Parallels

Parallels Server 4 Bare Metal

User's Guide

Parallels™

Contents

Managing Resources   87

Managing Services and Processes   125

Troubleshooting   194

Glossary   208

Index   210

CHAPTER 1

Introduction

Parallels Server 4 Bare Metal is a virtualization solution that combines the benefits provided by Parallels Server 3.0 with those present in Parallels Virtuozzo Containers 4.0 for Linux. Using Parallels Server Bare Metal, you can run both virtual machines and Containers on the same server.

In This Chapter

About This Guide

The *Parallels Server 4 Bare Metal User's Guide* provides comprehensive information on Parallels Server 4 Bare Metal - high-end virtualization software for bare metal servers. It covers the necessary theoretical conceptions as well as practical aspects of working with Parallels Server Bare Metal. The guide will familiarize you with the way to create and administer virtual machines and Containers using the Parallels command line interface.

Note: The guide does not explain how to install and configure your Parallels Server 4 Bare Metal system. For detailed information on these operations, see the *Parallels Server 4 Bare Metal Installation Guide*.

The primary audience for this guide is anyone responsible for administering one or more systems running Parallels Server 4 Bare Metal. We assume that you have some familiarity with how to work in the Linux command line.

Organization of This Guide

This guide is organized in the following way:

- Chapter 1, Introduction, gives an overview of the Parallels Server Bare Metal product and this guide.
- Chapter 2, Parallels Server Bare Metal Basics, explains the general principles of Parallels Server Bare Metal operation.
- Chapter 3, Operations on virtual machines and Containers, covers those operations that you can perform on a virtual machine and Container: creating and deleting virtual machines and Containers, starting and stopping them, backing up and restoring, etc. You will also learn how to perform different kinds of migration: migrate virtual machines and Containers between Parallels servers, migrate a physical server to a virtual machine and Container, and migrate a Container to a virtual machine.
- Chapter 4, Managing Resources, focuses on configuring and monitoring the resource control parameters for virtual machines and Containers. These parameters comprise disk quotas, network accounting and shaping, CPU and system resources.
- Chapter 5, Managing Services and Processes, familiarizes you with the operations you can perform on processes and services in Parallels Server Bare Metal.
- Chapter 6, Managing Parallels Server Bare Metal Network, familiarizes you with the Parallels Server Bare Metal network structure and explains how to manage networks in Parallels Server Bare Metal systems.
- Chapter 7, Managing Licenses, provides detailed information on managing licenses in Parallels Server Bare Metal.
- Chapter 8, Keeping Your System Up To Date, informs you of the ways to keep all the software components of a Parallels server up to date.
- Chapter 9, Advanced Tasks, enumerates those tasks that are intended for advanced system administrators who would like to obtain deeper knowledge about Parallels Server Bare Metal capabilities.
- Chapter 10, Troubleshooting, suggests ways to resolve common inconveniences should they occur during your work with Parallels Server Bare Metal.

Documentation Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it.

The table below presents the existing formatting conventions.

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 Resources tab.
	Titles of chapters, sections, and subsections.	Read the Basic Administration chapter.

Italics	Used to emphasize the importance of a point, to introduce a term or to designate a command-line placeholder, which is to be replaced with a real name or value.	These are the so-called *EZ templates*. To destroy a Container, type `vzctl destroy ctid`.
`Monospace`	The names of commands, files, and directories.	Use `vzctl start` to start a Container.
`Preformatted`	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	`Saved parameters for Container 101`
`Monospace Bold`	What you type, as contrasted with on-screen computer output.	**`# rpm –V virtuozzo-release`**
Key+Key	Key combinations for which the user must press and hold down one key and then press another.	Ctrl+P, Alt+F4

Besides the formatting conventions, you should also know about the document organization convention applied to Parallels documents: chapters in all guides are divided into sections, which, in their turn, are subdivided into subsections. For example, About This Guide is a section, and Documentation Conventions is a subsection.

Getting Help

In addition to this guide, there are a number of other resources available for Parallels Server Bare Metal which can help you use the product more effectively. These resources include:

Manuals:

- *Parallels Server 4 Bare Metal Installation Guide*. This guide provides detailed information on installing Parallels Server Bare Metal on your server, including the pre-requisites and the stages you shall pass.

- *Getting Started With Parallels Server 4 Bare Metal*. This guide provides basic information on how to install Parallels Server Bare Metal on your server, create new Containers and virtual machines, and perform main operations on them. As distinct from the *Parallels Server 4 Bare Metal Installation Guide*, it does not contain detailed description of all the operations needed to install and set Parallels Server Bare Metal to work (e.g. installing Parallels Server Bare Metal in text mode).

- *Parallels Server 4 Bare Metal Templates Management Guide*. This guide is meant to provide complete information on Parallels templates - an exclusive Parallels technology allowing you to efficiently deploy standard Linux applications inside your Containers and to greatly save the physical server resources (physical memory, disk space, etc.).

- *Parallels Command Line Reference Guide*. This guide is a complete reference on all Parallels Server Bare Metal configuration files and command line utilities.

- *Deploying Clusters in Parallels-Based Systems*. This guide describes the process of creating Parallels failover and GFS clusters using the Red Hat Cluster Suite (RHCS) software.

Help systems:

- *Getting Started with Parallels Management Console*. This help system provides information on how to start working in Parallels Management Console. You will learn how to install this application on your computer, connect to a server running Parallels Server Bare Metal, and perform the basic operations on your virtual machines.

- *Parallels Management Console User's Guide*. This help system provides detailed information on Parallels Management Console - a graphical user interface tool for managing physical servers and their virtual machines.

Feedback

If you spot a typo in this guide, or if you have thought of a way to make this guide better, you can share your comments and suggestions with us by completing the feedback form at the Parallels documentation feedback page (http://www.parallels.com/en/support/usersdoc/).

CHAPTER 2

Parallels Server 4 Bare Metal Basics

This chapter provides a brief description of Parallels Server 4 Bare Metal, Parallels virtual machines and Containers, their specifications and underlying technologies.

In This Chapter

Parallels Server 4 Bare Metal Overview

Parallels Server 4 Bare Metal provides you with the possibility to simultaneously run Parallels virtual machines and Containers on the same server. Using this software, you can efficiently use your server's hardware resources by sharing them among multiple virtual machines and Containers.

Graphically, a Mac with the Parallels Server Bare Metal software installed can be represented as follows:
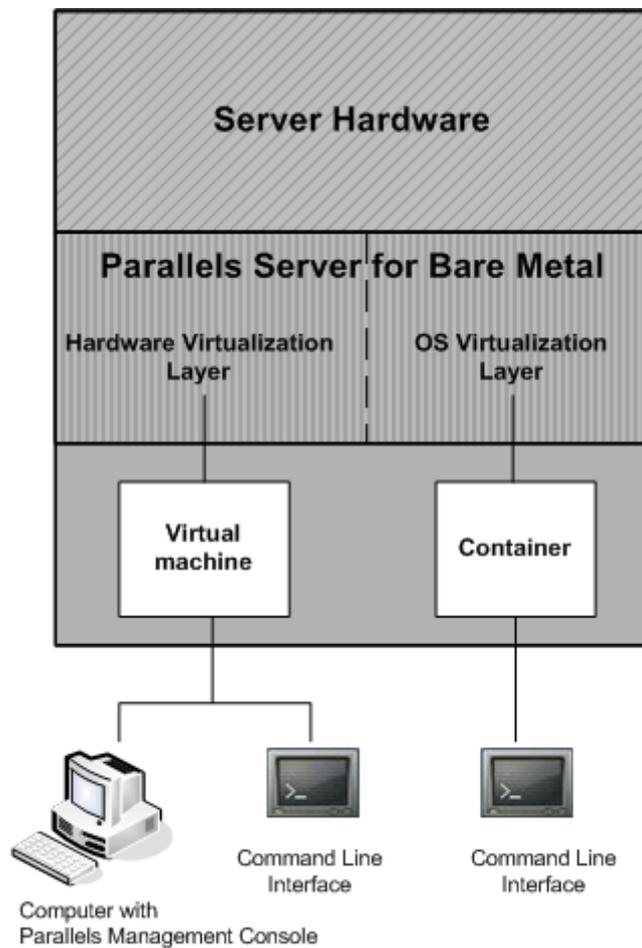
At the base resides server hardware. Next is the Parallels Server Bare Metal software which is installed directly on the server hardware and does not need any operating system for its functioning. Parallels Server Bare Metal includes two virtualization layers:

- **Hardware virtualization layer**. This layer provides the necessary environment for creating and managing Parallels virtual machines.
- **OS virtualization layer**. This layer provides the necessary environment for creating and managing Parallels Containers.

For more information on both layers, see **OS Virtualization Layer** (p. 13) and **Hardware Virtualization Layer** (p. 16).

Effectively uniting both virtualization technologies, Parallels Server Bare Metal provides the best value for cost conscious organizations enabling them to:

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

Parallels Server Bare Metal allows you to create virtual machines and Containers and Containers and manage them using the same tools you would use on systems running Parallels Server 3.0 and Parallels Virtuozzo Containers 4.0. These tools include:

- **Command line interface (CLI)**. This tool comprises a set of Parallels command line utilities and can be used to manage virtual machines and Containers both locally and remotely.
- **Parallels Management Console**. Parallels Management Console is a remote management tool for Parallels Server Bare Metal with a graphical user interface. This tool can be used to manage physical servers and Parallels virtual machines residing on them.

> **Note**: In this version of Parallels Server Bare Metal, you cannot use Parallels Management Console to create and manage Parallels Containers.

OS Virtualization Layer

This section provides detailed information on the OS virtualization layer, one of the two components of Parallels Server Bare Metal, responsible for providing support for Parallels Containers.

Basics of OS Virtualization

The OS virtualization allows you to virtualize physical servers on the operating system (kernel) layer. The diagram below shows the basic architecture of OS virtualization.
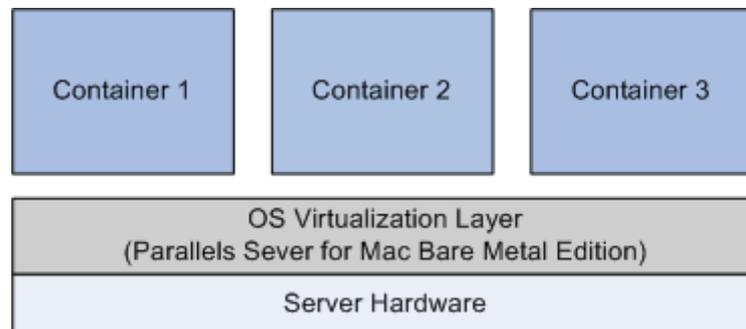
The OS virtualization layer ensures isolation and security of resources between different Containers. The virtualization layer makes each Container appear as a standalone server. Finally, the Container itself houses its own applications and workload. OS virtualization is streamlined for the best performance, management, and efficiency. Its main advantages are the following:

- Containers perform at levels consistent with native servers. Containers have no virtualized hardware and use native hardware and software drivers making its performance unbeatable.

- Each Container can seamlessly scale up to the resources of an entire physical server.

- OS virtualization technology provides the highest density available from a virtualization solution. You can create and run up to 100s of Containers on a standard production physical server.

- Containers use a single OS, making it extremely simple to maintain and update across Containers. Applications may also be deployed as a single instance.

Parallels Containers

From the point of view of applications and Container users, each Container is an independent system. This independence is provided by the Parallels Server Bare Metal OS virtualization layer. Note that only a negligible part of the CPU resources is spent on virtualization (around 1-2%). The main features of the virtualization layer implemented in Parallels Server Bare Metal are the following:

- A Container looks like a normal Linux system. It has standard startup scripts; software from vendors can run inside Containers without any modifications or adjustment.

- A user can change any configuration file and install additional software inside Containers.

- Containers are fully isolated from each other (file system, processes, sysctl variables) and Parallels virtual machines.

- Containers share dynamic libraries, which greatly saves memory.

- Processes belonging to a Container are scheduled for execution on all available CPUs. Consequently, Containers are not bound to only one CPU and can use all available CPU power.

Virtuozzo File System

Virtuozzo File System (VZFS) is a file system that allows sharing common files among multiple Containers without sacrificing flexibility. It is possible for Container users to modify, update, replace, and delete shared files. When a user modifies a shared file, VZFS creates a private copy of the file transparently for the user. Thus, the modifications do not affect the other users of the file. Main benefits of VZFS are the following:

- VZFS saves memory required for executables and libraries. A typical Container running a simple web site might consume around 20–30 MB of RAM just for executable images. Sharing this memory improves scalability and total system performance.

- VZFS saves disk space. A typical Linux server installation occupies several hundred MB of disk space. Sharing the files allows you to save up to 90% of disk space.

- VZFS does not require having different physical partitions for different Containers or creating a special "file system in a file" setup for a Container. This significantly simplifies disk administration.

- Disk quota enables the administrator to limit disk resources available to a Container on the fly. Disk quota for users and groups inside Containers is also supported.

Templates

A template (or a package set) in Parallels Server Bare Metal is a set of original application files repackaged for mounting over Virtuozzo File System. Usually it is just a set of RPM packages for Red Hat like systems. Parallels Server Bare Metal provides tools for creating templates, installing, upgrading, adding them to and removing them from a Container. Using templates lets you:

- share the RAM among similar applications running in different Containers to save hundreds of megabytes of memory

- share the files comprising a template among different Containers to save gigabytes of disk space

- deploy applications simultaneously in many Containers

- use different versions of an application on different Containers (for example, perform an upgrade only in certain Containers)

There are two types of templates: OS templates and application templates. An OS template is an operating system and the standard set of applications to be found right after the installation. Parallels Server Bare Metal uses OS templates to create new Containers with a preinstalled operating system. An application template is a set of repackaged software packages optionally accompanied with configuration scripts. Application templates are used to add extra software to existing Containers. For example, you can create a Container on the basis of the `redhat` OS template and add the MySQL application to it with the help of the `mysql` template.

For detailed information on Parallels templates, see the *Parallels Server Bare Metal Templates Management Guide*.

Parallels Server Bare Metal Configuration

Parallels Server Bare Metal allows you to flexibly configure various settings for the physical server in general as well as for each and every Container. Among these settings are disk and user quota, network parameters, default file locations and configuration sample files, and others.

Parallels Server Bare Metal stores all OS virtualization-related configuration information in two types of files: the global configuration file /etc/vz/vz.conf and Container configuration files /etc/vz/conf/<CT_ID>.conf. The global configuration file defines global and default parameters for Container operation, for example, logging settings, enabling and disabling disk quota for Containers, the default configuration file and OS template on the basis of which a new Container is created, and so on. On the other hand, a Container configuration file defines the parameters for a given particular Container, such as disk quota and allocated resources limits, IP address and hostname, and so on. If a parameter is configured both in the global configuration file and in the Container configuration file, the Container configuration file takes precedence. For a list of parameters constituting the global configuration file and the Container configuration files, refer the *Parallels Command Line Reference Guide*.

The configuration files are read when the Parallels Server Bare Metal software and/or Containers are started. However, Parallels Server Bare Metal standard utilities (for example, pctl) 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 the Parallels Server Bare Metal software and/or Containers are started).

Hardware Virtualization Layer

This section familiarizes you with the second component of Parallels Server Bare Metal - the hardware virtualization layer. This layer provides the necessary environment for creating and managing Parallels virtual machines.

Basics of Hardware Virtualization

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

The following diagram shows the basic architecture of hardware virtualization.
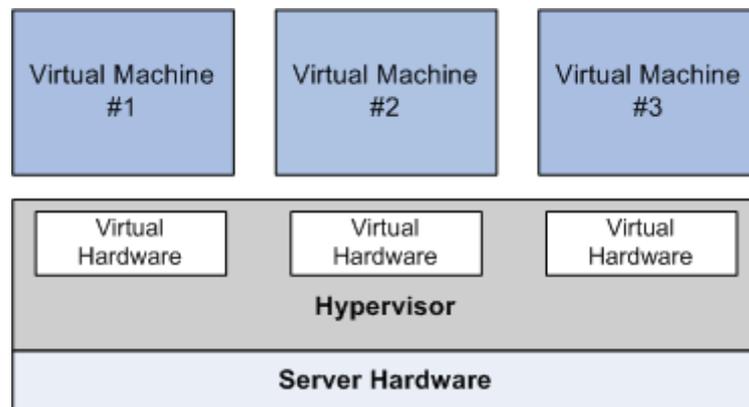
Like OS virtualization, hardware virtualization also provides many benefits the main of which are listed below:

- Create multiple virtual machines with different operating systems on a single physical computer.
- Manage several physical servers at a time using Parallels Management Console, an integrated GUI-based multi-server and cross-platform management tool.
- Run several guest operating systems and their applications simultaneously on a single physical computer without rebooting.
- Consolidate and virtualize the computing environment, reduce hardware costs, lower operating expenses, and increase productivity.
- Use open APIs and SDK to extend management integration with in-house and third-party applications.

Parallels Virtual Machines

From the point of view of applications and virtual machine users, each virtual machine is an independent system with an independent set of virtual hardware. This independence is provided by the Parallels Server Bare Metal hardware virtualization layer. The main features of the virtualization layer are the following:

- A virtual machine looks like a normal computer. It has its own virtual hardware, and software applications can run in virtual machines without any modifications or adjustment.
- A user can easily change the virtual machine configuration (e.g. add a new virtual disk or increase memory).
- Virtual machines are fully isolated from each other (file system, processes, `sysctl` variables) and Parallels Server Bare Metal.
- Install any of the supported operating systems in the virtual machine. The guest operating system and its applications are isolated inside a virtual machine and share physical hardware resources with other virtual machines.

Intel and AMD Virtualization Technology Support

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

Virtual Machine Hardware

A Parallels virtual machine works like a stand-alone computer with the following hardware:

CPU	Up to 12-core Intel/AMD CPU (Intel Celeron or AMD Duron for legacy OS compatibility)
Motherboard	Intel i965 chipset-based motherboard
RAM	Up to 64 GB of main memory
Video Adapter	VGA and SVGA with VESA 3.0 compatible video adapter
Video RAM	Up to 256 MB of video memory
Floppy Disk Drive	1.44 MB floppy disk drive mapped to an image file or to a physical floppy drive
IDE Devices	Up to 4 IDE devices
▪ Hard Disk	Hard disk drive mapped to an image file (up to 2 TB each)
▪ CD/DVD-ROM Drive	CD/DVD-ROM drive mapped to a physical drive or to an image file
SCSI Devices	Up to 15 SCSI devices
▪ Hard Disk	Hard disk drive mapped to an image file (up to 2 TB each)
▪ Generic SCSI Device	Generic SCSI device
Network Interfaces	Up to 16 network interfaces, including Ethernet virtual network cards compatible with RTL8029
Serial (COM) Ports	Up to 4 serial (COM) ports mapped to a socket or to an output file
Parallel (LPT) Ports	Up to 3 parallel (LPT) ports mapped to an output file, to a real port, or to a printer
Sound Card	AC'97-compatible sound card, sound recording support
Keyboard	Generic PC keyboard
Mouse	PS/2 wheel mouse

Virtual Machine Files

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

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

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

Support of Virtual and Real Media

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

Supported Types of Hard Disks

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

Virtual Hard Disks

The capacity of a virtual hard disk can be set from 100 MB to 2 TB.

Virtual hard disks can be of either *plain* or *expanding* format. When you create a virtual machine in Express Windows or Typical mode (in the New Virtual Machine wizard), the disk is created in the *expanding* format.

plain	A plain virtual hard disk image file has a fixed size. The size is determined when the disk is created. Plain disks can be created with the help of New Virtual Machine wizard (the Custom mode.)
expanding	An expanding virtual hard disk image file is small initially. Its size grows as you add applications and data to the virtual hard disk in the guest OS.

Split disks

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

CD/DVD Discs and Their Images

Parallels Server can access real CD/DVD discs and images of CD/DVD discs.

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

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

Parallels Server supports CD/DVD disc images in ISO, CUE, and CCD formats.

Floppy Disks and Floppy Disk Images

Parallels Server can use two types of floppy disks:

- Real diskettes inserted into a real floppy disk drive that is connected to the virtual machine.
- Floppy disk image files having the .fdd extension and connected to the virtual machine.

Parallels Server treats floppy disk images like real diskettes. Parallels Server supports floppy disk image files that have the .fdd extension and are 1.44 MB in size.

With Parallels Server, you can also create an image of a blank floppy using the Floppy Disk pane of the Virtual Machine Configuration dialog.

Note: Parallels Server cannot create images of real diskettes.

