzipped tar archive. This allows to greatly sped up the creation of a new Virtual Private Server: instead of installing all the packages comprising a Linux distribution, `vzctl` just unpacks the archive.

Template metadata are a set of files containing the information needed to recreate the template cache. It contains the following information:

- List of packages this template comprises
- Locations of (network) package repositories
- Scripts needed to be executed on various stages of template installation
- Public GPG key(s) needed to check signatures of packages
- Additional OpenVZ-specific packages

vzpkgls

This utility lists templates installed on the Hardware Node or already installed into a particular VPS. It has the following syntax:

```
vzpkgls [-c|--cached]
vzpkgls vpsid
```

If you specify a Virtual Private Server ID to this command, it lists templates applied to the Virtual Private Server. Without the *`vpsid`* argument, the utility lists templates available for Virtual Private Servers on the Hardware Node. Other options available to the `vzpkgls` command are listed below:

`-c, --cached`	This option has no effect if the *`vpsid`* argument is given. If used for listing templates available on the Hardware Node this option makes `vzpkgls` to omit OS templates for which cache was not created by running `vzpkgcache`. In other words, with this option on, `vzpkgls` will list only the templates ready to be used for Virtual Private Servers.

vzpkgcache

This utility creates/updates template caches for OS templates. You should run this utility before you can use a newly installed OS template for creating Virtual Private Servers. It has the following syntax:

```
vzpkgcache [-f|--force] [osname ...]
vzpkgcache -r|--remove osname [...]
```

This utility checks the metadata for all the templates installed on the Hardware Node and if it finds an OS template for which no cache exists, it starts downloading and installing all packages listed in the configuration file and creates a cache at the end. In case a cache already exists, the utility updates it, i.e. installs all the updated packages that have been issued since the cache was created or updated last time.

If you want to create or update specific OS template(s), supply their name(s) on the command line.

Normally you run `vzpkgcache` without any options. However, it understands the following options:

`-r, --remove` *`osname`* `[…]`	Remove the cache for the templates specified in the command line (*`osname`*). This option requires an explicit list of templates, i.e. there is no default action to remove all caches.

vzrpm

This utility acts as a simple RPM wrapper to be used for a specific VPS. It has the following syntax:

```
vzrpm vpsid [rpm_argument ...]
```

This utility runs `rpm` package manager tool for a given VPS, passing all further options to `rpm`.

vzyum

This utility acts as a wrapper for the `yum` package manager utility to be used for a specific VPS. It has the following syntax:

```
vzyum vpsid [yum_argument ...]
```

This utility runs the `yum` package manager tool for a given VPS with all the given options. It also supplies `yum` with arguments specifying the proper repository locations for the distribution this VPS is based on, and makes `yum` use the single repository cache residing on the Hardware Node.

Supplementary Tools

vzdqcheck

This utility counts inodes and disk space used using the same algorithm as OpenVZ quota. It has the following syntax:

```
vzdqcheck [options] path
```

The command traverses directory tree given as the *path* argument and calculates space occupied by all files and number of inodes. The command does not follow mount points.

Options available to the `vzdqcheck` command are:

`-h`	Usage info.
`-V`	Vzquota version info.
`-v`	Verbose mode.
`-q`	Quiet mode.

vzdqdump and vzdqload

The `vzdqdump` and `vzdqload` utilities are used for dumping the VPS user/group quota limits and grace times from the kernel or the quota file or for loading them to a quota file, respectively. `vzdqdump` displays the corresponding values on the console screen, and `vzdqload` gets the information from the standard input.

The syntax of the commands is the following:

```
vzdqdump [general_options] quota_id [-f] [-c quota_file] –G|-U|-T
vzdqload [general_options] quota_id [-c quota_file] –G|-U|-T
```

The general options are described in the table below:

`-h`	Usage info.
`-V`	vzquota version info.
`-v`	Verbose mode.
`-q`	Quiet mode.

The *quota_id* parameter corresponds to the ID of the Virtual Private Server for which you wish to dump/load the quotas. Other options are the following:

`-f`	Dump the user/group quota information from the kernel rather than from the quota file
`-c` *quota_file*	Specifies a quota file to process other than the default quota file (`/var/vzquota/quota.`*vpsid*)
`-G, --grace`	Dump/load user/group grace times
`-U, --limits`	Dump/load user/group disk limits
`-T, --exptimes`	Dump/load user/group expiration times

Quotas must be turned off when the `vzdqload` utility is working. Mind that only 2nd-level disk quotas are handled by the utilities.

vzcpucheck

This utility displays the current Hardware Node utilization in terms of allocated CPU units as well as total hardware node CPU units capacity. It has the following syntax:

```
vzcpucheck [-v]
```

Without arguments, the utility prints the sum of CPU units of all running Virtual Private Servers and total Hardware Node capacity. If the `–v` option is given, the utility prints per Virtual Private Server CPU units information.

vzmemcheck

This utility shows the Node memory parameters: low memory utilization, low memory commitment, RAM utilization, memory+swap utilization, memory+swap commitment, allocated memory utilization, allocated memory commitment, allocated memory limit. It has the following syntax:

```
vzmemcheck [-v] [-A]
```

The following options can be specified in the command line:

`-v`	Display information for each VPS.
`-A`	Display absolute values (in megabytes).

It is possible to use any of the available options, both of them, or to do without any options.

vzcalc

This utility is used to calculate Virtual Private Server resource usage. It has the following syntax:

```
vzcalc [-v] vpsid
```

This utility displays what part of Hardware Node resources Virtual Private Server *`vpsid`* is using. An optional `–v` switch produces verbose output including number of processes, low memory, allocated memory and memory and swap statistics.