Parallels Management Console

Parallels Management Console is a remote tool with a graphical user interface (GUI) for managing your physical servers with Parallels Server Bare Metal and virtual machines residing on them. This tool uses a typical client-server architecture.
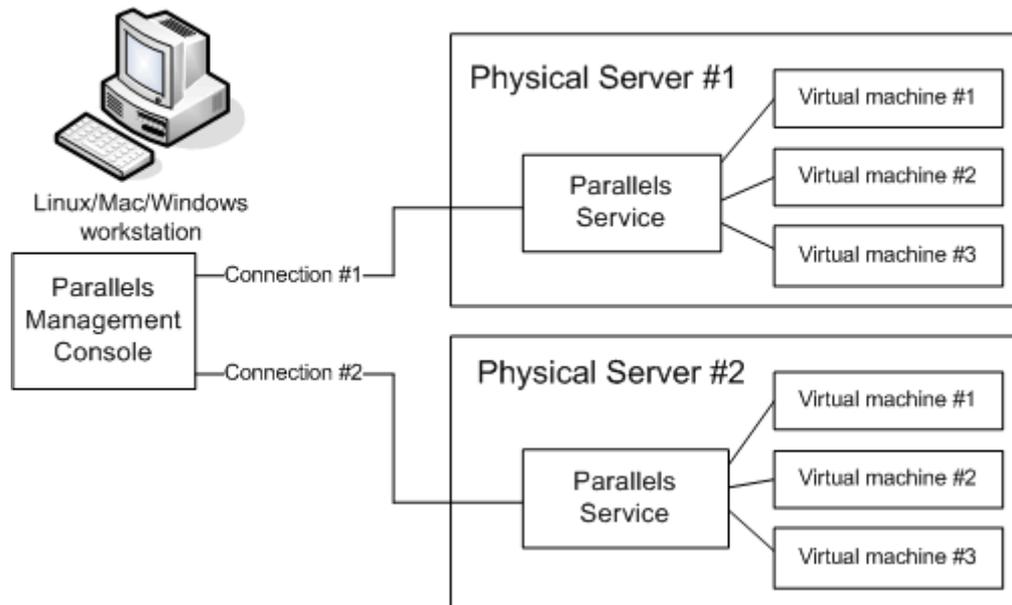
The client application with the graphical user interface is installed on a computer running one of the supported Linux, Mac, or Windows operating systems. Once the client application is up and running, it can connect to the Parallels Server Bare Metal software on a physical server. The client application can control multiple physical servers simultaneously (e.g. *Physical Server #1* and *Physical Server #2* as shown in the picture above). After the connection to the required physical server has been established, you can start managing this server and its virtual machines using the intuitive and comfortable GUI.

Resource Management

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

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

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

Understanding Licensing

To start using the Parallels Server Bare Metal software, you need a special license - *Parallels Server Bare Metal license*. You must install this license on your server after or when installing Parallels Server Bare Metal on it. Every physical server hosting virtual machines and Containers must have its own license. Licenses are issued by Parallels and define a number of parameters in respect of your physical server. The main licensed parameters are listed below:

- The number of CPUs which can be installed on the physical server. Keep in mind that each of the Dual Core and Hyperthreading processors is regarded as one CPU.
- The license expiration date. Any license can be time-limited or permanent.

  Parallels Server Bare Metal licenses have a start date, and if they are time-limited, can also have an expiration date specified in them. You must set up your system clock correctly; otherwise, the license validation may fail.

- The number of virtual machines and Containers the physical server will be able to host.
- The platform and architecture with which the Parallels Server Bare Metal software is compatible.

Physical Server Availability Considerations

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

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

- Use a RAID storage for critical virtual machines and Containers. Do prefer hardware RAIDs, but software mirroring RAIDs might suit too as a last resort.

- Do not run any software on the server itself. Create special virtual machines and Containers where you can host necessary services such as BIND, FTPD, HTTPD, and so on. On the server, you need only the SSH daemon. Preferably, it should accept connections from a pre-defined set of IP addresses only.

- Do not create users on the server itself. You can create as many users as you need in any virtual machine and Container. Remember: compromising the server means compromising all virtual machines and Containers as well.

CHAPTER 3

Operations on Virtual Machines and Containers

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

Note: We assume that you have successfully installed, configured, and deployed your Parallels Server Bare Metal system. If you have not, refer to the *Parallels Server Bare Metal Installation Guide* providing detailed information on these operations.

In This Chapter

Creating a Virtual Machine and Container

This section explains how to create a new Parallels virtual machine and Container. The options you should pass to this command differ depending on whether you want to create a virtual machine or Container.

Creating a Container

To create a Container, you can use the `pctl create` command. This command requires the following parameters:

Argument	Description
Container ID	A numeric ID associated with a Container (101, 403, and so on). The Container ID should be an integer greater than 100 and unique for a given Parallels server.
OS template name	The name of the OS template to base your Container on. Parallels Server Bare Metal is shipped with a number of ready-to-use OS templates. To find out the names of the available templates, use the `vzpkg list -O` command.
	For the list of operating systems you can run in your virtual machines and Containers, see **Supported Guest Operating Systems** (p. 28).
Configuration file	The name of the sample configuration file that will be used for setting all the Container resource control parameters. The sample configuration files are residing in the `/etc/vz/conf` directory on the physical server and have names with the following mask: `ve-<configname>.conf-sample`. The most commonly used sample is the `ve-basic.conf-sample` file. This sample file has resource control parameters suitable for most Containers.

Thus, for example, you can create a new Container by executing the following command:

```
# pctl create 101 --ostemplate fedora-core-9-x86 --config basic
Creating Container private area (fedora-core-9-x86)
...
Container private area was created
```

In this case Parallels Server Bare Metal will create a Container with ID 101, the Fedora 9 OS installed inside, and the configuration parameters taken from the `ve-basic.conf-sample` sample configuration file.

Note: For more information on options you can pass to `pctl create` when creating Containers, see the *Parallels Command Line Reference Guide*.

Creating a Virtual Machine

The process of creating a new virtual machine includes the following steps:

1 Creating a virtual machine configuration. To create a virtual machine configuration, you can use either the `pctl create` command or Parallels Management Console.

2 Installing an operating system in the virtual machine. This operation can be performed using Parallels Management Console only.

The example below shows you how to create a new virtual machine configuration using `pctl create`:

```
# pctl create MyVM --distribution win-2008 --location /vz/VMs
Creating the virtual machine...
Generate the VM configuration for win-2008.
The VM has been successfully created.
```

This will create a virtual machine with the name of `MyVM`, adjust its configuration for installing the Windows Server 2008 operating system in it, and place all virtual-machine-related files in the `/vz/VMs` directory. Now you can use Parallels Management Console to install Windows Server 2008 OS in this virtual machine. For information on how you can do it, see the *Parallels Management Console User's Guide* or *Getting Started With Parallels Management Console* guide.

Note: For more information on options you can pass to `pctl create` when creating virtual machines, see the *Parallels Command Line Reference Guide*.

Supported Guest Operating Systems

Listed below are the operating systems that you can run in your virtual machines and Containers:

Operating System	Virtual Machine	Container
Windows		
Windows 7 (x32, x64)	+	-
Windows Server 2008 R2 (x32, x64)	+	-
Windows Server 2003 R2 (x32, x64)	+	-
Windows Vista with Service Pack 1 and 2 (x32, x64)	+	-
Windows XP with Service Pack 2 and 3 (x32, x64)	+	-
Windows 2000 with Service Pack 4 (x32, x64)	+	-
Linux		
Red Hat Enterprise Linux 5.3 (x32, x64)	+	+
Red Hat Enterprise Linux 4.7 (x32, x64)	+	+
Fedora 11 (x32, x64)	+	+
Fedora 10 (x32, x64)	+	+
CentOS 5.3 (x32, x64)	+	+
CentOS 4.7 (x32, x64)	+	+
SUSE Linux Enterprise Server 10 (x32, x64)	+	+
Debian GNU/Linux 5.0 (x32, x64)	+	+
Debian GNU/Linux 4.0 (x32, x64)	+	+
Ubuntu Linux 9.04 (x32, x64)	+	+
Ubuntu Linux 8.10 (x32, x64)	+	+
BSD		
FreeBSD 7 (x32, x64)	+	-
FreeBSD 6 (x32, x64)	+	-

Choosing a Container ID

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

- ID 0 is used for the Parallels server itself. You cannot and should not try to create a Container with ID 0.
- Parallels Server Bare Metal reserves the IDs ranging from 0 to 100. *Please do not create Containers with IDs below 101.*

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

Another approach to assigning Container IDs is to follow some pattern of Container 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 Container with the 10.0.17.15 IP address, the 39108 ID to the Container with the 10.0.39.108 IP address, and so on. This makes it much easier to run a number of Parallels utilities eliminating the necessity to check up the Container IP address by its ID and similar tasks. You can also think of your own patterns for assigning Container IDs depending on the configuration of your network and your specific needs.

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

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

This output shows that Container 101 does not exist on the particular server; otherwise it would be present in the list.

WARNING! When deciding on a Container ID, do not use the ID of any Container that was ever present in the system unless you are sure that no data belonging to the old Container remains on the server. The fact is that the administrator of the newly-created Container might have access to these data in this case, i.e. to the backups of the old Container, its logs, statistics, etc.

Choosing OS EZ Template

Before starting to create a Container, you shall decide on which OS EZ template your Container will be based. There might be several OS EZ templates installed on the server and prepared for the Container creation; use the `vzpkg list` command to find out what OS EZ templates are available on your system:

```
# vzpkg list -O
redhat-el5-x86                    2009-05-21 23:59:44
fedora-core-8-x86         2009-12-11 12:45:52
```

The `-O` option passed to the `vzpkg list` command allows you to list only OS EZ templates installed on the server. As you can see, the `redhat-el5-x86` and `fedora-core-8-x86` OS EZ templates are currently available on the server. The time displayed beyond OS EZ templates indicates when the corresponding EZ template was cached.

You can also use the `--with-summary` option to display brief information on the installed OS EZ templates:

```
# vzpkg list -O --with-summary
redhat-el5-x86    :Red Hat Enterprise Linux v.5 Server EZ OS template
fedora-core-8-x86 :Fedora Core 8 EZ OS template
```

For complete information on the `vzpkg list` command, you can consult the *Parallels Command Line Reference Guide*.

Performing Initial Configuration

Before starting your newly created virtual machine and Container, you first need to configure it. This section describes the configuration steps for virtual machines and Containers.

Configuring Network Settings

To be accessible from the network, a virtual machine and Container must be assigned a correct IP address. DNS servers must also be configured. The session below illustrates setting the main network parameters for a virtual machine having the name of `MyVM`:

▪ To set the IP address of `10.0.186.1`, run this command:

```
# pctl set MyVM --device-set net0 --ipadd 10.0.186.1
```

(`net0` is the network card to assign the IP address to.)

▪ To set the DNS server IP address of `192.168.1.165`, run this command:

```
# pctl set MyVM --nameserver 192.168.1.165
```

To set the aforementioned parameters for a Container, just replace the virtual machine name (`MyVM`) with the corresponding Container ID (e.g. 101). You also need to specify the `--save` flag in the commands above if you wish to save all the parameters to the Container configuration file, while running these commands for a virtual machine will automatically save the changes in its configuration file.

Note: You can configure the network settings only inside virtual machines that have Parallels Tools installed.

Setting the Password for a Virtual Machine and Container

In Parallels Server Bare Metal, you can use the `--userpasswd` option of the `pctl set` command to create new accounts in your virtual machines and Containers directly from the Parallels server. The created account can then be used to log in to the virtual machine and Container. The easiest way of doing it is to run this command:

```
# pctl set MyVM --userpasswd user1:2wsx123qwe
```

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

The `pctl set` command can also be used to change passwords for existing accounts in your virtual machines and Containers. For example, to change the password for `user1` in the `MyVM` virtual machine to `0pi65jh9`, run this command:

```
# pctl set MyVM --userpasswd user1:0pi65jh9
```

Note: You can use manage user accounts only inside virtual machines that have Parallels Tools installed.

Setting Startup Parameters

The `pctl set` command allows you to define the `onboot` startup parameter for virtual machines and Containers. Setting this parameter to `yes` makes your virtual machine and Container automatically boot at the physical server startup. For example, to enable Container 101 and the `MyVM` virtual machine to automatically start on your server boot, you can execute the following commands:

- For Container 101:

```
# pctl set 101 --onboot yes --save
Saved parameters for Container 101
```

- For the `MyVM` virtual machine:

```
# pctl set MyVM --onboot yes
```

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

Installing Parallels Tools

If you are creating a Parallels virtual machine, you are also recommended to install Parallels Tools. Parallels Tools are a set of special utilities that help you use your virtual machines in the most comfortable and efficient way. With Parallels Tools, you can move the mouse seamlessly outside the guest OS window without pressing any key, change the virtual machine's screen resolution by simply resizing its window, synchronize your virtual machine's time and date settings with the time setting of the host computer, and share clipboard of your computer with the virtual machine's clipboard.

The process of installing Parallels Tools differs depending on the operating system running in a virtual machine and is described in the following sections.

Installing Parallels Tools in a Windows Virtual Machine

Parallels Tools are available for the following Windows guest operating systems:

- Windows 2000
- Windows Server 2003
- Windows XP
- Windows Vista
- Windows Server 2008

To install Parallels Tools in any of these operating systems:

1 Make sure the respective virtual machine is running.

2 Execute the `pctl installtools` *VM_Name* command on the server. For example, you can run this command to install Parallels Tolls in the `MyVM` virtual machine.

```
# pctl installtools MyVM
```

For more information on Parallels Tools, refer to the *Parallels Management Console User's Guide*.

Installing Parallels Tools in a Linux Virtual Machine

Parallels Tools can be installed in any supported Linux guest operating systems that have the following packages:

- `X.org 6.7` and later
- `glibc2.4` and later

To install Parallels Tools, do the following:

1   Make sure the respective virtual machine is running.

2   Mount the CD with Parallels Tools to the virtual machine:

```
pctl installtools VM_Name
```

3   Start a terminal in your Linux guest OS, and gain the `root` privileges using the `su` command.

4   Go to the CD/DVD-ROM directory:

```
cd /media/cdrom/
```

> **Note:** In some Linux operating systems, the mount point for the virtual CD/DVD-ROM drive may be different (e.g. `/media/Parallels\ Tools/`).

5   Launch the Parallels Tools installation:

```
./install
```

6   Follow the instructions to complete the installation.

7   Once the installation of Parallels Tools is complete, restart the virtual machine.

Mounting Parallels Tools Manually

In some Linux operating systems, the image with Parallels Tools may be mounted as non-executable after running the `pctl installtools` command (see **Step 2** above). In this case, you need to unmount the Parallels Tools image and then mount it again with the `exec` flag. The following session demonstrates how you can do this:

1   Check that the Parallels Tools CD image is mounted as non-executable.

```
mount | grep iso9660
```

The command should return something like this:

```
/dev/cdrom on /media/cdrom type iso9660 (ro,noexec,nosuid,nodev,uid=0)
```

The `noexec` option in the command output confirms that the CD image is currently mounted as non-executable.

2   Unmount the CD image:

```
umount /dev/cdrom
```

3   Mount the CD image with the `exec` option:

```
mount -o exec /dev/cdrom /media/cdrom
```

4   Proceed to **Step 4** above.

For more information on Parallels Tools, see the *Parallels Management Console User's Guide*.

Starting, Stopping, and Querying Status of a Virtual Machine and Container

After a Parallels virtual machine and Container has been created, it can be managed like an ordinary computer.

Starting a Virtual Machine and Container

You can use the `pctl start` command to start your virtual machines and Containers:

- To start Container 101:

```
# pctl start 101
Starting the Container ...
```

- To start a virtual machine with the name of `MyVM`:

```
# pctl start MyVM
Starting the VM ...
```

Stopping a Virtual Machine and Container

The `pctl stop` command is used to stop your virtual machines and Containers:

- To stop Container 101:

```
# pctl stop 101
Stopping the Container ...
```

- To stop a virtual machine with the name of `MyVM`:

```
# pctl stop MyVM
Stopping the VM ...
```

Checking the Status of a Virtual Machine and Container

Depending on whether you want to check the status of a Container or a virtual machine, you can use the following commands:

- `pctl status` to check the Container status:

```
# pctl status 101
VEID 101 exist mounted running
```

- `pctl list` to check the virtual machine status:

```
# pctl list MyVM
stopped 10.12.12.121 MyVM
```

You can also get more detailed information on a virtual machine by specifying the `-i` option after `pctl list`.

Restarting a Virtual Machine and Container

Sometimes, you may need to restart a virtual machine and Container. To do this, use the following commands:

- To restart a Container, use the `pctl restart` command:

```
# pctl restart 101
Stopping Container ...
```

```
Container was stopped
Container is unmounted
Starting Container ...
Container is mounted
Adding IP address(es): 10.0.186.101
Container start in progress...
```

- To restart a virtual machine, use the `pctl reset` command:

```
# pctl reset MyVM
```

Listing Virtual Machines and Containers

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

```
# pctl list -a
STATUS    IP_ADDR       NAME
started   10.10.1.101   101
stopped   10.10.100.1   MyVM
```

The `-a` option tells the `pctl list` command to output both running and stopped virtual machines and Containers. By default, only running virtual machines and Containers are shown. The default columns inform you of the Container IDs and virtual machine names, the virtual machine and Container status and IP addresses. This output can be customized as desired by using `pctl list` command line options. For example:

```
# pctl list -a -o name,ctid -a
NAME                                        ID
-                                           101
MyVm    {b8cb6d99-1af1-453d-a302-2fddd8f86769}
```

This command displays only the names and IDs of the virtual machines and Containers existing on the physical server. The full list of the `pctl list` command options for virtual machines and Containers is available in the *Parallels Command Line Reference Guide*.

Storing Extended Information on a Virtual Machine and Container

Sometimes, it may be difficult to remember the information on certain virtual machines and Containers. The probability of this increases together with the number of virtual machines and Containers and with the time elapsed since their creation. Parallels Server Bare Metal allows you to set the description of any virtual machine and Container on the physical server and view it later on, if required. The description can be any text containing any virtual machine and Container-related information. For example, you can include the following in the virtual machine and Container description:

- the owner of the virtual machine and Container
- the purpose of the virtual machine and Container
- the summary description of the virtual machine and Container

Let us assume that you are asked to create a virtual machine for a Mr. Johnson who is going to use it for hosting the MySQL server. So, you create the `MyVM` virtual machine and, after that, execute the following command on the physical server:

```
# pctl set MyVM --description "MyVM
> owner - Mr. Johnson
> purpose - hosting the MySQL server" -
The VM has been successfully configured.
```

This command saves the following information related to the virtual machine: its name, owner, and the purpose of its creation. At any time, you can display this information by issuing the following command:

```
# pctl list -o description MyVM
MyVM
owner - Mr. Johnson
purpose - hosting the MySQL server
```

When working with virtual machine and Container descriptions, keep in mind the following:

- You can use any symbols you like in the virtual machine and Container description (new lines, dashes, underscores, spaces, etc.).
- If the virtual machine and Container description contains one or more spaces or line breaks (as in the example above), it must be put in single or double quotes.
- As distinct from a virtual machine and Container name and ID, a description cannot be used for performing virtual machine and Container-related operations (e.g. for starting or stopping a virtual machine and Container) and is meant for reference purposes only.

Copying a Virtual Machine and Container Within the Server

Parallels Server Bare Metal allows you to create a complete copy of a particular virtual machine and Container (in respect of all the virtual machine and Container data and resources parameters), or a *clone*. This saves your time because you do not have to think of setting up the virtual machine and Container configuration parameters and the like. Moreover, you can create a number of virtual machine and Container clones at a sitting.

In Parallels Server Bare Metal-based systems, you can use the following commands to copy a virtual machine and Container within the given physical server:

- `vzmlocal` to clone a Container. For example, you can create Container 111 and make it be a complete copy of Container 101 by running this command:

```
# vzmlocal -C 101:111
Moving/copying Container#101 -> Container#111, [], [] ...
...
Successfully completed
```

  You can clone both running and stopped Containers.

- `pctl clone` to clone a virtual machine. For example, you can create a clone of the `MyVM` virtual machine and assign the `Cloned_VM` name to it as follows:

```
# pctl clone MyVM --name ClonedVM
Clone the MyVM VM to the VM ClonedVM...
The VM has been successfully cloned.
```

  You can create clones of stopped virtual machines only.

Checking the Cloned Virtual Machine and Container

To check that your virtual machine and Container has been successfully moved, run this command:

```
# pctl list -a
STATUS        IP_ADDR       NAME
stopped       10.0.10.101   101
stopped       10.0.10.101   111
stopped       10.0.10.115   MyVM
stopped       10.0.10.115   ClonedVM
```

As you can see from the example above, the clones of Container 101 (Container 111) and the `MyVM` virtual machine (`ClonedVM`) have been successfully created. However, before starting to use the clones, you should assign different IP addresses to them which are currently identical to those of Container 101 and `MyVM`. Refer to **Performing Initial Configuration** (p. 30) to learn how you can do it.

Note: If you are cloning a running Container, the created clone is stopped to prevent an IP address conflict.

Configuring the Default Directories

When cloning a virtual machine and Container, you can also override the following default directories:

- /vz/*dest_VM_Name*.pvm storing the files of a cloned virtual machine (where *dest_VM_Name* denotes the name of the resulting virtual machine). For example, for the ClonedVM virtual machine, this directory is /vz/ClonedVM.pvm. To store the files of the ClonedVM virtual machine in a custom directory, you can run the following command:

```
# pctl clone MyVM --name ClonedVM --location /vz/VM_directory
```

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

- /vz/private/*<dest_CTID>* and */vz/root/<dest_CTID>* defining the Container private area and root paths, respectively (where *<dest_CTID>* denotes the ID of the resulting Container). In the case of Container 111, these paths are /vz/private/111 and /vz/root/111. To define custom private area and root paths for Container 111, you can execute the following command:

```
# vzmlocal -C 101:111:/vz/private/dir_111/:/vz/root/ct111
Moving/copying Container#101 -> Container#111, [], [] ...
Syncing private area '/vz/private/101'->'/vz/private/dir_111'
...
Successfully completed
# ls /vz/private
1  101 dir_111
# ls /vz/root
1  101 ct111
```

Suspending a Virtual Machine and Container

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

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

You can use the pctl suspend command to save the current state of a virtual machine and Container. For example, you can issue the following command to suspend the MyVM virtual machine:

```
# pctl suspend MyVM
Suspending the VM...
The VM has been successfully suspended.
```

At any time, you can resume the MyVM virtual machine by executing the following command:

```
# pctl resume MyVM
Resuming the VM...
The VM has been successfully resumed
```

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

Running Commands in a Virtual Machine and Container

Parallels Server Bare Metal allows you to execute arbitrary commands inside virtual machines and Containers by running them on the physical server, i.e. without the need to log in to the respective virtual machine and Container. For example, this can be useful in these cases:

- If you do not know the virtual machine and Container login information, but need to run some diagnosis commands to verify that it is operational.
- If network access is absent for a virtual machine and Container.

In both these cases, you can use the `pctl exec` command to run a command inside the respective virtual machine and Container. The session below illustrates the situation when you run the stopped SSH daemon inside a Linux virtual machine with the name of `My_Linux`:

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

Notes:

1. You can use the `pctl exec` command only inside virtual machines that have Parallels Tools installed.

2. The `pctl exec` command is executed inside a virtual machine and Container from the `/` directory rather than from the `/root` one.

Deleting a Virtual Machine and Container

You can delete a virtual machine and Container that is not needed anymore using the `pctl delete` command. Notice that you cannot delete a running or mounted virtual machine and Container. The example below illustrates deleting Container 101 and the `MyVM` virtual machine:

Deleting Container 101

```
# pctl delete 101
Deleting Container private area: /vz/private/101
Container is currently mounted (unmount first)
# pctl stop 101
Stopping Container...
Container was stopped
Container is unmounted
# pctl delete 101
Deleting Container private area: /vz/private/101
Container private area was deleted
```

Deleting the MyVM virtual machine:

```
# pctl delete MyVM
Deleting the VM...
VM is currently running
# pctl stop MyVM
Stopping the VM...
VM was stopped
# pctl delete MyVM
Deleting the VM...
Container was deleted
```

Managing Virtual Machine and Container Backups

A regular backing up of the existing virtual machines and Containers is essential for any physical server reliability. In Parallels Server Bare Metal, you can use the following utilities to back up and restore your virtual machines and Containers:

- `pctl`
- `pbackup`
- `prestore`

Detailed information on these utilities is provided in the following subsections.

Backups Overview

Parallels Server Bare Metal backup utilities deal with three kinds of servers:

- *Source Server*. This is the server where virtual machines and Containers are hosted during their backing up.
- *Backup Server*. This is the server where virtual machine and Container backups are stored. A Backup Server can be any server running the Parallels Server Bare Metal software and having sufficient space for storing virtual machine and Container backups.
- *Destination Server*. This is the server where virtual machine and Container backups are restored.
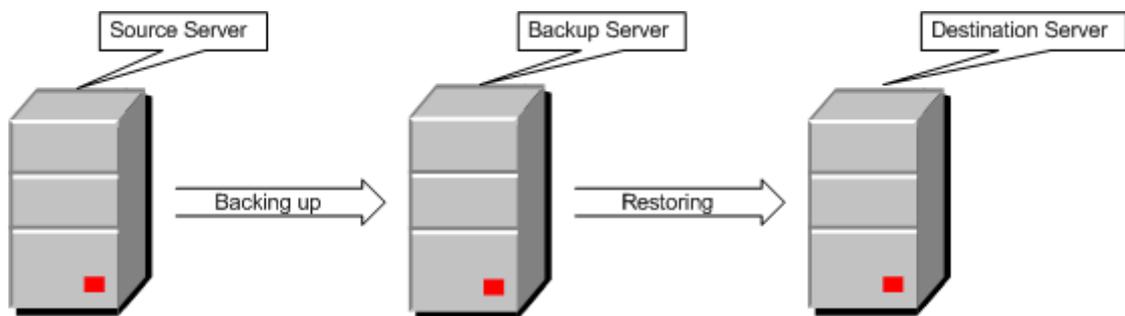
These servers are singled out by their functionality only. In reality, one and the same physical server can perform two or even three functions. Usually, the Source and Destination Servers are represented by one and the same server because you will likely want the virtual machines and Containers you back up to be restored to their original server. However, setting up a dedicated Backup Server is recommended.

Creating Consistent Backups of Virtual Machines

Parallels Server Bare Metal allows you to back up both running and stopped virtual machines. However, to create a consistent backup of a running virtual machine, it must meet the following requirements:

- Have Parallels Tools installed.
- Run one of the following operating systems:

  Windows operating systems
  - Windows Server 2003
  - Windows Server 2008
  - Windows Vista
  - Windows 7

  Linux operating systems
  - Suse, version 9.0 and higher
  - RHEL, version 4.0 and higher
  - CentOS, version 4.0 and higher
  - Fedora Core, version 3 and higher
  - Debian, version 3.1 and higher

- Ubuntu, version 4.10 and higher

Using pctl backup and pctl restore

This section describes how to perform the basic backup-related operations using the `pctl` utility.

Creating a Virtual Machine and Container Backup

You can use the `pctl backup` command to back up your virtual machines and Containers. This command is executed on the Source Server and can store the created virtual machine and Container backup on both the Source and Backup Servers. When creating a backup on the Source Server, you only need to specify the name of the virtual machine and Container to back up. For example, you can execute the following command to back up the `MyVM` virtual machine and store its backup archive on the Source Server:

```
# pctl backup MyVM
Backing up the VM MyVM
Operation progress 100%
The virtual machine has been successfully backed up with backup ID {746dba2a-
3b10-4ced-9dd6-76a2blcl4a69}
```

The command output informs you that the virtual machine backup has been successfully created and assigned ID `746dba2a-3b10-4ced-9dd6-76a2blcl4a69`. You can use this ID when managing the backup archive (e.g. remove the backup).

At the same time, you can run the following command to back up the `MyVM` virtual machine and store its backup archive on the Backup Server with the IP address of `129.129.10.10`:

```
# pctl backup MyVM -s root:1qaz2wsx@129.129.10.10
```

`root:1qaz2wsx` before the Destination Server IP address denotes the root credentials used to log in to this server. If you do not specify these credentials, you will be asked to do so during the command execution.

All newly created backups are placed to the directory specified as the value of the `BACKUP_DIR` parameter in the `/etc/vzbackup.conf` configuration file. By default, this directory is `/vz/backups/ID` (where `ID` is the ID of the respective virtual machine and Container).

Notes:

1. A Backup Server can be any server running the Parallels Server Bare Metal software and having sufficient space for storing virtual machine and Container backups.

2. For more information on the options you can pass to `pctl backup`, refer to the *Parallels Command Line Reference Guide*.

Listing the Existing Backups

You can use the `pctl backup-list` command to view the backups existing on the physical server. For example:

```
# pctl backup-list
                                ID    Backup_ID
Node              Date    Type        Size
```

```
{c1dee22f-8667-4870-9e11-278f1398eab0}  {209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca}
test.com  2009-06-30 10:19:32      f   411566405
```

This command lists the backups existing on the Source Server. If you want to list the backups on the Backup Server, you need to specify the IP address of this server.

The command output shows that currently only two backups exist on the Source Server. These backups were created for a virtual machine and Container with the ID of `c1dee22f-8667-4870-9e11-278f1398eab0`. The information on the backups is presented in the following table:

Column Name	Description
ID	The ID uniquely identifying the virtual machine and Container.
Backup ID	The ID assigned to the backup archive. You need to specify this ID when performing any backup-related operations.
Node	The hostname of the physical server storing the backup archive.
Date	The date and time when the backup archive was created.
Type	The backup type. Currently, you can create two types of backups: ▪ A full backup indicated by `f`. ▪ An incremental backup indicated by `i` and containing only the files changed since the previous full or incremental backup. This is the default backup type.
Size	The size of the backup archive, in bytes.

Removing a Virtual Machine and Container Backup

At any time, you can remove a backup that you do not need any more using the `pctl backup-delete` command. To do this, you need to specify the ID of the backup to remove and the ID of the respective virtual machine and Container. If you do not know these IDs, use the `pctl backup-list` and check the `ID` and `Backup ID` columns. For example:

```
# pctl backup-list
                                ID    Backup_ID
Node               Date    Type        Size
{c1dee22f-8667-4870-9e11-278f1398eab0}  {209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca}
test.com  2009-06-30 10:19:32      f   411566405
# pctl backup-delete c1dee22f-8667-4870-9e11-278f1398eab0 -t 209d54a0-e3b8-
4a03-9ca8-d4cc7a2a27ca
Delete the VM backup
The VM backup has been successfully removed.
```

You can also specify the virtual machine and Container name instead of its ID:

```
# pctl backup-delete MyVM -t 209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca
```

If you have several backups of a particular virtual machine and Container and want to delete them all at once, indicate only the virtual machine and Container name or ID:

```
# pctl backup-delete MyVM
```

This command removes all backups of the `MyVM` virtual machine from the local Backup Server. To remove backups stored remotely, you also need to specify the IP address of the remote Server:

```
# pctl backup-delete MyVM -s root:1qaz2wsx@129.129.10.10
```

Restore a Virtual Machine and Container

To restore a backup of a virtual machine and Container, you can use the `pctl restore` command. This command supports restoring backups to the Source Server only. For example, to restore a backup of the `MyVM` virtual machine stored on the Backup Server with the IP address of `10.10.100.1`, you can run this command on the Source Node:

```
# pctl restore MyVM -s root:1qaz2wsx@10.10.100.1
```

If you have two or more backups of the `MyVM` virtual machine, the latest backup is restored. If you want to restore a particular virtual machine and Container backup, you need to specify the ID of this backup. You can use the `pctl backup-list` command to list the existing backups and the IDs assigned to them:

```
# pctl backup-list -s root:1qaz2wsx@10.10.100.1
                                ID                         Backup_ID
Node               Date    Type      Size
{c1dee22f-8667-4870-9e11-278f1398eab0}  {209d54a0-e3b8-4a03-9ca8-d4cc7a2a27ca}
test.com  2009-06-30 10:19:32     i    11566405
{c1dee22f-8667-4870-9e11-278f1398eab0}  {24a3011c-092e-4f21-bb3b-29ccfe967e92}
test.com  2009-05-21 11:12:35     f    356798701
```

You can now indicate the desired ID after the `-t` option to tell `pctl backup` to restore this particular backup. For example, to restore the backup for the virtual machine with the ID of `c1dee22f-8667-4870-9e11-278f1398eab0` that was created on the 21st of May, you can execute this command:

```
# pctl restore -t {c1dee22f-8667-4870-9e11-278f1398eab0} -s
root:1qaz2wsx@10.10.100.1
```

Using pbackup and prestore

Along with `pctl`, you can use the following utilities to create and manage backups of your virtual machines and Containers:

- `pbackup`. This utility is used to create backups of individual virtual machines and Containers or entire Parallels servers.
- `prestore`. This utility is used to manage the existing backups of virtual machines and Containers.

Backing Up Virtual Machines and Containers

The `pbackup` utility is run on the Backup Server connecting via SSH to the Parallels server and backing up one or more virtual machines and Containers on this server. The created backup archive is then placed to the directory on the Backup Server defined in the `/etc/vzbackup.conf` global backup configuration file. By default, this directory is `/vz/backups`. Later on, the virtual machine and Container backups can be restored from this directory.

Assuming that you are going to back up the entire Parallels server (i.e. all virtual machines and Containers on this server) with the `test.com` hostname, you can run the following command on the Backup Server:

```
# pbackup test.com
```

During the command execution, you will be asked to provide the `test.com` credentials. After doing so, the command will back up all virtual machines and Containers on the `test.com` and put

- all backed up Containers to the Backup Server
- all backed up virtual machines to the Source Server

To save the backed up virtual machines also on the Backup Server, you should additionally specify the `-n` option. This option is used to indicate the IP address or hostname of the Backup Server and its credentials:

```
# pbackup -n root:7ujn6yhb@192.168.10.199 test.com
```

If you wish to back up not all, but specific virtual machines and Containers from the specified server, use the `-e` or `-x` switches (to include or exclude the specified virtual machines and Containers, respectively). For example:

```
# pbackup -n root:7ujn6yhb@192.168.10.199 test.com -e 101 MyVM
```

In this session, only Container 101 and the `MyVM` virtual machine residing on the Source Server with the `test.com` hostname will be included in the backup, and their backups will be stored on the Backup Server.

Notes:

1. A Backup Server can be any server running the Parallels Server Bare Metal software and having sufficient space for storing virtual machine and Container backups.

2. For the full list of configuration parameters and command line options for `pbackup`, refer to the *Parallels Command Line Reference Guide*.

Restoring Backups

To restore any individual virtual machines and Containers or entire Parallels servers, you may want to view first the information about them. This can be done using the `prestore -l` command:

```
# prestore -l -n test.com test.com
root@test.com's password:
...
Backups for node test.com:
                                   ID Backup_ID
Node     Date                 Type  Size
                                   101 2009-07-15T220425+0400@test.com
test.com 2009-06-30 09:42:19  f     18721280
{cd91b90b-469d-42c6-acf4-fefee09cfa61} {4ef87485-ec3b-4594-896b-c7ccbee859b5}
test.com 2009-07-16 17:15:47  f     92617398
```

The command output shows that currently only two backups exist for the `test.com` server on the Backup Server. If you omit the `-n test.com` option, the command will list:

- all Container backups for the `test.com` server stored on the Backup Server
- all virtual machine backups for the `test.com` server stored on the `test.com` server

The information on the backups is presented in the following table:

Column Name	Description
ID	The ID uniquely identifying the virtual machine and Container.
Backup ID	The ID assigned to the backup archive. You need to specify this ID when performing any backup-related operations.
Node	The hostname of the Source Server.
Date	The date and time when the backup archive was created.
Type	The backup type. Currently, you can create two types of backups: - A full backup indicated by `f`. - An incremental backup indicated by `i` and containing only the files changed since the previous full or incremental backup. This is the default backup type.
Size	The size of the backup archive, in bytes.

To restore Container 101 and the {cd91b90b-469d-42c6-acf4-fefee09cfa61} virtual machine, run this command:

```
# prestore -n test.com -e 101 {cd91b90b-469d-42c6-acf4-fefee09cfa61}
```

This command will restore the Container and the virtual machine to their Source Server.

You can also use the `-d` option to restore Container 101 to a Parallels server other than the Source Node. For example, this command

```
# prestore -d 192.168.10.199 test.com -e 101
```

restores Container 101 to the Destination Server with IP address `192.168.10.199`. If you want to restore all Containers backups for the `test.com` Parallels server, just skip the `-e` option.

Notes:

1. The current version of Parallels Server Bare Metal supports restoring virtual machines to the Source Server only.

2. The `prestore` utility can also manage (list, restore, etc.) backups created using the `pctl backup` command. However, you are recommended to use the same utility (either `pctl` or `prestore`) during the life cycle of a particular backup.

3. For the full list of command line options for `prestore`, refer to the *Parallels Command Line Reference Guide*.

Configuring Per-Server Backup Parameters

A number of default parameters in the global backup configuration file can be adjusted for a particular physical server to be backed up. To do this:

1   Create a new configuration file named `server.conf`.

2   Put the file to the backup directory. This directory is defined by the `BACKUP_DIR` parameter in the `/etc/vzbackup.conf` global backup configuration file and is set to `/vz/backups` by default.

The `server.conf` file should contain those parameters that you want to rewrite for a given Parallels server. For a complete list of those backup parameters that can be configured using per-server and configuration files, refer to the *Parallels Command Line Reference Guide*.

Configuring Passwordless Access to the Source Node

You need to provide the Source Server credentials each time you execute the pbackup and prestore commands. However, you can allow these utilities to log in to the Source Server without having to enter the root password. To do this, you should provide each Source Server with authorized public SSH RSA keys:

1   Log in to the Backup Server as root, and generate a pair of SSH keys - public and private:

```
# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
c6:19:a8:2c:67:31:15:e6:30:23:2b:8a:b0:63:77:8f root@dhcp-130.parallels.com
```

Note that you must leave an empty passphrase in the above procedure. The private key is saved by default in /root/.ssh/id_rsa, and the public key is saved in /root/.ssh/id_rsa.pub.

2   Transfer your public key to the /root/.ssh directory on each Source Server (use some intermediary name for the file not to overwrite the corresponding file on the Source Server):

```
# scp /root/.ssh/id_rsa.pub root@dhcp-129.parallels.com:/root/.ssh/temp_name
The authenticity of host 'dhcp-129.parallels.com (192.168.1.129)' can't be
established.
RSA key fingerprint is 01:fc:b6:e9:26:40:1f:1a:41:5f:7a:fb:cf:14:51.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dhcp-129.parallels.com,192.168.1.129' (RSA) to the
list of known hosts.
root@dhcp-129.parallels.com's password:
id_rsa.pub       100% |****************************|   235       00:00
```

3   Add the contents of the transferred file to the authorized_keys file in this very directory on the Source Node. To do this, log in to the Source Server, change to the /root/.ssh directory, and issue the following command:

```
# cat temp_name >> authorized_keys
```

Now the pbackup/prestore utilities should be able to log in to your Source Nodes as root without having to provide the root password.

Migrating Virtual Machines and Containers

The Parallels physical server is the system with higher availability requirements in comparison with a typical system. If you are running your company mail server, file server, and web server in different virtual machines and Containers on one and the same physical server, then shutting it down for hardware upgrade will make all these services unavailable at once. To facilitate hardware upgrades and load balancing between several Parallels servers, the Parallels Server Bare Metal software provides you with the ability to migrate virtual machines and Containers from one physical server to another.

Parallels Server Bare Metal is shipped with a special utility - `pmigrate` - allowing you to perform different types of migration. Using this utility, you can migrate

- Containers from one physical server to another
- virtual machines from one physical server to another
- a Container to a virtual machine
- a virtual machine to a Container
- a physical server to a virtual machine and Container

All these operations are described in the following subsections.

General Migration Requirements

Before deciding on the type of migration to perform, make sure that the source computer (i.e. the physical computer that you will migrate or that stores the virtual machine and Container before its migration) and the destination computer (i.e. the computer that runs Parallels Server Bare Metal and that will host the resulting virtual machine and Container) meet the requirements below.

Requirements for the Source Computer

The source computer can be a physical computer, a virtual machine, or a Container. The software requirements for source computers are given in the following table:

Operating System	Physical Computer	Virtual Machine	Container
Windows			
Windows 7 (x32, x64)	+	+	-
Windows Server 2003 (x32, x64)	+	+	+
Windows Server 2008 (x32, x64)	+	+	+
Windows 2000 Server (x32)	+	+	-
Windows XP (x32, x64)	+	+	-
Windows Vista (x32, x64)	+	+	-
Linux			
Red Hat Enterprise Linux 5 (x32, x64)	+	+	+
Red Hat Enterprise Linux 4 (x32, x64)	+	+	+
CentOS 5 (x32, x64)	+	+	+
CentOS 4 (x32, x64)	+	+	+
Fedora 11 (x32, x64)	+	+	+
Fedora 10 (x32, x64)	+	+	+
SUSE Linux Enterprise Server 10 (x32, x64)	+	+	+
Debian GNU/Linux 5 (x32, x64)	+	+	+
Debian GNU/Linux 4 (x32, x64)	+	+	+
Ubuntu Linux 9.04 Server (x32, x64)	+	+	+
Ubuntu Linux 8.10 Server (x32, x64)	+	+	+

Note: In the current version of Parallels Server Bare Metal, you cannot migrate Containers running Windows Server 2008 to virtual machines.

Requirements for the destination Server

The destination server must meet the following requirements:

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

Migrating Virtual Machines and Containers Between Parallels Servers

In Parallels Server Bare Metal, you can choose one of the following ways to migrate a virtual machine and Container:

- Migrating a virtual machine and Container using the standard migration technology. In this case there is a short downtime needed to stop and start the virtual machine and Container during its migration from the source server to the destination server.
- Migrating a virtual machine and Container using the zero downtime migration technology. In this case the *stop* and *start* operations are not performed and the migrated virtual machine and Container is restored on the destination server in the same state as it was at the beginning of the migration. This greatly reduces the migration time and puts it on the same footing as the delay caused by a short interruption in the network connectivity.

Note: In this version of Parallels Server Bare Metal, the zero-downtime migration is supported for Containers only.

Both ways are described in the following subsections in detail.

Standard Migration

The standard migration procedure allows you to move stopped, paused, suspended, and running virtual machines and Containers. Migrating a stopped, paused, or suspended virtual machine and Container includes copying all virtual machine and Container-related files from one Parallels server to another and does not differ from copying a number of files from one server to another over the network. In its turn, the migration procedure of a running virtual machine and Container is a bit more complicated and can be described as follows:

1   After initiating the migration process, all virtual machine and Container data are copied to the destination server. During this time, the virtual machine and Container on the source server continues running.