For stopped Virtual Private Servers the utility displays promised and maximum values the VPS can consume. For running Virtual Private Servers, it also outputs current values.

The high values of resource usage means that either Hardware Node is overcommitted or Virtual Private Server configuration is invalid.

vzpid

This utility prints Virtual Private Server id given process id (pid) number. It has the following syntax:

```
vzpid pid [pid …]
```

Multiple process ids can be specified as arguments. The utility will print Virtual Private Server number for each of the processes.

vzsplit

This utility is used to generate a sample VPS configuration file with a set of system resource control parameters. The syntax of this command is as follows:

```
vzsplit [-n num] [-f sample_name] [-s swap_size]
```

This utility is used for dividing Hardware Node into equal parts. It generates a full set of Virtual Private Servers system resource control parameters based on the total physical memory of the Hardware Node it runs on and the number of Virtual Private Servers the Hardware Node shall be able to run even if the given number of Virtual Private Servers consume all allowed resources.

Without any option the utility prompts for the desired number of Virtual Private Servers and outputs the resulting resource control parameters to the screen.

The utility accepts the following options:

`-n` *`num`*	Desired number of Virtual Private Servers to be simultaneously run on the Hardware Node.
`-f` *`sample_name`*	Name of the sample configuration to create.
`-s` *`swap_size`*	Size of the swap file on the Node. It is recommended to specify the swap size to be taken into account when the utility generates sample configurations.

The resulting sample configuration will be created in the `/etc/sysconfig/vz-scripts` directory. The file name will be `ve-`*`sample_name`*`.conf-sample`. Now you can use *`sample_name`* as an argument to the `--config` option of the `vzctl create` command. If a sample with this name already exists, the utility will output an error message and will not overwrite the existing configuration.

vzcfgvalidate

This utility is used to check resource management parameters consistency in the Virtual Private Server configuration file. It has the following syntax:

```
vzcfgvalidate vps_config_file
```

The utility has a number of constraints according to which it tests the configuration file. If a constraint is not satisfied utility prints a message with its severity status. Three severity statuses are thus defined in OpenVZ:

Recommendation	This is a suggestion, which is not critical for Virtual Private Server or Hardware Node operations. The configuration is valid in general; however, if the system has enough memory, it is better to increase the settings as advised.
Warning	A constraint is not satisfied and the configuration is invalid. Applications in a Virtual Private Server with such invalid configuration may have suboptimal performance or fail in a not graceful way.
Error	An important constraint is not satisfied and the configuration is invalid. Applications in a Virtual Private Server with such invalid configuration have increased chances to fail unexpectedly, to be terminated or to hang.

It is suggested to use this utility when applications in a Virtual Private Servers behave in unexpected way and there seem to be no resource shortage for the Virtual Private Server.

Glossary

Hardware Node (or *Node*) is a computer where OpenVZ is installed for hosting Virtual Private Servers.

HN is an abbreviation of *Hardware Node*.

Host Operating System (or *Host OS*) is an operating system installed on the *Hardware Node*.

MAC address stands for Media Access Control address, a hardware address that uniquely identifies each Node in a network. The MAC layer interfaces directly with the network media. Consequently, each different type of network media requires a different MAC layer.

mlock, *mlock'ed page* — `mlock()` (short for memory locking) is a system call. It disables paging out for a specific region of memory. `mlock`'ed pages are guaranteed to stay resident in RAM until they are unlocked by `munlock()` system call. There are two primary applications of memory locking — the real–time applications and high–security data processing. The former require the deterministic response. The latter needs to protect valuable data from paging out into the swap file.

OS template (or *Operating System template*) is used to create new Virtual Private Servers with a preinstalled operating system. See also *Template*.

Package set is a synonym for *Template*.

Private area is a part of the file system where *VPS* files that are not shared with other *Virtual Private Servers* are stored.

siginfo structure (or just *siginfo*) is a block of information about signal generation. If a process catches a signal, it may receive siginfo telling why the system generated that signal. If a process monitors its children, it may receive *siginfo* telling why a child has changed its state. In either case, the system returns the information in a structure of the *siginfo_t* type, which includes the following information: signal number, error number, and signal code.

SSH is an abbreviation of Secure Shell. It is a protocol for logging on to a remote machine and executing commands on that machine. It provides secure encrypted communications between two untrusted hosts over an insecure network.

TCP (*TCP/IP*) stands for Transmission Control Protocol/Internet Protocol. This suite of communications protocols is used to connect hosts on the Internet.

Template (or *package set*) is a set of original distribution packages intended to be installed to a Virtual Private Server.

UBC is an abbreviation of *User Beancounter*.

User Beancounter (or *UBC*) is the subsystem of OpenVZ for managing VPS memory and some system-related resources.

VPS is an abbreviation of *Virtual Private Server*.

VENET device is a virtual networking device, a gateway from a *VPS* to the external network.

Virtual Private Server (or *VPS*) is a virtual private server, which is functionally identical to an isolated standalone server, with its own IP addresses, processes, files, its own users database, its own configuration files, its own applications, system libraries, and so on. Virtual Private Servers share one *Hardware Node* and one OS kernel. However, they are isolated from each other. Virtual Private Server is a kind of 'sandbox' for processes and users. Virtual Private Server 0 is used to designate the Hardware Node itself.

OpenVZ is a complete server automation and virtualization solution allowing you to create multiple isolated *Virtual Private Servers* on a single physical server to share hardware, licenses, and management effort with maximum efficiency.

Index