2   The virtual machine and Container on the source server is stopped.

3   The virtual machine and Container data copied to the destination server are compared with those on the source server, and if any files were changed during the first migration step, they are copied to the destination server again and rewrite the outdated versions.

4   The virtual machine and Container on the destination server is started.

There is a short downtime needed to stop the virtual machine and Container on the source server, copy the virtual machine and Container data changes to the destination server, and start the virtual machine and Container on the destination server. However, this time is very short and almost unnoticeable to users.

Note: Before the migration, it might be necessary to detach the Container from its caches. For more information on cached files, see the Cleaning Up Containers subsection (p. 99).

Migrating a Container

The following session moves Container 101 from the current Parallels server to a new one named ts7.test.com:

```
# pmigrate c 101 c root:1qasdeqw3@ts7.test.com/101
root@ts7.test.com's password:
vzmsrc: Connection to destination server (ts7.test.com) is successfully
established
...
Successfully completed
```

The c option in the command above tells pmigrate that you are moving a Container to a Container. If you do not indicate the credentials to log in to the destination server, you will need to do so during the migration.

Important! For the command to be successful, a direct SSH connection (on port 22) should be allowed between the source and destination servers.

By default, after the migration process is completed, the Container private area and configuration file are renamed on the source server by receiving the .migrated suffix. However, if you wish the Container private area on the source server to be removed after the successful Container migration, you can override the default pmigrate behavior by changing the value of the REMOVEMIGRATED variable in the Parallels Server Bare Metal global configuration file (/etc/vz/vz.conf) to yes or by using the -r yes switch with the pmigrate command.

Migrating a Virtual Machine

In its turn, to migrate a virtual machine from the source server to `ts7.test.com`, you need just to specify `v` instead of `c` and the name of the resulting virtual machine instead of Container ID 101:

```
# pmigrate v MyVM v ts7.test.com/MyVM
Migrate the VM MyVM to test.com
root@ts7.test.com's password:
Operation progress 100%
The VM has been successfully migrated.
```

This command moves the `MyVM` virtual machine from the local server to the destination server `ts7.test.com`.

For virtual machines, `pmigrate` also supports the migration from a remote Parallels server to the local one:

```
# pmigrate v ts7.test.com/MyVM v localhost
root@ts7.test.com's password:
Migrate the VM MyVM to localhost
Operation progress 100%
The VM has been successfully migrated.
```

This command move the `MyVM` virtual machine from the `ts7.test.com` server to the local server.

Note: For more information on options that you can pass to `pmigrate`, refer to the *Parallels Command Line Reference*.

Zero-Downtime Migration

The `pmigrate` utility also allows you to migrate your Containers from one Parallels server to another with zero downtime. The zero downtime migration technology has the following main advantages as compared with the standard one:

- The process of migrating a Container to another Parallels server is transparent for you and the Container applications and network connections. This means that on the source and destination servers, no modifications of system characteristics and operational procedures inside the Container are performed.

- The Container migration time is greatly reduced. In fact, the migration eliminates the service outage or interruption for Container end users.

- The Container is restored on the destination server in the same state as it was at the beginning of the migration.

- You can move the Containers running a number of applications which you do not want to be rebooted during the migration for some reason or another.

Note: Zero-downtime migration cannot be performed on Containers having one or several opened sessions established with the `pctl enter` command.

Before performing zero-downtime migration, it is recommended to synchronize the system time on the source and destination servers, for example, by means of NTP (http://www.ntp.org). The reason for this recommendation is that some processes running in the Container might rely on the system time being monotonic and thus might behave unpredictably if they see an abrupt step forward or backward in the time once they find themselves on the new server with different system clock parameters.

In the current version of Parallels Server Bare Metal, you can make use of the following types of zero-downtime migration:

- *Simple online migration*. In this case a Container is 'dumped' at the beginning of the migration, i.e. all Container private data including the state of all running processes are saved to an image file. This image file is then transferred to the destination server where it is 'undumped'.

- *Lazy online migration*. Using this type of online migration allows you to decrease the size of the 'dumped' image file storing all Container private data and transferred to the destination server by leaving the main amount of memory in a locked state on the source server and swapping this memory from the source server on demand. Thus, the migrated Container can be started before the whole memory is transferred to the destination server, which drastically reduces the service delay of the corresponding Container. When a process tries to access a page of memory that has not yet been migrated, the request is intercepted and redirected to the source server where this page is stored.

- *Iterative online migration*. In this case the main amount of Container memory is transferred to the destination server before a Container is 'dumped' and saved to an image file. Using this type of online migration allows you to attain the smallest service delay.

- *Iterative + lazy online migration*. This type of online migration combines the techniques used in both the *lazy* and *iterative* migration types, i.e. some part of Container memory is transferred to the destination server before 'dumping' a Container and the rest is transported after the Container has been successfully 'undumped' on the server.

To migrate a Container by using the zero downtime migration technology, you should pass the --online option to the pmigrate utility. By default, the *iterative online migration* type is used to move a Container from one Parallels server to another. For example, you can migrate Container 101 from the current server to the destination server named my_node.com by executing the following command:

Note: If the CPU capabilities on the source server exceed those on the destination server (e.g. you migrate from a source server running the Pentium 4 processor to a destination server running the Pentium 3 processor), the migration may fail and you will be presented with the corresponding warning message. However, if you are sure that the CPU power on the destination server is sufficient to start and run the Container(s) being migrated, you can use the -f option to force the migration process.

```
# pmigrate c 101 c --online --require-realtime my_node.com
Enter password:
Connection to destination server (192.168.1.57) \
is successfully established
Moving/copying Container#101 -> Container#101, [], [] ...
Syncing private area '/vz/private/101'
 - 100% |**************************************
done
Suspending Container#101 ...
done
Dumping Container#101 ...
done
...
Migration completed
```

The --require--realtime option tells pmigrate to move the Container by using the *iterative online migration* type only. So, if this migration type cannot be carried out for some reason or other, the command will fail and exit. If this option is omitted and in the case of failure while performing the iterative migration, pmigrate will try to move your Container by means of the *simple online migration* type or the *lazy online migration* type (if the --lazy option is given). You can specify more than one Container ID simultaneously; in this case, all specified Containers will be moved to a new Parallels server one by one.

If you wish to use another migration type for moving your Containers to another server, you should additionally pass certain options to pmigrate:

- Specify the --noiter option to migrate a Container by using the *simple online migration* type;

- Specify the --noiter and --lazy options to migrate a Container by using the *lazy online migration* type;

- Specify the --lazy option to migrate a Container by using the *iterative + lazy online migrate* type.

Migrating a Container to a Virtual Machine

The `pmigrate` utility allows you to migrate Containers to virtual machines. The source server, i.e. the server where the Container resides before its migration, can be one of the following:

- a local server running Parallels Server Bare Metal
- a remote server running Parallels Server Bare Metal
- a remote server running Parallels Virtuozzo Containers

Currently, the destination server, i.e. the server where the resulting virtual machine will be created, can be only a local server with Parallels Server Bare Metal.

The process of migrating a Container to a virtual machine differs depending on whether the server where the Container resides is running the Parallels Server Bare Metal or Parallels Virtuozzo Containers software.

Migrating a Container from a Parallels Server

You can use the `pmigrate` utility to migrate Containers that reside on both local and remote servers running Parallels Server Bare Metal. When migrating a Container from a local server, you only need to specify the Container ID and the name of the resulting virtual machine. For example, the following command migrates Container 101 to the `VM_101` virtual machine on the same Parallels server:

```
# pmigrate c 101 v VM_101
Connecting to local agent...
Querying configuration...
Migrating...
Operation progress 100%
Registering VM...
PVC to VM /var/parallels/VM_101.pvm/config.pvs migration succeeded.
```

The resulting virtual machine will be put to the `/var/parallels` directory on the destination server.

If you want to migrate a Container from a remote Parallels server, you should additionally indicate the source server IP address and the credentials of the root user on this server:

```
# pmigrate c root:8uhytv4@192.168.12.12/101 v VM_101
Connecting to local agent...
Querying configuration...
Migrating...
Operation progress 100%
Registering VM...
PVC to VM /var/parallels/VM_101.pvm/config.pvs migration succeeded.
```

This command migrates Container 101 residing on the Parallels server with the IP address of `192.168.12.12` to the `VM_101` virtual machine on the local server. If you do not specify the root credentials on the source server, you will be asked to do so during the command execution.

Migrating a Container from a Server with Parallels Virtuozzo Containers

You can use the `pmigrate` utility to migrate Containers that reside on remote servers running the following versions of the Parallels Containers software:

- Parallels Virtuozzo Containers 4.0 for Linux with update TU-4.0.0-464 or higher
- Parallels Virtuozzo Containers 4.5 for Windows

Moving a Container from a remote Parallels Containers server to a virtual machine on the local server with Parallels Server Bare Metal involves completing the following steps:

1   Installing the Parallels agent on the server with Parallels Containers.

2   Running the `pmigrate` utility on the destination server to migrate the Container.

Installing the Parallels for Containers Agent

First, you must install the Parallels agent on the source Parallels Containers server. During migration, this agent collects essential information on the Container to be moved and transfers it to the `pmigrate` utility on the destination server. To install the Parallels agent, do the following:

1   Log in to the source Parallels Containers server as a user with administrative rights.

2   Copy the Parallels agent installation file to the source server. The installation file is located in the `/usr/share/pmigrate/tools` directory on the server with Parallels Server Bare Metal:

- `parallels-transporter-for-containers-XXXX.run`. Use this file to install the Parallels agent on servers running Parallels Virtuozzo Containers 4.0 for Linux.

- `ParallelsTransporterForContainers-parallels-XXXX.exe`. Use this file to install the Parallels agent on servers running Parallels Virtuozzo Containers 4.5 for Windows.

3   Execute the installation file on the source server.

4   Follow the instructions of the wizard to install the Parallels agent.

Migrating the Container

Once the Parallels agent is installed, you can use the `pmigrate` utility to move a Container to a virtual machine. For example, you can run the following command on the destination server to migrate Container 101 from the remote server with IP address `192.168.12.12` to the `VM_101` virtual machine:

```
# pmigrate c root:8uhytv4@192.168.12.12/101 v VM_101
Connecting to local agent...
Querying configuration...
Migrating...
Operation progress 100%
Registering VM...
PVC to VM /var/parallels/VM_101.pvm/config.pvs migration succeeded.
```

The resulting virtual machine will be put to the `/var/parallels` directory on the destination server. If you do not specify the administrative credentials on the source server (for `root` on Linux servers and `Administrator` on Windows servers), you will be asked to do so during the command execution.

Migrating a Physical Computer to a Virtual Machine and Container

You can also use the `pmgirate` utility to move a stand-alone physical computer to a virtual machine and Container. The migration process includes copying the whole contents of the physical computer (i.e. all its files, directories, quota limits, configuration settings, and so on) to a virtual machine and Container on the Parallels server. After migrating the computer, you will have its exact copy in a virtual machine and Container including the operating system, the IP addresses assigned, the amount of available disk space and memory, etc.

Moving a physical computer to a virtual machine and Container involves completing the following steps:

1   Installing the Parallels agent on the physical computer you want to migrate. This step is required only if you are migrating the physical computer to a virtual machine.

2   Migrating the physical computer by running the `pmigrate` utility on the server.

Installing the Agent

If you are planning to migrate a physical computer to a virtual machine, you must first install the Parallels agent on this computer. This agent collects essential system data on the physical computer and transfers it to the `pmigrate` utility on the Parallels server. To install the Parallels agent, do the following:

1   Make sure that your physical computer meets the necessary requirements for installing the Parallels agent. See **Requirements for Migrating to Virtual Machines** (p. 62) for details.

2   Log in to the physical computer as a user with administrative rights.

3   Copy the Parallels agent installation file to the physical computer. The installation file is located in the `/usr/share/pmigrate/tools` directory on the Parallels server:

- `parallels-transporter-agent-XXXX.run`. Use this file to install the Parallels agent on computers running a Linux operating system.

- `ParallelsTransporterAgent-parallels-XXXX.exe`. Use this file to install the Parallels agent on computers running a Windows operating system.

4   Execute the installation file on the physical computer.

5   Follow the instructions of the wizard to install the Parallels agent.

6   Restart the source computer to complete the installation.

Note: The Parallels agent is automatically launched after the restart, so you do not need to start it manually.

Migrating the Server

Once the physical computer is up and running, you can migrate to a virtual machine and Container on the Parallels server. For example, you can move a physical computer to a virtual machine by running the following command on the destination server:

```
# pmigrate h root:1qsde34rt@192.168.1.130 v MyVM
```

where

- `h` denotes that you are migrating a physical computer.

- `root:1qsde34rt@192.168.1.130` is the IP address and credentials of the physical computer to be migrated.

  You can omit the credentials in the command above. In this case you will be asked to provide them during the command execution.

- `v` indicates that the physical computer is to be moved to a virtual machine.

- `MyVM` is the name of the resulting virtual machine on the Parallels server.

Once the command is complete, you will find the resulting virtual machine in the `/var/parallels` directory on the Parallels server.

If you want to migrate the same physical computer to a Container, just specify `c` instead of `v` and the ID of the resulting Container (e.g. 101) instead of `MyVM.` For example, the following command will migrate the physical computer to Container 101:

```
# pmigrate h root:1qsde34rt@192.168.1.130 c 101
```

Note: In the current version of Parallels Server Bare Metal, you cannot migrate physical computers running a Windows operating system to Containers.

Requirements for Migrating to Containers

To avoid delays and problems when migrating a physical server to a Container, make sure that the following requirements are fulfilled:

Migrating to Containers on Linux servers:

- The Linux distribution installed on the physical server is supported by Parallels Server Bare Metal. To find out if your Linux distribution can be recognized by Parallels Server Bare Metal, you can check the `/etc/vz/conf/dists` directory on the Parallels server and look for the configuration file of your Linux distribution. It should have the name of `Linux_Distribution_Name-version.conf` where `Linux_Distribution_Name` and `version` denote the name of the Linux distribution and its version, respectively (e.g. `redhat-5.conf`). If there is no corresponding distribution in the directory, you can do one of the following:

  - Create a new distribution configuration file and place it to the `/etc/vz/conf/dists` directory on the Parallels server. Detailed information on how to create new configuration files is provided in the **Creating Configuration Files for New Linux Distribution** section (p. 193).

  - Start the migration process without having the right configuration file for your Linux distribution. In this case the `unknown.conf` distribution configuration file from the `/etc/vz/conf/dists` directory will be used for tuning the Container after the physical server migration. However, using the `unknown.conf` configuration file means that you will not be able to use standard Parallels Server Bare Metal utilities (e.g. `pctl`) for performing the main operations on the created Container (such as setting the Container IP address or configuring the DNS parameters) and have to manually complete these tasks from inside the Container.

- `ssh` is installed on both the physical server and the Parallels. `ssh` is used to provide secure encrypted and authenticated communication for both physical servers. You can check if the `ssh` package is already installed on the server by executing the `ssh -V` command.

- `rsync` is installed on the physical server. `rsync` is used to copy the physical server contents to the Container. If the physical server `rsync` happens to be incompatible with the Parallels server, use the statically linked `rsync` from the `/usr/local/share/vzlinmigrate` directory on the physical server as well.

Migrating to Containers on Windows servers:

- The source and destination physical servers must have the same major and minor versions of Windows Server 2003 and 2008 and Service Pack, if any.

- The `Server` and `Remote Registry` services are running on the source server.

- The `Volume Shadow Copy` and `Microsoft Software Shadow Copy Provider` services are enabled on the source server.

- The default administrative shares (especially, `ADMIN$`) are enabled on the physical server.

- The following ports are opened on the physical server:

  - standard Windows Server ports used to access the physical server via the network sharing and remote registry capabilities (e.g. 445, 137, 138)

  - Parallels Containers-specific ports: 4433, 4434, 4435

- We also recommend that you disable your antivirus program on the physical server before migrating it to a Container on the Hardware Node.

Migration Restrictions for Containers

Listed below are the limitations you should take into account when deciding on the migration process.

Migrating to Containers on Linux servers:

- During the migration, all the filesystems available on your physical server are joined to one filesystem inside the Container - VZFS (Virtuozzo File System). Detailed information on VZFS is provided in the **Virtuozzo File System** subsection (p. 15).

- If there are several IP addresses assigned to the physical server, all these IP addresses will be reassigned to one and the same device on the Parallels server - venet0. This virtual network adapter is used to connect all the Containers on the given Parallels server among themselves and with the server. After the migration, you can create additional virtual network adapters inside the Container and decide what IP address to be assigned to what network adapter. For detailed information on how to create and manage Container virtual network adapters, turn to **Managing Adapters in Containers** (p. 142).

- During the migration process, you may specify only one partition on the physical server which will be migrated to the Container together with all quotas imposed on it. All the other partitions of the server will be copied without keeping their quota limits. Moreover, the quota limits imposed on the selected partition will be applied to the entire Container after the server migration.

- While migrating your physical server running a Linux operating system with the security-enhanced (SE) Linux kernel, keep in mind that the SE Linux kernel is currently not supported by Parallels Server Bare Metal. Therefore, the Container where the server running the SE Linux distribution has been migrated will not support the SE security features.

- If any of your files and/or directories on the physical server have extended attributes associated with them, these attributes will be lost after the server migration.

- Raw devices on the physical server cannot and will not be migrated to the Container on the Parallels server.

- If you are running an application which is bound to the physical server MAC address, you will not be able to run this application inside the Container after the server migration. In this case, you can do one of the following:

    - If you are running a licensed application, you should obtain a new license and install the application inside the Container anew.

    - If you are running a non-licensed application, you can try to reconfigure the application and to make it work without being bound to any MAC address.

- If the migration process fails on the step of transferring files and directories from the physical server to the Container by means of rsync, the /vz/private/*CT_ID* directory on the Parallels server will contain all the copied files and directories and may occupy a great amount of disk space. You can keep the directory, which will greatly speed up the repeated migration procedure, or manually remove the directory by using the rm utility.

Migrating to Containers on Windows servers:

- The following migration types are supported:

    - Physical servers running Windows Server 2008 can be migrated only to destination servers running Windows Server 2008.

    - Physical servers running Windows Server 2003 can be migrated only to destination servers running Windows Server 2003.

- Non-NTFS volumes cannot be migrated.
- After the physical server migration, the Quality of Service packet scheduler is disabled inside the Container irrespective of its state on the server before the migration began.
- You cannot migrate physical servers running the 32-bit version of Windows Server to destination servers running the 64-bit version of Parallels Containers, neither can you move physical servers running the 64-bit version of Windows Server to servers running the 32-bit version of Parallels Containers.

Requirements for Migrating to Virtual Machines

Any physical computer that you plan to migrate to a virtual machine must have the Parallels agent installed. The agent can be installed on computers meeting the following requirements.

Hardware Requirements

- 700 (or higher) MHz x86 or x64 processor (Intel or AMD).
- 256 MB or more RAM.
- 50 MB of hard disk space for installing the Parallels agent package.
- Ethernet or WiFi network adapter.

Software Requirements

For software requirements, see the table in General Migration Requirements (p. 50).

Additional Requirements for Migrating Servers with Parallels Server Bare Metal

If you plan to migrate a server running the Parallels Server Bare Metal software, you should first make sure that the snapapi26 and snumbd26 modules are not loaded on the server. You can use the following command to check this:

```
# lsmod | grep snapapi26
# lsmod | grep snumbd26
```

If any of these modules are loaded, unload them by running the rmmod command. If the rmmod command fails to unload any of these modules, do the following:

1 Stop the Parallels Server Bare Metal service:

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

2 Unload the modules:

```
# rmmod snapapi26
# rmmod snumbd26
```

3 Start the Parallels Server Bare Metal service again:

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

Once the modules are unloaded, proceed with migrating the server.

Notes:

1. Migrating Windows dynamic volumes and Linux logical volumes (LVM) is not supported.

2. You may also try to migrate servers with unsupported file systems. However, in this case all disk sectors are copied successively, and you may experience problems with using the resulting virtual machine.

Migrating a Virtual Machine to a Container

The process of migrating a virtual machine to a Container on the Parallels server is the same as migrating a physical computer to a Container. For example, you can execute the following command to move a virtual machine with the IP address of 192.168.1.130 to Container 101 on your Parallels server:

```
# pmigrate h root:1qsde34rt@192.168.1.130 c 101
```

You can omit the virtual machine credentials in the command above. In this case you will be asked to provide them during the command execution.

Notes:

1. For more information on migrating physical computers to Containers, refer to Migrating a Physical Computer to a Virtual Machine and Container (p. 58).

2. The requirements a virtual machine must meet are the same as for migrating physical computers and are described in Requirements for Migrating to Containers (p. 60).

Performing Container-Specific Operations

This section provides the description of operations specific for your Containers.

Setting Name for Container

You can assign an arbitrary name to your Container and use it, along with the Container ID, to refer to the Container while performing this or that Container-related operation on the server. For example, you can start or stop a Container by specifying the Container name instead of its ID.

You can assign names to your Containers using the `--name` option of the `pctl set` command. For example, to set the `computer1` name for Container 101, you should execute the following command:

```
# pctl set 101 --name computer1 --save
Name computer1 assigned
Saved parameters for Container 101
```

You can also set a name for Container 101 by editing its configuration file. In this case you should proceed as follows:

1   Open the configuration file of Container 101 (`/etc/vz/conf/101.conf`) for editing and add the following string to the file:

```
NAME="computer1"
```

2   In the `/etc/vz/names` directory on the server, create a symbolic link with the name of `computer1` pointing to the Container configuration file. For example:

```
# ln --symbolic /etc/vz/conf/101.conf /etc/vz/names/computer1
```

When specifying names for Containers, please keep in mind the following:

- Names may contain the following symbols: `a-z`, `A-Z`, `0-9`, underscores (`_`), dashes (`-`), spaces, the symbols from the ASCII character table with their code in the 128 - 255 range, and all the national alphabets included in the Unicode code space.

- Container names cannot consist of digits only; otherwise, there would be no way to distinguish them from Container IDs.

- If it contains one or more spaces, the Container name should be put in single or double quotes.

After the name has been successfully assigned to Container 101, you can start using it instead of ID 101 to perform Container-related operations on the Node. For example:

- You can stop Container 101 with the following command:

```
# pctl stop computer1
Stopping Container ...
Container was stopped
Container is unmounted
```

- You can start Container 101 anew by issuing the following command:

```
# pctl start computer1
Starting Container ...
...
```

You can find out what name is assigned to Container 101 in one of the following ways:

- Using the `vzlist` utility:

```
# vzlist -o name 101
NAME
computer1
```

- Checking the NAME parameter in the Container configuration file (/etc/vz/conf/101.conf). For example:

```
# grep NAME /etc/vz/conf/101.conf
NAME="computer1"
```

- Checking the NAME parameter in the /etc/vz/names/computer1 file which is a symlink to the Container configuration file. For example:

```
# grep NAME /etc/vz/names/computer1
NAME="computer1"
```

Moving Container Within the Parallels Server

The `vzmlocal` utility allows you to move Containers within your server. Moving a Container within one and the same server consists in changing the Container ID and its private area and root paths. You can use `vzmlocal` to change the ID of the corresponding Container only or to additionally modify its private area and root path.

Let us assume that you wish to change the ID of your Container from 101 to 111 and modify its private area and root paths from /vz/private/101 to /vz/private/my_dir and from /vz/root/101 to /vz/root/ct111, respectively. To do this, execute the following command on the server:

```
# vzmlocal 101:111:/vz/private/my_dir:/vz/root/ct111
Moving/copying Container#101 -> Container#111,
[/vz/private/my_dir], [/vz/root/ct111] ...
...
Successfully completed
```

To check if Container 101 has been successfully moved to Container 111, you can use the following commands:

```
# vzlist -a
CTID      NPROC STATUS   IP_ADDR          HOSTNAME
   1          43 running 10.0.10.1        localhost
 111          - stopped 10.0.10.101      myContainer
# ls /vz/private
1  my_dir
# ls /vz/root
1  ct111
```

As can be seen from the example above, the ID of Container 101 has been changed to 111, its private area is now located in the /vz/private/my_dir directory on the server, and the path to its root directory is /vz/root/ct111.

Notes:

1. You can perform a number of moving operations by a single invocation of the `vzmlocal` utility.

2. You can run the `vzmlocal` utility on both running and stopped Containers.

Disabling Container

There may appear situations when you wish to forbid Container owners to use their Containers. For example, it may happen in case the Container owner uses it for unallowed purposes: intruding into computers of other users, participating in DoS attacks, etc.

In such cases, you can disable a Container, thus, making it impossible to start the Container once it was stopped. For example, you can execute the following command to disable Container 101 residing on your server:

```
# pctl set 101 --disabled yes
```

After the Container stopping, the Container user will not be able to start it again until you enable this Container again by passing the --disabled no option to pctl set. You can also use the --force option to start any disabled Container. For example:

```
# pctl start 101
Container start disabled
# pctl start 101 --force
Starting Container...
Container is mounted
Adding port redirection to Container(1): 4643 8443
Adding IP address(es): 10.144.144.101
Hostname for Container set: Container_101
Container start in progress...
```

Reinstalling Container

Reinstalling a Container is used if a Container administrator has inadvertently modified, replaced, or deleted any file that is part of an application or OS template, which has brought about the Container malfunction. You can reinstall the Container in the two following ways:

1   The `pctl recover` command restores the original VZFS symlinks of the Container private area to the OS and/or application template(s) as they were at the time when the Container was created and/or when the application template(s) were added to the Container. This command does not deal with any user files on the Container:

```
# pctl recover 101
Optimizing Container private area...
vzquota : (warning) Quota is running for id 101 already
Setting quota ...
Container is mounted
Setup slm memory limit
Setup slm subgroup (default)
Container is unmounted
Recover OS template: redhat-el5-x86
Creating Container private area (redhat-el5-x86)
...
Recovering Container completed successfully
```

2   The `pctl reinstall` command creates a new private area for the problem Container from scratch using its configuration files and its OS and application templates. Thus, a clean working copy of the Container is created:

```
# pctl reinstall 101
Optimizing Container private area...
Calculating Container disk usage...
Creating Container private area (redhat-el5-x86)
Starting Container ...
Initializing quota...
Container is mounted
Setup slm memory limit
Setup slm subgroup (default)
Container start in progress...
Calculating Container disk usage...
Copying Container credentials...
Stopping Container ...
Container was stopped
Container is unmounted
Old Container file system has been moved to /old
Initializing quota...
Container reinstallation completed successfully
```

Note: If any of the Container application templates cannot be added to the Container in a normal way, the reinstallation process will fail. This may happen, for example, if an application template was added to the Container using the `--force` option of the `vzpkgadd` or `vzpkg install` command (for more information on these commands, see the *Parallels Command Line Reference Guide*).

In order to retain the personal data inside the old Container, the utility also copies the contents of the old private area to the `/old` directory of the new private area (unless the `--skipbackup` option is given). The personal data can then be copied to the corresponding directories of the new private area and the `/old` directory eventually deleted:

```
# pctl start 101
Starting Container ...
Container is mounted
Setup slm memory limit
```

```
Setup slm subgroup (default)
Setting devperms 20002 dev 0x7d00
Adding port redirection to Container(1): 4643 8443
Adding IP address(es) to pool:
Adding IP address(es): 10.14.14.101
Hostname for Container set: localhost.localdomain
Container start in progress...
# pctl exec 101 ls /
bin
boot
dev
[...other directories...]
old
[...other directories...]
tmp
usr
var
```

Both the `pctl recover` and `pctl reinstall` commands retain the users' credentials base, unless the `--resetpwdb` option is specified.

Customizing Container Reinstallation

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

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

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

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

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

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

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

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

1 Remove the `vps.reinstall` and `vps.configure` scripts from the `/etc/vz/conf` directory, or at least rename them;

2 Modify the last line of the `vps.reinstall` script so that it would read

```
exit 128
```

instead of

```
exit 0
```

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

Performing Virtual Machine-Specific Operations

This section focused on operations specific for your virtual machines.

Pausing a Virtual Machine

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

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

```
# pctl pause My_VM
Pause the VM...
The VM has been successfully paused.
```

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

```
# pctl list -a
STATUS    IP_ADDR        NAME
running   10.10.10.101   101
paused    10.10.10.201   My_VM
```

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

```
# pctl start My_VM
Starting the VM...
The VM has been successfully started.
```

Managing Snapshots

In Parallels Server Bare Metal, you can save the current state of a virtual machine by creating a snapshot. You can then continue working in your virtual machine and return to the saved state any time you wish. For example, you can make use of snapshots in the following cases:

- You are going to configure an application with a lot of settings. In this case, you may first wish to play with settings before applying them to your application. So, you create a snapshot before starting to experiment with the application settings.
- You are involved in a large development project. In this case, you may wish to mark milestones in the development process by creating a snapshot after each milestone. If anything goes wrong, you can easily revert to the previous milestone and start the development anew.

In Parallels Server Bare Metal, you can manage snapshots as follows:

- create a new snapshot of a virtual machine
- list the existing snapshots of a particular virtual machine
- revert to a snapshot
- remove a snapshot

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

Creating a Snapshot

To create a snapshot of a virtual machine in Parallels Server Bare Metal, you can use the `pctl snapshot` command. For example, you can execute the following command to create a snapshot of the `MyVM` virtual machine:

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

A newly created snapshot is saved to the `/vz/VM_Name.pvm/Snapshots/Snapshot_ID.pvs` file where `VM_Name` denotes the name of the corresponding virtual machine and `Snapshot_ID` is a random ID assigned to the snapshot. In the command above, the snapshot is assigned the ID of `{12w32198-3e30-936e-a0bbc104bd20}` and saved to the `/vz/MyVM/Snapshots/{12w32198-3e30-936e-a0bbc104bd20}.pvs` file.

```
# ls /vz/MyVM.pvm/Snapshots/
{063615fa-f2a0-4c14-92d4-4c935df15840}.pvc
```

The ID assigned to the snapshot can be used to manage this snapshot (e.g. get detailed information on the snapshot or delete it) .

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

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

You can then view the set name and description in the `/vz/MyVM/Snapshots.xml` file or in Parallels Management Console.

When working with snapshots, keep in mind the following:

- If a virtual machine name contains spaces, use quotation marks to specify the name in `pctl` commands (e.g. "Windows XP").

- Before creating a snapshot, it is recommended that you complete all operations of installing, downloading, or writing to external devices. You should also complete or cancel any transactions performed via the virtual machine in external databases.

Creating Branches

The branches are created when you do the following:

1 Create several sequential snapshots.

2 Revert to an intermediate snapshot.

3 Make some changes to the virtual machine.

4 Save the virtual machine state by creating a new snapshot.

In this case, the newly created snapshot will start a new branch using the intermediate snapshot from Step 2 as the baseline.

Listing Snapshots

To list all snapshots of a particular virtual machine, use the `pctl snapshot-list` command:

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

This command shows that currently two snapshots exist for the `MyVM` virtual machine. The snapshot with ID {063615fa-f2a0-4c14-92d4-4c935df15840} is based on the snapshot with ID {989f3415-3e30-4494-936e-a0bbc104bd20}, i.e. the latter acts as the parent for the snapshot with ID {063615fa-f2a0-4c14-92d4-4c935df15840}. The * sign before {063615fa-f2a0-4c14-92d4-4c935df15840} denotes that this is the current snapshot for the given virtual machine.

You can also view the relationship between snapshots by specifying the -t option:

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

The command output shows you that currently 2 branches exist for the `MyVM` virtual machine. The snapshot with ID {989f3415-3e30-4494-936e-a0bbc104bd20} is the baseline used as a starting point for these branches.

You can get detailed information on a particular snapshot using the -i option and specifying the snapshot ID:

```
# pctl snapshot-list MyVM -i {063615fa-f2a0-4c14-92d4-4c935df15840}
ID: {063615fa-f2a0-4c14-92d4-4c935df15840}
Name: Clean_System
Date: 2009-07-22 22:39:06
Current: yes
State: power_off
Description: <![CDATA[This snapshot was created right after installing Windows
XP operating system]]>
```

The `pctl snapshot-list` command displays the following information about snapshots:

Field	Description
ID	The ID assigned to the snapshot.
Name	The name assigned to the snapshot.
Date	The date and time when the snapshot was created.
Current	Denotes whether this is the current snapshot of the virtual machine.
State	The state the virtual machine was in at the time you took the snapshot.
Description	The description set for the snapshot.

Reverting to a Snapshot

You can use the `pctl snapshot-switch` command to revert to a snapshot. When you revert to a snapshot, the current state of the virtual machine is discarded, and all changes made to the system since the previous snapshot are lost. So, before returning to a specific snapshot, you may first wish to save these states by creating a new snapshot. Refer to the **Creating a Snapshot** subsection (p. 72) for information on how you can do it.

The `pctl snapshot-switch` command requires the virtual machine name and the snapshot ID to be specified as arguments:

```
pctl snapshot-switch "Windows XP" --id {cedbc4eb-dee7-42e2-9674-89d1d7331a2d}
Switch to the snapshot...
The VM has been successfully switched.
```

This command restores the snapshot with ID {cedbc4eb-dee7-42e2-9674-89d1d7331a2d} for the `Windows XP` virtual machine.

Deleting a Snapshot

In Parallels Server Bare Metal, you can use the `pctl snapshot-delete` command to delete those snapshots that you do not need any more. Assuming that you want to delete the snapshot with ID {903c12ea-f6e6-437a-a2f0-a1d02eed4f7e} for the `MyVM` virtual machine, you can run this command:

```
# pctl snapshot-delete MyVM --id {903c12ea-f6e6-437a-a2f0-a1d02eed4f7e}
Deleting the snapshot...
The snapshot has been successfully deleted.
```

When you delete a parent snapshot, its children are not deleted, and the information the parent snapshot contains is merged into them.

For example, the following session demonstrates the process of deleting the snapshot with ID {903c12ea-f6e6-437a-a2f0-a1d02eed4f7e} acting as a parent for another snapshot:

```
# pctl snapshot-list MyVM
PARENT_SNAPSHOT_ID                       SNAPSHOT_ID
                                         {989f3415-3e30-4494-936e-a0bbc104bd20}
{989f3415-3e30-4494-936e-a0bbc104bd20}  {063615fa-f2a0-4c14-92d4-4c935df15840}
{063615fa-f2a0-4c14-92d4-4c935df15840} *{58c9941e-f232-4273-892a-82e836536889}
# pctl snapshot-delete MyVM --id {903c12ea-f6e6-437a-a2f0-a1d02eed4f7e}
Deleting the snapshot...
The snapshot has been successfully deleted.
# pctl snapshot-list MyVM
PARENT_SNAPSHOT_ID                       SNAPSHOT_ID
                                         {063615fa-f2a0-4c14-92d4-4c935df15840}
{063615fa-f2a0-4c14-92d4-4c935df15840} *{58c9941e-f232-4273-892a-82e836536889}
```

Managing Templates

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

In Parallels Server Bare Metal, you can perform the following operations on templates:

- create a new template
- list the existing templates
- create a virtual machine from a template

These operations are described in the following subsections in detail.

Create a Template

In Parallels Server Bare Metal, you can create a virtual machine template using the `pctl clone` utility. Making a template may prove useful if you need to create several virtual machines with the same configuration. In this case, your steps can be as follows:

1 You create a virtual machine with the required configuration.

2 You make a template on the basis of the created virtual machine.

3 You use the template to create as many virtual machines as necessary.

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

```
# pctl clone My_VM --name template1 --template
Clone the My_VM VM to VM template template1...
Operation progress 98%
The VM has been successfully cloned.
```

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

Listing Templates

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

```
# pctl list -t
{4ad11c28-9f0e-4086-84ea-9c0487644026}  win-2003        template1
{64bd8fea-6047-45bb-a144-7d4bba49c849}  rhel            template3
{6d3c9d6f-921a-484d-9772-bc7096f68df1}  win-2003        template2
```

In this example, 3 virtual machine templates exist on the Parallels server. The information on these templates is presented in the form of a table with the following columns (from left to right): the template ID, the operating system contained in the template, and the template name.

Deploying a Template

Though a template has the same components as a virtual machine (hardware, software, etc.), it cannot be started. To run a template as a virtual machine, you need first to convert the template. By converting a template, you create a virtual machine with the configuration identical to that of the template.

To convert a template into a virtual machine, use the `--ostemplate` option of the `pctl create` command. For example, to convert the `template1` template to a virtual machine with the `Converted_VM` name, you can run this command:

```
# pctl create Converted_VM --ostemplate template1
Creating the VM on the basis of the template1 template...
Clone the template1 VM to the VM Converted_VM...
Operation progress 99%
The VM has been successfully cloned.
```

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

```
# pctl list -a
STATUS          IP_ADDR          NAME
running         10.12.12.101     111
stopped         10.12.12.34      Converted_VM
running         10.30.17.149     Windows XP
```

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

```
# pctl list -t
{4ad11c28-9f0e-4086-84ea-9c0487644026}  win-2003        template1
{64bd8fea-6047-45bb-a144-7d4bba49c849}  rhel            template2
```

Managing Virtual Machine Disks

In Parallels Server Bare Metal, you can manage virtual machine disks as follows:

- change the type of your virtual disks
- increase the capacity of your virtual disks
- reduce the capacity of your virtual disks
- reduce the size occupied by your virtual disks on the physical hard drive

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

Changing the Disk Type

A virtual disk can be one of the two types:

- `plain`. A plain virtual hard disk has a fixed size from the moment of its creation.
- `expanding`. An expanding virtual hard disk is small initially. Its size grows as you add applications and data to it.

To change the type of a virtual disk in Parallels Server Bare Metal, you can use the `pctl set` command. Let us assume that the current type of the `hdd0` virtual disk in the `MyVM` virtual machine is plain and you want to change it to expanding. In this case, you can execute the following command:

```
# pctl set MyVM --device-set hdd0 --type expand
```

To change the disk type back to plain, run this command:

```
# pctl set MyVM --device-set hdd0 --type plain
```

Increasing the Virtual Disk Capacity

If you find that the capacity of your virtual machine's hard disk does not fit your needs anymore, you can increase it using the `prl_disk_tool` utility. For example, you can execute the following command to set the capacity for the `MyVM-0.hdd` disk to 80 GB:

```
# prl_disk_tool resize --size 80000 --hdd /vz/MyVM.pvm/MyVM-0.hdd/
Operation progress 100%
```

This command adds additional disk space as unallocated space. You can use standard means (e.g. the Disk Management tool in Windows-based virtual machines) to allocate this space by creating a new partition or expanding an existing on.

At the same time, you can use the `--resize_partition` option to automatically add additional space to the last partition on the virtual disk:

```
# prl_disk_tool resize --size 80000 --hdd /vz/MyVM.pvm/MyVM-0.hdd/ --
resize_partition
Operation progress 100%
```

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

- You cannot increase the capacity of a virtual disk if the virtual machine using this disk is running.
- The virtual machine using the virtual disk you want to configure must not have any snapshots. Otherwise, the operation will fail:

```
# prl_disk_tool resize --size 68000 --hdd /vz/MyVM.pvm/MyVM-0.hdd/
This disk has one or more snapshots and cannot be resized.
You need to delete snapshots using the pctl tool before resizing the disk.
```

  In this case, you should delete all existing snapshots and run the command again. To learn how to delete virtual machine snapshots, refer to **Deleting a Snapshot** (p. 74).

- The capacity of an expanding virtual disk shown from inside the virtual machine and the size the virtual disk occupies on the server's physical disk may differ.

Reducing the Virtual Disk Capacity

Parallels Server Bare Metal provides a possibility to reduce the size of an expanding virtual disk by setting the limit the disk cannot exceed. In general, the process of reducing a virtual disk includes these steps:

1   Finding out the minimum capacity to which the disk can be reduced.

2   Running the `prl_dsk_tool resize` command to reduce the disk.

Checking the Minimum Disk Capacity

Before reducing a virtual disk, you may wish to see the minimum capacity to which it can be reduced. To do this, use the `prl_disk_tool resize --info` command. For example, you can run the following command to get detailed information on the `MyVM-0.hdd` disk:

```
#  prl_disk_tool resize --info --hdd /vz/MyVM.pvm/MyVM-0.hdd
Operation progress 100 %
Disk information:
        Size:                                     65537M
        Minimum:                                   2338M
        Minimum without resizing the last partition:   65523M
```

The information on the virtual disk is presented in the form of the following table:

Column Name	Description
Size	The virtual disk disk capacity, in megabytes, as it is seen from inside the virtual machine.
Minimum	The virtual disk capacity, in megabytes, after resizing the disk using the `prl_disk_tool` utility with the `--resize_partition` option.
Minimum without resizing the last partition	The virtual disk capacity, in megabytes, after resizing the disk using the `prl_disk_tool` utility without the `--resize_partition` option.

Reducing the Disk Size

Once you know the minimum capacity of the virtual disk, you can start reducing it. For example, to reduce the `MyVM-0.hdd` disk to 30 GB, you can execute the following command:

```
#  prl_disk_tool resize --size 30G --hdd /vz/MyVM.pvm/MyVM-0.hdd --
resize_partition
Operation progress 100 %
```

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

- You cannot reduce the capacity of a virtual disk if the virtual machine using this disk is running.

- The virtual machine using the virtual disk you want to configure must not have any snapshots. Otherwise, you will be informed of this fact:

```
# prl_disk_tool resize --size 68000 --hdd /vz/MyVM.pvm/MyVM-0.hdd/
This disk has one or more snapshots and cannot be resized.
You need to delete snapshots using the pctl tool before resizing the disk.
```

In this case, you should delete all existing snapshots and run the command again. To learn how to delete virtual machine's snapshots, refer to **Deleting a Snapshot** (p. 74).

- The capacity of an expanding virtual disk shown from inside the virtual machine and the size the virtual disk occupies on the server's physical disk may differ.

Compacting the Virtual Disk

In Parallels Server Bare Metal, you can decrease the space your virtual machines occupy on the Parallels server's disk drive by compacting their virtual disks. Compacting virtual disks allows you to save your server's disk space and host more virtual machines and Containers on the server.

Note: Plain disk cannot be compacted.

To compact a virtual disk, you can use the `prl_disk_tool  compact` command. For example, to compact the `MyVM-0.hdd` disk, you can run this command:

```
# prl_disk_tool compact --hdd /vz/MyVM.pvm/MyVM-0.hdd/
Operation progress 100 %
```

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

Managing Virtual Machine Devices

Parallels Server Bare Metal allows you to manage the following virtual machine devices:

- hard disk drives
- CD/DVD-ROM drives
- floppy disk drives
- network adapters
- serial and parallels ports
- sound cards
- USB controllers

The main operations you can perform on these devices are:

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

Adding a New Device

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

Option Name	Description
hdd	Adds a new hard disk drive to the virtual machine. You can either connect an existing image to the virtual machine or create a new one.
cdrom	Adds a new CD/DVD-ROM drive to the virtual machine.
net	Adds a new network adapter to the virtual machine.
fdd	Adds a new floppy disk drive to the virtual machine.
serial	Adds a new serial port to the virtual machine.
parallel	Adds a new parallel port to the virtual machine.
sound	Adds a new sound device to the virtual machine.
usb	Adds a new USB controller to the virtual machine.

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

```
# pctl set MyVM --device-add hdd
Creating hdd1 (+) scsi:0 image='/var/parallels/MyVM.pvm/harddisk1.hdd
Create the expanding disk, 32768Mb...
The VM has been successfully configured.
```

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

- name: `hdd1`
- disk type: SCSI
- image file name and location: `/var/parallels/MyVM.pvm/harddisk1.hdd`
- disk format: expanding
- disk capacity: 32768 MB

You can redefine some of these parameters by specifying specific options during the command execution. For example, to create an IDE virtual disk that will have the capacity of 64 GB, you can run this command:

```
# pctl set MyVM --device-add hdd --size 64000 --iface ide
Creating hdd1 (+) ide:1 image='/var/parallels/MyVM.pvm/harddisk1.hdd
Create the expanding disk, 64000Mb...
The VM has been successfully configured.
```

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

When managing devices, keep in mind the following:

- Detailed information on all options that can be passed to `pctl set` when creating a new virtual machine device is provided in the *Parallels Command Line Reference Guide*.
- You can connect up to 4 IDE devices and up to 15 SCSI devices (virtual disks or CD/DVD-ROM drives) to a virtual machine.

- If you want to use an existing image file as a virtual CD/DVD-ROM drive, keep in mind that Parallels Server Bare Metal supports `.iso`, `.cue`, `.ccd` and `.dmg` (non-compressed and non-encrypted) image files.
- A virtual machine can have only one floppy disk drive.
- A virtual machine can have up to 16 virtual network adapters.
- A virtual machine can have up to four serial ports.
- A virtual machine can have up to three parallel ports.
- Any virtual machine can have only one sound device.
- A virtual machine can have only one USB controller.

Initializing the Newly Added Disk

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

Initializing the New Virtual Hard Disk in Windows

To Initialize the new virtual hard disk in a Windows guest OS, you will need the Disk Management utility available through:

- In Windows Vista: **Start** > **Control Panel** > **System and Maintenance** > **Administrative Tools** > **Create and Format Hard Disk Partitions** > **Disk Management**.
- In Windows XP: **Start** > **Control Panel** > **Administrative Tools** > **Computer Management** > **Storage** > **Disk Management**.

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

1 In the introduction window, click **Next**.

2 In the **Select Disks to Initialize** window, select the newly added disk and click **Next**.

3 In the **Select Disks to Convert** window, select the newly added disk and click **Finish**.

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

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

Initializing the New Virtual Hard Disk in Linux

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

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

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

1 Launch Terminal.

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

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

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

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

```
fdisk /dev/hdc
```

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

4 To get extensive information about the disk, enter:

```
p
```

5 To create a new partition, enter:

```
n
```

6 To create the primary partition, enter:

```
p
```

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

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

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

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

```
w
```

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

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

> **Note:** `<FileSystem>` stands for the file system you want to use on this disk. It is recommended to use `ext3` or `ext2`.

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

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

```
mkdir /mnt/hdc1
```

> **Note:** You can specify a different mount point.

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

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

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

Configuring Virtual Devices

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

1   Finding out the name of the device you want to configure.

2   Running the `pctl set` command to configure the necessary device properties.

Finding Out the Device Name

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

```
# pctl list --info MyVM
...
Hardware:
  cpu 2 VT-x accl=high mode=32
  memory 256Mb
  video 46Mb
  fdd0 (+) real='/dev/fd0' state=disconnected
  hdd0 (+) ide:0 image='/var/parallels/MyVM.pvm/harddisk.hdd' 27000Mb
  hdd1 (+) scsi:0 image='/var/parallels/MyVM.pvm/harddisk1.hdd' 32768Mb
  cdrom0 (+) ide:1 real='Default CD/DVD-ROM'
  parallel0 (+) real='/dev/lp0'
  usb (+)
  net0 (+) type=bridged iface='eth1' mac=001C4201CED0
...
```

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

Configuring a Virtual Device

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

```
# pctl set MyVM --device-set hdd1 --iface ide
The VM has been successfully configured.
```

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

```
# pctl list --info MyVM
...
  hdd0 (+) ide:0 image='/var/parallels/MyVM.pvm/harddisk.hdd' 27000Mb
  hdd1 (+) ide:2 image='/var/parallels/MyVM.pvm/harddisk1.hdd' 32768Mb
...
```

Deleting a Device

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

Option Name	Description
hdd	Deletes the specified hard disk drive from the virtual machine.
cdrom	Deletes the specified CD/DVD-ROM drive from the virtual machine.
net	Deletes the specified network adapter from the virtual machine.
fdd	Deletes the floppy disk drive from the virtual machine.
serial	Deletes the specified serial port from the virtual machine.
parallel	Deletes the specified parallel port from the virtual machine.
sound	Deletes the sound device from the virtual machine.
usb	Deletes the USB controller from the virtual machine.

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

1   Finding out the name of the device to be deleted.

2   Deleting the device from the virtual machine.

Finding Out the Device Name

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

```
# pctl list --info MyVM
...
Hardware:
  cpu 2 VT-x accl=high mode=32
  memory 256Mb
  video 46Mb
  fdd0 (+) real='/dev/fd0' state=disconnected
  hdd0 (+) ide:0 image='/var/parallels/MyVM.pvm/harddisk.hdd' 27Mb
  hdd1 (+) scsi:0 image='/var/parallels/MyVM.pvm/harddisk1.hdd' 32768Mb
  cdrom0 (+) ide:1 real='Default CD/DVD-ROM'
  parallel0 (+) real='/dev/lp0'
  usb (+)
  net0 (+) type=bridged iface='eth1' mac=001C4201CED0
...
```

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

Deleting a Virtual Device

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

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

When deleting virtual machine devices, keep in mind the following:

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

- Detailed information on all options that can be used with pctl set when deleting a device is given in the *Parallels Command Line Reference Guide*.

Making Screenshots

In Parallels Server Bare Metal, you can use the pctl capture command to capture an image (or screenshot) of your virtual machine screen. You can take screenshots of running virtual machines only. The session below demonstrates how to take a screenshot of the My_VM virtual machine screen and save it to the /usr/screenshots/image1.png file:

1   Make sure that the virtual machine is running:

```
# pctl list
STATUS    IP_ADDR        NAME
running   10.10.10.101   101
running    10.10.10.201  My_VM
```

2   Take the virtual machine screenshot:

```
# pctl capture My_VM --file /usr/screenshots/image1.png
Capture the VM screen...
The VM screen has been saved to /usr/screenshots/image1.png
```

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

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

C H A P T E R 4

Managing Resources

The main goal of resource control in Parallels Server Bare Metal is to provide Service Level Management or Quality of Service for virtual machines and Containers. Correctly configured resource control settings prevent serious impacts resulting from the resource over-usage (accidental or malicious) of any virtual machine and Container on the other virtual machines and Containers. Using resource control parameters for resources management also allows you to enforce fairness of resource usage among virtual machines and Containers and better service quality for preferred virtual machines and Containers, if necessary.

In This Chapter

What are Resource Control Parameters?

The system administrator can control the resources available to a virtual machine and Container through a set of resource management parameters. All these parameters can be set and configured as follows:

- For Containers, by manually editing the global (`/etc/vz/vz.conf`) and Container (`/etc/vz/conf/`*CT_ID*) configuration files or using the Parallels Server Bare Metal command-line utilities.

- For virtual machines, by manually editing virtual machine configuration files (`/etc/vz/`*VM_Name*`.pvm/config.pvs`) or using the Parallels Server Bare Metal command-line utilities.

The following sections describe in detail how to configure resource parameters for both virtual machines and Containers.

Managing Resources for Containers

All resource management parameters for Containers can be divided into the disk, network, CPU, and system groups. The table below summarizes these groups:

Group	Description	Parameter names	Explained in
Disk	This group of parameters determines disk quota in Parallels Server Bare Metal. The disk quota is implemented on two levels: the per-Container 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, IOPRIO	Managing Disk Quotas
Network	This group of parameters determines the management of network bandwidth available to different Containers (network shaping). You can turn on/off network shaping and configure the settings for different Containers.	TRAFFIC_SHAPING, BANDWIDTH, TOTALRATE, RATE, RATEBOUND	Managing Network Accounting and Bandwidth
CPU	This group of parameters defines the CPU time different Containers are guaranteed to receive.	VE0CPUUNITS, CPUUNITS, CPUS, BURST_CPULIMIT, BURST_CPU_AVERAGE_USAGE	Managing Container CPU Resources
System	This group of parameters allows you to easily and effectively configure and control all memory-related parameters inside Containers.	slmmemorylimit	Managing System Parameters

Managing Container CPU Resources

The current section explains the CPU resource parameters that you can configure and monitor for each Container.

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

Parameter	Description	File
ve0cpuunits	This is a positive integer number that determines the minimal guaranteed share of the CPU time Container 0 (the server itself) will receive at its startup. It is recommended to set the value of this parameter to be 5-10% of the power of the server. After the server is up and running, you can redefine the amount of the CPU time allocated to the server by using the --cpuunits parameter with the pctl set command.	G
cpuunits	This is a positive integer number that determines the minimal guaranteed share of the CPU time the corresponding Container will receive.	V
	Note: In the current version of Parallels Server Bare Metal, you can also use this parameter to define the CPU time share for the server.	
cpulimit	This is a positive number indicating the CPU time, in percent, the corresponding Container is not allowed to exceed.	V
cpus	The number of CPUs to be used to handle the processes running inside the corresponding Container.	V

Managing CPU Share

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

Note: The CPU time shares and limits are calculated on the basis of a one-second period. Thus, for example, if a Container is not allowed to receive more than 50% of the CPU time, it will be able to receive no more than half a second each second.

To get a view of the optimal share to be assigned to a Container, check the current server CPU utilization:

```
# vzcpucheck
Current CPU utilization: 11142
Power of the node: 125504
```

The output of this command displays the total number of the so-called CPU units consumed by all running Containers and server processes. This number is calculated by Parallels Server Bare Metal with the help of a special algorithm. The above example illustrates the situation when the server is underused. In other words, the running Containers receive more CPU time than was guaranteed to them.

In the following example, Container 102 is guaranteed to receive about 4% of the CPU time even if the server is fully used, or in other words, if the current CPU utilization equals the power of the server. Besides, Container 102 will not receive more than 25% of the CPU time even if the CPU is not fully loaded:

```
# pctl set 102 --cpuunits 5000 --cpulimit 25 --save
Saved parameters for Container 102
# pctl start 102
Starting Container ...
Container is mounted
Adding IP address(es): 192.168.1.102
Container start in progress...
# vzcpucheck
Current CPU utilization: 15154
Power of the Node: 125504
```

Container 102 will receive from 4% to 25% of the server CPU time unless the server is overcommitted, i.e. the running virtual machines and Containers have been promised more CPU units than the power of the server. In this case the Container might get less than 4 percent.

Note: To set the `--cpuunits` parameter for the server, you should indicate 0 as the Container ID (e.g. `pctl set 0 --cpuunits 5000 --save`).

Configuring Number of CPUs Inside Container

If your server has more than one physical processor installed, you can control the number of CPUs which will be used to handle the processes running inside separate Containers. By default, a Container is allowed to consume the CPU time of all processors on the server, i.e. any process inside any Container can be executed on any processor on the server. However, you can modify the number of physical CPUs which will be simultaneously available to a Container using the --cpus option of the `pctl set` command. For example, if your server has 4 physical processors installed, i.e. any Container on the server can make use of these 4 processors, you can set the processes inside Container 101 to be run on 2 CPUs only by issuing the following command:

```
# pctl set 101 --cpus 2 --save
```

Note: The number of CPUs to be set for a Container must not exceed the number of physical CPUs installed on the server. In this case the 'physical CPUs' notation designates the number of CPUs the Parallels Server Bare Metal kernel is aware of (you can view this CPU number using the /proc/cpuinfo command).

You can check if the number of CPUs has been successfully changed by running the cat /proc/cpuinfo command inside your Container. Assuming that you have set two physical processors to handle the processes inside Container 101, your command output may look as follows:

```
# pctl exec 101 cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      : Intel(R) Xeon(TM) CPU 2.80GHz
stepping        : 1
cpu MHz         : 2793.581
cache size      : 1024 KB
...

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      : Intel(R) Xeon(TM) CPU 2.80GHz
stepping        : 1
cpu MHz         : 2793.581
cache size      : 1024 KB
...
```

The output shows that Container 101 is currently bound to only two processors on the server instead of 4 available for the other Containers on this server. It means that, from this point on, the processes of Container 101 will be simultaneously executed on no more than 2 physical CPUs while the other Containers on the server will continue consuming the CPU time of all 4 server processors, if needed. Also notice that the physical CPUs proper of Container 101 might not remain the same during the Container operation; they might change for load balancing reasons, the only thing that cannot be changed is their maximal number.

Managing Disk Quotas

This section explains the basics of disk quotas, defines disk quota parameters, and describes how to perform the following disk quota related operations:

- turning on and off per-Container (first-level) disk quotas
- setting up first-level disk quota parameters for a Container
- turning on and off per-user and per-group (second-level) disk quotas inside a Container
- setting up second-level quotas for a user or for a group
- checking disk quota statistics
- cleaning up Containers

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 Container can use. These quotas are known as per-Container quotas or first-level quotas in Parallels Server Bare Metal. In addition, the Parallels Server Bare Metal software enables the Container administrator to limit disk space and the number of inodes that individual users and groups in that Container can use. These quotas are called per-user and per-group quotas or second-level quotas.

By default, first-level quotas on your server are enabled (which is defined in the `/etc/vz/vz.conf` configuration file), whereas second-level quotas must be turned on for each Container separately (in the corresponding Container configuration files). It is impossible to turn on second-level disk quotas for a Container if first-level disk quotas are off for that Container.

Parallels Server Bare Metal keeps quota usage statistics and limits in `/var/vzquota/quota.<CT_ID>` - 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 Container usage. This means that when the disk space or inodes usage changes during the Container operation, these statistics are not automatically synchronized with the quota file, the file just gets the "dirty" flag. They are synchronized only when the Container is stopped or when the server is shut down. After synchronization, the "dirty" flag is removed. If the server 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 Container startup. This operation may noticeably increase the server startup time. Thus, it is highly recommended to shut down the server 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 global configuration file (G), in the Container configuration files (V), or it is defined in the global configuration file but can be overridden in a separate Container configuration file (GV).

Parameter	Description	File
disk_quota	Indicates whether first-level quotas are on or off for all Containers or for a separate Container.	GV
diskspace	Total size of disk space the Container may consume, in 1-Kb blocks.	V
diskinodes	Total number of disk inodes (files, directories, and symbolic links) the Container can allocate.	V
quotatime	The grace period for the disk quota overusage defined in seconds. The Container is allowed to temporarily exceed its quota soft limits for no more than the QUOTATIME period.	V
quotaugidlimit	This parameter defines the maximum aggregate number of user IDs and group IDs for which disk quota inside the given Container will be accounted. If set to 0, the UID and GID quota will be disabled.	V
ioprio	The Container priority for disk I/O operations. The greater the priority, the more time the Container has for writing to and reading from the disk.	V

Turning On and Off Per-Container Disk Quotas

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

This parameter can be specified in the Container configuration file (/etc/vz/conf/*<CT_ID>*.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 Containers 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 Container which does not need quotas.

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

```
[checking that quota is on]
# grep DISK_QUOTA /etc/vz/vz.conf
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 Container configuration file to add DISK_QUOTA=no]
# vi /etc/vz/conf/101.conf

[checking that quota is off for Container 101]
# grep DISK_QUOTA /etc/vz/conf/101.conf
DISK_QUOTA=no

# pctl start 101
Starting Container ...
Container is mounted
Adding IP address(es): 10.0.16.101
Hostname for Container set: ve101
Container start in progress...
# pctl exec 101 df
Filesystem            1k-blocks      Used Available Use% Mounted on
vzfs                   8282373    747060   7023242  10% /
```

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

Setting Up Per-Container Disk Quota Parameters

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

DISKSPACE	The total size of disk space that can be consumed by the Container, in 1-Kb blocks. When the space used by the Container hits the soft limit, the Container can allocate additional disk space up to the hard limit during the grace period specified by the QUOTATIME parameter.
DISKINODES	The total number of disk inodes (files, directories, and symbolic links) the Container can allocate. When the number of inodes used by the Container hits the soft limit, the Container 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, in seconds. The Container 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 Container configuration files and in the command line.

The following session sets the disk space available to Container 101 to approximately 1 GB and allows the Container to allocate up to 90,000 inodes. The grace period for the quotas is set to 10 minutes:

```
# pctl set 101 --diskspace 1000000:1100000 --save
Saved parameters for Container 101
# pctl set 101 --diskinodes 90000:91000 --save
Saved parameters for Container 101
# pctl set 101 --quotatime 600 --save
Saved parameters for Container 101
# pctl exec 101 df
Filesystem            1k-blocks       Used Available Use% Mounted on
vzfs                    1000000     747066    252934  75% /
# pctl exec 101 stat -f /
  File: "/"
    ID: 0        0        Namelen: 255      Type: UNKNOWN (0x565a4653)
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 Container. The changes will take effect immediately. If you do not want your changes to persist till the next Container startup, do not use the --save switch.

Turning On and Off Second-Level Quotas for a Container

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

2   The value that you assign to this parameter will be the limit for the number of file owners and groups of this Container, including Linux system users. Notice that you will theoretically be able to create extra users of this Container, but if the number of file owners inside the Container has already reached the limit, these users will not be able to own files.

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

When managing the QUOTAUGIDLIMIT parameter, keep in mind the following:

- If you delete a registered user but some files with their ID continue residing inside your Container, the current number of ugids (user and group identities) inside the Container will not decrease.

- If you copy an archive containing files with user and group IDs not registered inside your Container, the number of ugids inside the Container will increase by the number of these new IDs.

The session below turns on second-level quotas for Container 101:

```
# pctl set 101 --quotaugidlimit 100 --save
Unable to apply new quota values: ugid quota not initialized
Saved parameters for Container 101
# pctl restart 101
Stopping Container ...
Container was stopped
Container is unmounted
Starting Container ...
Container is mounted
Adding IP address(es): 192.168.1.101
Hostname for Container set: ct101
Container start in progress...
```

Setting Up Second-Level Disk Quota Parameters

Parallels Server Bare Metal provides the standard Linux `quota` package for working inside Containers:

```
# pctl exec 101 rpm -q quota
quota-4.03-1.1.parallels
```

This command shows that the `quota` package installed in the Container is built and shipped by Parallels. Use the utilities from this package (as is prescribed in your Linux manual) to set second-level quotas for the given Container. For example:

```
# ssh ct101
root@ct101's password:
Last login: Sat Jul 5 00:37:07 2009 from 10.100.40.18
[root@ct101 root]# edquota root
Disk quotas for user root (uid 0):
  Filesystem    blocks       soft       hard     inodes      soft       hard
   /dev/vzfs    38216       50000      60000     45454      70000      70000
[root@ct101 root]# repquota -a
*** Report for user quotas on device /dev/vzfs
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@ct101 root]# dd if=/dev/zero of=test
dd: writing to `test': Disk quota exceeded
23473+0 records in
23472+0 records out
[root@ct101 root]# repquota -a
*** Report for user quotas on device /dev/vzfs
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, consult the system administration guide shipped with your Linux distribution or online manual pages included in the package.

Checking Quota Status

As the server administrator, you can check the quota status for any Container with the `vzquota stat` and `vzquota show` commands. The first command reports the status from the kernel and shall be used for running Containers. The second command reports the status from the quota file (located at `/var/vzquota/quota.<CT_ID>`) and shall be used for stopped Containers. Both commands have the same output format.

The session below shows a partial output of Container 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 Container. 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.

Configuring Container Disk I/O Priority Level

Parallels Server Bare Metal provides you with the capability of configuring the Container disk I/O (input/output) priority level. The higher the Container I/O priority level, the more time the Container will get for its disk I/O activities as compared to the other Containers on the server. By default, any Container on the server has the I/O priority level set to 4. However, you can change the current Container I/O priority level in the range from 0 to 7 using the `--ioprio` option of the `pctl set` command. For example, you can issue the following command to set the I/O priority of Container 101 to 6:

```
# pctl set 101 --ioprio 6 --save
Saved parameters for Container 101
```

To check the I/O priority level currently applied to Container 101, you can execute the following command:

```
# grep IOPRIO /etc/vz/conf/101.conf
IOPRIO="6"
```

The command output shows that the current I/O priority level is set to 6.

Cleaning Up Containers

The first-level quota assigned to this or that Container essentially shows how much space may be occupied by the Container *private* files, i.e. not by the OS or common applications files. The real OS and application files reside in the `/vz/template` directory on the server and practically do not add up to the Container quota (except for the symlinks to them located inside the Container and occupying insignificant space).

However, there are situations when one and the same application or application update is installed not as a template, but separately inside each and every Container. A good example of this is the CPanel application with its robust auto-update features. If a certain version of CPanel is installed in a number of Containers, and then an update is released, CPanel automatically updates itself in all these Containers, thus creating a vast amount of identical files (not symlinks already) throughout the Containers. These files tell dramatically on the Container quotas, which may be avoided by putting all the identical files to the server template area and creating symlinks instead of real files inside the affected Containers.

The problem like the one described above can be solved in two ways:

1 A special subarea is created inside the server template area - `/vz/template/vc` - for housing the files identical among multiple Containers with the help of the `vzcache` utility.

2 If the application or application update installed directly into one or more Containers has a corresponding application template or template update installed on the server, the real files inside the Containers are replaced with symlinks to the template files on the server with the help of the `vzpkg link` utility. This utility is used to create symlinks to application EZ templates.

Moving Container Files to the Cache Area

We will illustrate the effect produced by vzcache by copying one and the same huge dummy file into two Containers. First, let us learn the disk space occupied by the whole /vz partition and by the two Containers - Container 101 and Container 102:

```
# df /vz
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda3            13756796    1348292  11622123  11% /vz
# pctl exec 101 df
Filesystem           1K-blocks      Used Available Use% Mounted on
vzfs                  1048576      22311   1026265   3% /
# pctl exec 102 df
Filesystem           1K-blocks      Used Available Use% Mounted on
vzfs                  1048576      22311   1026265   3% /
```

After that, we copy the dummy file, which is around 600 MB in size, to the root of these Containers:

```
# cp foo /vz/root/101
# cp foo /vz/root/102
```

Now check the disk space once again:

```
# df /vz
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda3            13756796    2569060  10401355  20% /vz
# pctl exec 101 df
Filesystem           1K-blocks      Used Available Use% Mounted on
vzfs                  1048576     632430    416146  61% /
# pctl exec 102 df
Filesystem           1K-blocks      Used Available Use% Mounted on
vzfs                  1048576     632430    416146  61% /
```

We see that around 600 MB has been added to the space occupied by each Container and, consequently, around 1.2 GB has been added to the space used on the /vz partition. Now it's time to resort to vzcache to get rid of identical files inside the Containers:

```
# vzcache -v 101 102
Processing VZFSv2 Container 101
VZFSv2 Container 101          78 regular files
Processing VZFSv2 Container 102
VZFSv2 Container 102          78 regular files
```

During the command execution, vzcache does the following:

- Looks for identical files inside Container 101 and Container 102.

- Creates the *CT_UUID* subdirectory (where *CT_UUID* denotes the Container unique identifier and can be determined by viewing the UUID parameters in the Container configuration file) within the server template area (/vz/template/vc by default) for each Container.

- Moves the identical files to the created subdirectories in the server template area.

Let us now take the final look at the disk space usage:

```
# df /vz
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda3            13756796    1953053  11017362  16% /vz
# pctl exec 101 df
Filesystem           1K-blocks      Used Available Use% Mounted on
vzfs                  1048576      15105   1033471   2% /
# pctl exec 102 df
Filesystem           1K-blocks      Used Available Use% Mounted on
vzfs                  1048576      15138   1033438   2% /
```

As you can see, both the server and the Containers have each gained more than 600 MB of disk space. In real life, the disk space is gained by caching not one huge file in two Containers but a number of identical files across many Containers.

The operation of the vzcache utility may be customized to a certain extent by using vzcache command line switches (see the *Parallels Command Line Reference Guide* for details).

Associating Container Files With Application Templates

It may often happen that a security update should immediately be applied to a package installed as a template on the server and added to a number of Containers hosted there. However, it takes certain time to prepare a template update, so the server and/or Container administrators are not inclined to wait for it and they install the original security update directly inside the Containers. As to the template update, it becomes available a few days afterwards. In other cases, a Container administrator might not know that there is a certain template installed on the server, so they install the corresponding application directly inside their Container.

To eliminate cluttering up the Container disk space with application files that are present as part of an application template on the server, the vzpkg link utility is used. When executed, this utility links your Container to the application EZ templates installed on the server. Assuming that you manually installed the openssl application inside Container 101 running Fedora 8, you can use the following command to replace the openssl files inside this Container with symlinks to these files in the /vz/template/fedora-core/8/x86/config/app/openssl directory on the server:

```
# vzpkg link 101
```

Detaching Containers From Caches

Whereas the vzcache utility helps effectively gain disk space both on the Parallels server and within Containers, there may be situations when it is necessary to detach a Container from its cache and copy the cached files back to the Container private area. A typical example of this is migrating a Container to another Parallels server. The migration is not possible if there are links in the Container private area pointing to the /vz/template/vzcaches directory on the server.

To copy the cached files back to the Container private area, the vzuncache utility is used:

```
# vzuncache 101 -a
[Optimization messages skipped...]
Container 101                            53 magic symlinks to convert

Container 101 will be detached from the following caches:
Cache name                               Size
dhcp0-84.sw.ru-2005030316237             607972K
```

Now Container 101 can safely be migrated to another Parallels server. Note that unlike vzcache, the vzuncache utility shall be called for only one Container at a time. The -a switch tells the utility to detach the Container from all the cache directories specified in its configuration file as the value of the VZCACHE parameter.

Managing Network Accounting and Bandwidth

This section explains how to perform the following tasks:

- setting up network classes
- viewing network traffic statistics
- turning on and off network bandwidth management
- setting up the bandwidth limit for a Container

Note: In the current version of Parallels Server Bare Metal, you can manage network accounting and bandwidth for Containers only.

Network Traffic Parameters

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

Parameter	Description	File
traffic_shaping	If set to yes, traffic limitations for outgoing traffic are set for Containers. The default is no.	G
bandwidth	This parameter lists all the network adapters installed on the server and their bandwidth.	G
totalrate	This parameter defines the bandwidth to be allocated for each and every network class. It is active if traffic shaping is turned on.	G
rate	If traffic shaping is turned on, this parameter specifies the bandwidth guarantee for any Container.	GV
ratebound	If this parameter is set to yes, the bandwidth guarantee (the global rate parameter) is also the limit for the Container, and the Container cannot borrow the bandwidth from the TOTALRATE bandwidth pool.	V

Configuring Network Classes

Parallels Server Bare Metal allows you to track the inbound and outbound network traffic as well as to shape the outgoing traffic for a Container. To provide the ability to distinguish between domestic and international traffic, a concept of network classes is introduced. It is important to fully understand this notion, because network classes IDs are used in the values of some network traffic parameters. A network class is a range of IP addresses for which Parallels Server Bare Metal counts and shapes the traffic.

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

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

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

Classes 0 and 1 have special meanings. Class 0 defines the IP address range for which no accounting is performed. Usually, it corresponds to the server subnet (the server itself and its Containers). Setting up Class 0 is not required; however, its correct setup improves performance.

Class 1 is defined by Parallels Server Bare Metal to match any IP address. It must be always present in the network classes definition file. Therefore, it is suggested not to change the default line

```
1 0.0.0.0/0
```

in the `networks_classes` file. Other Classes should be defined after Class 1. They represent exceptions from the "matching-everything" rule of Class 1. The example below illustrates a possible configuration of the network classes definition file:

```
# server Containers networks
0 192.168.0.0/16

# any IP address (all traffic)
1 0.0.0.0/0

# class 2 – addresses for the "foreign" traffic
2 10.0.0.0/8
2 11.0.0.0/8

# inside "foreign" network there
# is a hole belonging to "local" traffic
1 10.10.16.0/24
```

In this example, the IP addresses in the range of 192.168.0.0 to 192.168.255.255 are treated as Class 0 addresses and no accounting is done for the traffic from Containers destined to these addresses.

Class 2 matches addresses in two ranges: from 10.0.0.0 to 10.255.255.255 and from 11.0.0.0 to 11.255.255.255 with the exception of addresses in the sub-range of 10.10.16.0 to 10.10.16.255, which are treated as Class 1. All other IP addresses belong to Class 1. As far as the Class 2 addresses in this example are used for foreign routing, the Class 1 addresses are used for local (domestic) routing, by the exclusion method.

Viewing Network Traffic Statistics

Parallels Server Bare Metal allows you to view the current network traffic statistics with the help of the `vznetstat` command. The session below shows the traffic statistics for Container 101:

```
# vznetstat -v 101
CTID  Net.Class  Input(bytes) Input(pkts)  Output(bytes) Output(pkts)
101       1         2202448        19527       9081832        19584
101       2               0            0             0            0
```

In this case, around 2 MB of data were uploaded to the Container and about 9 MB were downloaded from it. All the traffic matches the definition of Class 1 and no data was exchanged with any hosts from Class 2 networks.

Without specifying Container ID with the `-v` parameter, the command will display the statistics for all running Containers.

Turning On and Off Network Bandwidth Management

Traffic shaping also known as network bandwidth management allows you to control what network bandwidth a Container receives for outgoing traffic. Traffic shaping is off by default in Parallels Server Bare Metal and is controlled by the `TRAFFIC_SHAPING` variable in the `/etc/vz/vz.conf` global configuration file.

Note: Container incoming traffic cannot be controlled in Parallels Server Bare Metal.

To turn traffic shaping on, you need to complete the following steps:

- Set the value of `TRAFFIC_SHAPING` to `yes` in the global configuration file.
- Correctly set up the `BANDWIDTH` and `TOTALRATE` parameters values.
- Start traffic shaping with the `/etc/init.d/vz shaperon` command.

The `BANDWIDTH` variable is used for specifying the network rate (in kilobits per second) of available network adapters. By default, it is set to `eth0:102400`, which corresponds to a 100Mb/s Fast Ethernet card. If your server has more network adapters installed, you need to update this variable to list all the adapters participating in shaping. For example, in case of two Fast Ethernet cards this variable shall be set to `eth0:102400 eth1:102400`.

The `TOTALRATE` variable specifies the size of the so-called bandwidth pool for each network class being shaped. The bandwidth from the pool can be borrowed by Containers when they need more bandwidth for communicating with hosts from the corresponding network class. It is used to limit the total available outgoing traffic Containers can consume; the next section explains it in more detail. The format of this variable is `<NIC>:<network_class>:<bandwidth_in_Kbits_per_second>` and defines the pool size per network class for a given network adapter. Multiple entries for different network classes and adapters shall be separated by spaces. The default value for `TOTALRATE` is `eth0:1:4096`, which corresponds to the pool size of 4Mb/s for Network Class 1 on the first Ethernet adapter.

In the `/etc/vz/vz.conf` configuration file, you can also define the `RATE` variable whose value amounts to the number of kilobits per second any Container is guaranteed to receive for outgoing traffic with a network class on an Ethernet device. The default value of this parameter is `eth0:1:8`, which means that any Container is guaranteed to receive the bandwidth of at least 8 Kb/s for sending data to Class 1 hosts on the first Ethernet device. This bandwidth is not the limit for a Container (unless the `RATEBOUND` parameter is set to `yes` in the Container configuration file) – the Container is able to take the needed bandwidth from the `TOTALRATE` bandwidth pool if it is not used by other Containers.

After setting up the above variables, start bandwidth management as follows:

```
# /etc/init.d/vz shaperon
Starting shaping: Ok
Set shaping on running Container :
vz WARNING: Can't get tc class for Container(101).
vz WARNING: Can't access file /var/run/vz_tc_classes. \
Creating new one.
vz WARNING: Can't get tc class for Container(1).
```

Now you have activated the network bandwidth limits. To turn traffic shaping off temporarily, use the `/etc/init.d/vz shaperoff` command. If you want to disable bandwidth management permanently, set the `TRAFFIC_SHAPING` variable to `no` in the `/etc/vz/vz.conf` configuration file.

Configuring Network Bandwidth Management for Container

The network bandwidth for outgoing traffic a Container receives is controlled by two variables in the Container configuration file (`/etc/vz/conf/<CT_ID>.conf`): `RATE` and `RATEBOUND`.

Note: Container incoming traffic cannot be controlled in the current version of Parallels Server Bare Metal.

The `RATE` variable has the same format as `TOTALRATE` - `<NIC>:<network_class>:<bandwidth>`. This variable specifies the guaranteed outgoing traffic rate that the corresponding Container receives. This rate can be specified differently for different network classes and network adapters; use space to separate several rate descriptions.

Bandwidth values are specified in Kb/s. It is recommended to increase this value in 8 Kb/s chunks and to set it no lower than 8 Kb/s.

The `RATEBOUND` variable specifies whether the network bandwidth available to the Container for outgoing traffic is limited by the bandwidth specified in the `RATE` variable. The possible values of the `RATEBOUND` variable are `yes` and `no`; the default is `no`. In this case the Container is allowed to take free bandwidth from the `TOTALRATE` pool.

The actual network bandwidth available to the Containers depends on the number of Containers and the total sum of the `RATE` values, and normally does not coincide with the bandwidth specified in their own `RATE` variables. If the `RATEBOUND` variable is set to `yes`, the Container bandwidth is limited by the value of the `RATE` variable.

If the Container configuration file does not specify any of these parameters, the values from the `/etc/vz/vz.conf` configuration file are taken. By default, Parallels Server Bare Metal does not set `RATEBOUND`, which corresponds to `no`, and `RATE` is set to `eth0:1:8`.

The network bandwidth management in Parallels Server Bare Metal works in the following way. The bandwidth pool for a given network class (configurable through the `TOTALRATE` variable in the global configuration file) is divided among the Containers transmitting data proportionally to their `RATE` settings. If the total value of the `RATE` variables of all Containers transmitting data does not exceed the `TOTALRATE` value, each Container gets the bandwidth equal or greater than its `RATE` value (unless this Container has the `RATEBOUND` variable set to `yes`). If the total value of the `RATE` variables of all Containers transmitting data exceeds the `TOTALRATE` value, each Container may get less than its `RATE` value.

The example below illustrates the scenario when there are two Containers, 101 and 102, which have `RATEBOUND` set to `no`, and Container 103 has `RATEBOUND` set to `yes`:

```
# grep ^RATE /etc/vz/conf/101.conf /etc/vz/conf/102.conf
RATE="eth0:1:8"
RATEBOUND="no"
RATE="eth0:1:8"
RATEBOUND="no"
# grep ^RATE /etc/vz/conf/103.conf
RATE="eth0:1:64"
RATEBOUND="yes"
```

With the default TOTALRATE of 4096 Kb/s, bandwidth pool will be distributed according to the following table:

Container 101	Container 102	Container 103	Bandwidth consumed by Containers
transmits	idle	idle	Container101: 4096 Kb/s
idle	idle	transmits	Container103: 64 Kb/s
transmits	transmits	idle	Container101: 2048 Kb/s Container102: 2048 Kb/s
transmits	idle	transmits	Container101: 4032 Kb/s Container103: 64 Kb/s
transmits	transmits	transmits	Container101: 2016 Kb/s Container102: 2016 Kb/s Container103: 64 Kb/s

After you have set up Container bandwidth settings, activate your changes as shown below:

```
# /etc/init.d/vz shaperrestart
Stopping shaping: Ok
Starting shaping: Ok
Set shaping on running Container: Ok
```

This command clears off all existing shaping settings and sets them again using the configuration files of running Containers.

Managing System Parameters

The given section provides information on how you can manage the system resource parameters, which a Container may allocate, using the Service Level Management (SLM) system. This system allows you to easily and effectively configure and control all memory-related parameters inside Containers.

Overview

Service Level Management (SLM) is a special system allowing you to configure and control the service levels provided to Container users. SLM can be used to manage the Container memory resources, i.e. to adjust the amount of memory that any Container on the server is allowed to consume. The SLM scheme has been developed to replace the UBC scheme of managing system resources parameters, thus simplifying the resources management inside Containers by uniting all memory-related parameters into a single `slmmemorylimit` parameter.

Note: Detailed information on all UBC parameters is provided in the *Administrator's Guide to Managing UBC Resources* guide available at http://www.parallels.com/products/virtuozzo/docs/.

SLM can be used to ensure that:

- The memory consumption by every Container on the server does not exceed its instant memory limit.
- The memory usage by every Container on the server does not exceed its average limit.
- The total memory consumption by all Containers does not exceed the amount of memory available on the server and prevents the total memory from reaching the point when the server performance begins to significantly degrade.
- The 'low memory' usage by all Containers on the server does not leave the safe range.

Computing Memory Usage in SLM

As the server administrator, you may often need to properly set the amount of memory this or that Container will be allowed to consume. Therefore, you should have a clear idea of the memory computation mechanism used in the SLM scheme. On the whole, the memory usage inside every particular Container for which the SLM functionality is enabled is calculated in the same way as it would be done on a standalone server. It means that the same set of applications running inside a Container will require approximately the same amount of RAM for their functioning as it would require on any other standalone server. Consequently, the amount of memory to be allocated to any Container largely depends on the number of applications you are going to deploy inside the Container and their memory requirements. For example, if you are going to use your Container as a web server only, there is no need to allocate much RAM to this Container (e.g. no more than 50 MB). At the same time, running such memory intensive applications as MySQL, Perl, PHP requires the memory limit be set to no less than 300 MB.

The situation above provides only the general description of memory usage inside Containers. In fact, the process of memory computation used in the SLM scheme is more complicated. It includes the calculation of the `oomguarpages`, `kmemsize`, `lockedpages`, and `socket buffer` parameters and the unification of these parameters into a single `slmmemorylimit` parameter. It also assumes a number of accounting rules to be taken into consideration while deciding on the amount of memory to be allocated to a Container. The main rules are given below:

- The memory allocated to a Container includes both memory itself and the swap space.

- The memory consumption inside a Container is calculated by taking into account the data sharing among applications. So, if two Containers share one and the same memory page, each Container is considered to consume half a page. As the number of Containers sharing the same memory pages grows, the memory consumption inside each of these Containers decreases. Thus, an application running inside a Container can consume much less memory than the identical application launched in the Host OS or on a standalone server. Especially much data can be shared when Containers use the same versions of applications and shared libraries (e.g. in the case of using the same versions of the `apache` Web server with the same set of modules and the same versions of system libraries). In such cases the difference in memory usage may reach tens of megabytes.

- The total amount of used memory and swap space in the system is computed on the basis of the memory consumption inside all Containers plus memory usage in the Host OS.

Controlling Memory Usage by Container

SLM has a number of means at its disposal allowing you to effectively control and configure the memory usage on the server and inside its Containers. These means include:

a Using the `free` command to check the memory limit set for a Container and the current memory consumption inside this Container. If the SLM functionality is disabled, running this command inside your Containers will display the total and used memory on the server.

b Restricting the rate of creating new processes and threads inside a Container.

c Denying memory allocation requests from a Container.

d Sending the `SISTERM` signal to applications intensively consuming the memory and requesting them to terminate all their operations, save the data, and exit.

e Killing a 'dangerous' application by sending the `SIGKILL` signal to it.

Various means of managing the Container memory consumption on the server makes SLM more application-friendly as compared to the management scheme by means of UBC parameters (the latter has only the methods described in items `c` and `e` at its disposal). This allows SLM to select the right means while deciding on the steps to be taken in respect of this or that application. Among other things, SLM takes into account the following characteristics:

- the severity of a memory limit excess
- the duration and frequency of excesses

SLM Modes

SLM is automatically enabled during the Parallels Server Bare Metal installation on the server, i.e. you do not have to perform any additional operations to start using this functionality on your server. After the installation, you can manage SLM in one of the following ways:

- Disable SLM on a global basis. In this case no Container on the server will be able to make use of this functionality. To disable SLM, complete the following tasks:

    - Specify `no` as the value of the `SLM` parameter in the `/etc/vz/vz.conf` global configuration file.

    - Reboot the server:

```
# shutdown -r now
```

- Control the SLM mode for a particular Container on the server. The current version of Parallels Server Bare Metal allows you to set one of the following SLM modes for your Container:

    - *limited mode*. In this mode, the SLM functionality for the corresponding Container is enabled and can be used to control the 'total' and 'low' memory consumption by all Containers on the server, which prevents the memory from being overused and guarantees the reliable performance of the server. At the same time, you can use various UBC parameters to manage particular resources of the Container. If the Container does not have any UBC parameters set, SLM also undertakes the control over the consumption of these resources by this Container. By default, any Container created is functioning in the *limited mode*. If your Container is working in another mode, you can return it to this mode by executing the `pctl set` command and passing the `--slmmode all` option to it.

    - *full mode*. In this mode, the SLM functionality for the corresponding Container is enabled and can be used to the full extent for managing the amount of memory which can be allocated to and consumed by the Container. Enabling the *full mode* automatically sets the values of all UBC parameters to unlimited. When functioning in this mode, SLM may significantly improve the resources allocation among individual Containers. For example, it allows you to avoid situations when the memory allocation for some application inside the Container fails although the system has a lot of free resources. The *full mode* can be set by using the `--slmmode slm` option with the `pctl set` command.

    - *compatibility mode*. In this mode, the SLM functionality for the corresponding Container is disabled and the system resources control management is performed by using UBC parameters only: `numproc`, `numtcpsock`, `numothersock`, `vmguarpages`, `kmemsize`, etc. Detailed information on all UBC parameters is provided the *Administrator's Guide to Managing UBC Resources* guide available at http://www.parallels.com/products/virtuozzo/docs/. The *compatibility mode* can be set by using the `--slmmode ubc` option with the `pctl set` command.

Note: You can also enable any of the aforementioned modes by editing the Container configuration file and setting the corresponding value (`all`, `slm`, or `ubc`, respectively) of the `SLM` parameter in this file.

Managing Container Memory Usage

The SLM mechanism allows you to manage the amount of memory a Container can consume by configuring a single parameter - `slmmemorylimit`. This significantly simplifies the process of memory management on the server and inside its Containers and represents the main SLM advantage over the old memory management mechanism (implemented on the basis of multiple UBC parameters). You can set or configure the Container memory usage limit by means of the `--slmmemorylimit` parameter of the `pctl set` command.

Let us assume that you wish to use SLM to manage the amount of memory which can be consumed by Container 101 and set its memory limit to 100 MB. This can be done by executing the following command:

```
# pctl set 101 --slmmemorylimit 102400000
Saved parameters for Container 101
```

By default, the memory limit to be allocated to your Container is set in bytes; however, you can change the default units of measurement by adding the following symbols after the value:

- `K`: specifying this symbol after the value allows you to set the Container memory limit in kilobytes (e.g. `1000K`).

- `P`: specifying this symbol after the value allows you to set the Container memory limit in pages (e.g. `200P`).

- `M`: specifying this symbol after the value allows you to set the Container memory limit in megabytes (e.g. `100M`).

- `G`: specifying this symbol after the value allows you to set the Container memory limit in gigabytes (e.g. `1G`).

After the memory limit has been successfully set for Container 101, you can view it by running the `free` command inside this Container:

```
# pctl exec 101 free
               total     used      free  shared  buffers  cached
Mem:          102400    46216     56184       0    10532   27748
-/+ buffers/cache:      17936     49748
Swap:         204800        0    204800
```

As can be seen from the example above, the specified memory limit is shown as the total memory available to Container 101.

Grouping Applications Inside Container

SLM provides a mechanism of classifying available applications (or processes representing instances of these running applications) inside a Container, uniting them into certain groups, and ensuring a sort of isolation among these groups. Such application grouping allows you to separately control each application group and, if the Container exceeds its memory limit and some application group inside this Container overuses the memory, to reduce the memory consumption only by the corresponding application group rather than to impose memory restrictions on the whole Container and all its applications. For example, this can help you keep the remote SSH connection to your Container in the case of the `apache` Web server misbehaviour or keep this Web service working if the 'dangerous' application is the `sendmail` service.

In the current version of Parallels Server Bare Metal, all applications (processes) inside a Container are by default included in one of the following groups:

- `'other'` (also referred to as group 0): this group contains all the processes not included in the `'daemons'`, `'httpd'`, and `'mysql'` groups. The termination of any process belonging to this group affects certain (usually uncritical) Container functionality only and does not lead to the entire Container DoS (denial of service).

- `'daemons'` (also referred to as group 1): this group includes `init`, `rc`, and all system daemons (e.g. `sshd`). The `'daemons'` group is the most important one and provides the basis for the Container functioning.

- `'httpd'` (also referred to as group 2): this group includes the `apache` Web server only. The processes in this group and the `'mysql'` one provide the main workload of any Container.

- `'mysql'` (also referred to as group 3): this group includes the MySQL database server only. The processes in this group and the `'httpd'` one provide the main workload of any Container.

By default, any new process inherits the group from its parent process. For example, all children of the `httpd` process are placed to the `'httpd'` group whereas all children of the `'mysql'` process are included in the `'mysql'` group. However, the group of a process can be changed during its forking and/or execution on the basis of special SLM pattern rules. The default SLM pattern rules are specified in the `/etc/vzslm.d/default.conf` file on the server in the table having the following four columns:

- *first_column*: the name of the process to which the rule is to be applied.

- *second_column*: a bitwise set of values defining the scheme on the basis of which the process is to be moved to the corresponding group.

- *third_column*: the group the process belongs to before the rule is applied. The `-1` value, if specified, means any group.

- *fourth_column*: the group where the process will be moved after the rule is applied.

The *flags* field represents a number containing one or several of the following bitwise values:

Hexadecimal Notation	Binary Notation	Description
0x0001	|_0_|_0_|_0_|_0_|_0_|_0_|_0_|_1_|	This bit, if set to 1, indicates that the rule is to be applied to the

		process if it is a daemon.
0x0002	|_0_|_0_|_0_|_0_|_0_|_0_|_1_|_0_|	This bit, if set to 1, indicates that the rule is to be applied to the process if it is not a daemon.
0x0004	|_0_|_0_|_0_|_0_|_0_|_1_|_0_|_0_|	This bit, if set to 1, indicates that the rule is to be applied to the process during its forking (i.e. on the `fork()` call).
0x0008	|_0_|_0_|_0_|_0_|_1_|_0_|_0_|_0_|	This bit, if set to 1, indicates that the rule is to be applied to the process during its execution (i.e. on the `exec()` call).
0x0010	|_0_|_0_|_0_|_1_|_0_|_0_|_0_|_0_|	This bit, if set to 1, indicates that the name of the process is to be checked before applying the rule.

Let us take as an example the following rule from the `/etc/vzslm.d/default.conf` file

```
"httpd"  0000001c  -1  2
```

and examine what processes are affected by this rule and in what way. The flags in this rule (`0000001c` or `|_0_|_0_|_0_|_1_|_1_|_1_|_0_|_0_|` in the binary notation) involve checking the name of the process (the fifth bit from the right equals 1) and, if this name is `httpd`, moving the process to the 'httpd' group ($destination\_subgroup$ = 2) regardless of the group it originally belongs to ($source\_subgroup$ = -1) during the process forking and execution (the third and forth bits from the right equal 1).

The following table lists all the rules present in the `/etc/vzslm.d/default.conf` file:

Rule Name			Explanation
#1 "init"	00000018	-1 9	If the process has the name of init, move it to group 9 during the process execution irrespective of the group it originally belongs to. As there is no default group numbered 9, it will be created when this rule is first applied.
#2 "httpd"	0000001c	-1 2	If the process has the name of httpd, move it to group 2 during the process forking and execution irrespective of the group it originally belongs to.
#3 "httpsd"	0000001c	-1	If the process has the name of httpsd, move it to group 2 during the process forking and

	2			execution irrespective of the group it originally belongs to.
#4	"lighthttpd"	0000001c	-1	If the process has the name of lighthttpd, move it to group 2 during the process forking and execution irrespective of the group it originally belongs to.
	2			
#5	"mysqld"	0000001c	-1	If the process has the name of mysqld, move it to group 3 during the process forking and execution irrespective of the group it originally belongs to.
	3			
#6	"syslogd"	00000018	0	If the process has the name of syslogd and originally belongs to group 0, move it to group 8 during the process execution.
	8			
#7	"sshd"	00000018	0	If the process has the name of sshd and originally belongs to group 0, move it to group 8 during the process execution.
	8			
#8	"inetd"	00000018	0	If the process has the name of inetd and originally belongs to group 0, move it to group 8 during the process execution.
	8			
#9	"xinetd"	00000018	0	If the process has the name of xinetd and originally belongs to group 0, move it to group 8 during the process execution.
	8			
#10	"cron"	00000018	0	If the process has the name of cron and originally belongs to group 0, move it to group 8 during the process execution.
	8			
#11	"crond"	00000018	0	If the process has the name of crond and originally belongs to group 0, move it to group 8 during the process execution.
	8			
#12	""	00000004	9	If the process originally belongs to group 9, move it to group 0 during the process forking. As there is only one process belonging to group 9 - init, this rule will be applied to the init children only (see #1).
	0			
#13	""	00000004	8	If the process originally belongs to group 8, move it to group 1 during the process forking.
	1			
#14	""	00000004	1	If the process originally belongs to group 1, move it to group 0 during the process forking.
	0			

Note: As all processes (parents) in rules #6 - #11 belong to group 1, the instances these rules can be applied to can only be children (see rule #14).

During its life cycle, any process running inside the Container is checked against the available rules in the `/etc/vzslm.d/default.conf` file from top to bottom and the first matching rule is applied to it. So, if the following 2 rules are present in the `default.conf` file

```
"httpd"   0000001c   -1   2
"httpd"   00000016   -1   1
```

the first rule (`"httpd"   0000001c   -1   2`) will be applied to all `httpd` processes inside all Containers on the server.

You can create your own SLM pattern configuration files with your own rules and apply them to particular Containers on the server. For example, if you want Container 101 to start using a configuration file different from `/etc/vzslm.d/default.conf`, you can proceed as follows:

1 Create a new file with an arbitrary name and the `.conf` extension (e.g. by means of `vi`) and place it to the `/etc/vzslm.d` directory on the server.

2 Make Container 101 use the newly created configuration file. Assuming that the configuration file name is `light.conf`, you can do it by issuing the following command on the server:

```
# pctl set 101 --slmpattern light --save
Saved parameters for Container 101
```

> **Note:** If you want to make all Containers on the server use another SLM pattern configuration file, you should specify the name of this file without the `.conf` extension (e.g. `light`) as the value of the `SLMPATTERN` parameter in the `/etc/vz/vz.conf` configuration file.

Managing Container Resources Configuration

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

1 Using configuration sample files shipped with Parallels Server Bare Metal. These files are used when a new Container is being created (for details, see the **Creating and Configuring New Container** section). Currently, the following configuration sample files are provided:

- `basic` – to be used for creating standard Containers.

- `confixx` – to be used for creating Containers that are to use the Confixx control panel.

- `slm.plesk` - to be used for creating Containers with the Plesk control panel.

- `slm.256MB` - to be used for creating Containers with 256 MB of main memory.

- `slm.512Mb` - to be used for creating Containers with 512 MB of main memory.

- `slm.1024Mb` - to be used for creating Containers with 1024 MB of main memory.

- `slm.2048Mb` - to be used for creating Containers with 2048 MB of main memory.

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

Any sample configuration file may also be applied to a Container 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 Container:

```
# pctl set 101 --applyconfig basic --save
```

This command applies all the parameters from the `ve-basic.conf-sample` file to the given Container.

When you install Parallels Server Bare Metal on your server, the default Container samples having the `ve-<name>.conf-sample` names are put to the `/etc/vz/conf` directory. In this connection you should keep in mind the following when working with Container samples:

- When you create a Container using the `pctl create` command utility and base it on some Container sample, this sample is taken from the `/etc/vz/conf` directory.

- If you modify an existing Container sample or create a new sample using specific command line utilities (e.g. `vzsplit`, `vzcfgscale`), the changes are made to the corresponding file in the `/etc/vz/conf` directory or the resulting Container sample is put to this directory.

2 Using specific utilities for preparing configuration files in their entirety. The tasks these utilities perform are described in the following subsections of this section.

3 The direct creating and editing of the corresponding Container configuration file (`/etc/vz/conf/<CT_ID>.conf`). This can be performed with the help of any text editor. The instructions on how to edit Container configuration files directly are provided in the four preceding sections. In this case you have to edit all the configuration parameters separately, one by one.

Splitting server Into Equal Pieces

It is possible to create a Container configuration roughly representing a given fraction of the server. If you want to create such a configuration that up to 20 fully loaded Containers would be able to be simultaneously running on the given server, you can do it as follows:

```
# cd /etc/vz/conf
# vzsplit -n 20 -f mytest
Config /etc/vz/conf/ve-mytest.conf-sample was created
```

Notice that the configuration produced depends on the given server 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. For example:

```
# vzcfgvalidate ve-mytest.conf-sample
Recommendation: kmemsize.lim-kmemsize.bar should be > 253952 \
(currently, 126391)
Recommendation: dgramrcvbuf.bar should be > 132096 (currently, 93622)
```

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

```
# pctl create 101 --ostemplate redhat-el5-x86 --config mytest
Creating Container private area (redhat-el5-x86)
Container is mounted
Postcreate action done
Container is unmounted
Container private area created
Container registered successfully
# pctl set 101 --ipadd 192.168.1.101 --save
Saved parameters for Container 101
# pctl start 101
Starting Container ...
Container is mounted
...
# vzcalc 101
Resource      Current(%)  Promised(%)  Max(%)
Memory          0.53        1.90        6.44
```

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

Note: If you generate a Container configuration sample using the vzsplit command line utility, the resulting Container sample is put to the /etc/vz/conf directory. This sample can then be used by pctl create when creating a new Container on its basis.

Scaling Container Configuration

Any configuration or configuration sample file can prove insufficient for your needs. You might have an application which does not fit into existing configurations. The easiest way of producing a Container configuration is to scale an existing one.

Scaling produces a "heavier" or "lighter" configuration in comparison with an existing one. All the parameters of the existing configuration are multiplied by a given number. A heavier configuration is produced with a factor greater than 1, and a lighter one – with a factor between 0 and 1.

Note: If you create a new sample on the basis of an existing sample using the `vzcfgscale` command line utility, the resulting Container sample is put to the `/etc/vz/conf` directory. This sample can then be used by `pctl create` when creating a new Container on its basis.

The session below shows how to produce a configuration sample 50% heavier than the `basic` configuration shipped with Parallels Server Bare Metal:

```
# cd /etc/vz/conf
# vzcfgscale -a 1.5 -o ve-improved.conf-sample ve-basic.conf-sample
# vzcfgvalidate ve-improved.conf-sample
Recommendation: kmemsize.lim-kmemsize.bar should be > 245760 \
(currently, 221184)
Recommendation: dgramrcvbuf.bar should be > 132096 (currently, 98304)
Validation completed: success
```

Now `improved` can be used in the `pctl create` command for creating new Containers.

It is possible to use the same technique for scaling configurations of the existing Containers. Notice that the output file cannot be the same as the file being scaled. You have to save the scaling results into an intermediate file.

Validating Container Configuration

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

The typical validation scenario is shown below:

```
# vzcfgvalidate  /etc/vz/conf/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)
# pctl set 101 --kmemsize 2211840:2359296 --save
Saved parameters for Container 101
# vzcfgvalidate  /etc/vz/conf/101.conf
Recommendation: kmemsize.lim-kmemsize.bar should be > 163840 \
(currently, 147456)
Recommendation: dgramrcvbuf.bar should be > 132096 (currently, 65536)
Recommendation: othersockbuf.bar should be > 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 Container or server 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 Container 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 Container 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 Container can be safely run with this configuration.

Applying New Configuration Sample to Container

Parallels Server Bare Metal allows you to change the configuration sample file a Container is based on and, thus, to modify all the resources the Container may consume and/or allocate at once. For example, if Container 101 is currently based on the `basic` configuration sample and you are planning to run the Plesk application inside the Container, you may wish to apply the `slm.plesk` sample to it instead of `basic`, which will automatically adjust the necessary Container resource parameters for running the Plesk application inside Container 101. To do this, you can execute the following command on the server:

```
# pctl set 101 --applyconfig slm.plesk --save
Saved parameters for Container 101
```

This command reads the resource parameters from the `ve-slm.plesk.conf-sample` file located in the `/etc/vz/conf` directory and applies them one by one to Container 101.

When applying new configuration samples to Containers, keep in mind the following:

- All Container sample files are located in the `/etc/vz/conf` directory on the server and are named according to the following pattern: `ve-<name>.conf-sample`. You should specify only the `<name>` part of the corresponding sample name after the `--applyconfig` option (`slm.plesk` in the example above).

- The `--applyconfig` option applies all the parameters from the specified sample file to the given Container, except for the `OSTEMPLATE`, `TEMPLATES`, `VE_ROOT`, `VE_PRIVATE`, `HOSTNAME`, `IP_ADDRESS`, `TEMPLATE`, `NETIF` parameters (if they exist in the sample file).

- You may need to restart your Container depending on the fact whether the changes for the selected parameters can be set on the fly or not. If some parameters could not be configured on the fly, you will be presented with the corresponding message informing you of this fact.

Managing Virtual Machine Resources

Parallels Server Bare Metal allows you to manage the following resources of your virtual machines:

- main memory
- number of CPUs
- video memory

The procedure of managing these resources is described below in this section.

Configuring Main Memory

To configure the amount of memory that will be available to the virtual machine, use the --memsize option of the pctl set command. The following session shows how to change the amount of memory for the MyVM virtual machine from 512 MB to 756 MB and to check that the new value has been successfully set:

```
# pctl list -i MyVM | grep memory
  memory 512Mb
# pctl set MyVM --memsize 756
Set the memsize parameter to 756Mb
The VM has been successfully configured.
# pctl list -i MyVM | grep memory
  memory 756Mb
```

You can configure the memory size for both running and stopped virtual machines.

Configuring the Number of CPUs

If the Parallels server has more than one physical processor installed, you can control the number of CPUs which will be used to handle the processes running inside your virtual machines. By default, a virtual machine is allowed to consume the CPU time of one processor only. However, you can modify the number of physical CPUs which will be simultaneously available to a virtual machine using the --cpus option of the pctl set command. For example, if your server has 4 physical processors installed, you can set the processes inside the MyVM virtual machine to be run on 2 CPUs by issuing the following command:

```
# pctl set MyVM --cpus 2
Set cpus(2): 2
The VM has bee successfully configured.
```

Note: The maximum allowable number of virtual CPUs depends on the number of physical CPU cores available on the Parallels server. For example, if you have a Core 2 Duo physical processor, the maximum allowable number of virtual CPUs will be 2.

You can check if the number of CPUs has been successfully changed by running this command:

```
# pctl list -i MyVM | grep cpu
  cpu 2 VT-x acc1=high mode=32
```

Configuring Video Memory

To set the amount of video memory to be available to the virtual machine's video card, use the --videosize option of the `pctl set` command. Assuming that the current video memory size of the `MyVM` virtual machine is set to 32 MB, you can increase it to 64 MB by running the following command:

```
# pctl set MyVM --videosize 64
Set the --videosize parameter to 64Mb.
The VM has been successfully configured.
```

To check that the new value has been successfully set, use this command:

```
# pctl list -i MyVM | grep video
  video 64Mb
```

C H A P T E R   5

Managing Services and Processes

This chapter provides information on what services and processes are, the influence they have on the operation and performance of your system, and the tasks they perform in the system.

You will learn how to use the command line utilities in order to manage services and processes in Parallels Server Bare Metal. In particular, you will get to know how you can monitor active processes in your system, change the mode of the `xinetd`-dependent services, identify the Container ID where a process is running by the process ID, start, stop, or restart services and processes, and edit the service run levels.

Note: In the current version of Parallels Server Bare Metal, you cannot manage services and processes in virtual machines using Parallels Server Bare Metal utilities. However, you can log in to a particular virtual machine (e.g. via RDP to a Windows virtual machine and SSH to a Linux virtual machine) and manage its services and processes in the same way you would manage them on a standalone computer.

In This Chapter

What Are Services and Processes

Instances of any programs currently running in the system are referred to as processes. A process can be regarded as the virtual address space and the control information necessary for the execution of a program. A typical example of a process is the *vi* application running on your server or inside your Linux-based Containers. Along with common processes, there are a great number of processes that provide an interface for other processes to call. They are called services. In many cases, services act as the brains behind many crucial system processes. They typically spend most of their time waiting for an event to occur or for a period when they are scheduled to perform some task. Many services provide the possibility for other servers on the network to connect to the given one via various network protocols. For example, the `nfs` service provides the NFS server functionality allowing file sharing in TCP/IP networks.

You may also come across the term "daemon" that is widely used in connection with processes and services. This term refers to a software program used for performing a specific function on the server system and is usually used as a synonym for "service". It can be easily identified by "d" at the end of its name. For example, `httpd` (short for the HTTP daemon) represents a software program that runs in the background of your system and waits for incoming requests to a web server. The daemon answers the requests automatically and serves the hypertext and multimedia documents over the Internet using HTTP.

When working with services, you should keep in mind the following. During the lifetime of a service, it uses many system resources. It uses the CPUs in the system to run its instructions and the system's physical memory to hold itself and its data. It opens and uses files within the file systems and may directly or indirectly use certain physical devices in the system. Therefore, in order not to decrease your system performance, you should run only those services on the Parallels server that are really needed at the moment.

Besides, you should always remember that running services in the Host OS is much more dangerous than running them in virtual machines and Containers. In case violators get access to one of the virtual machines and Containers through any running service, they will be able to damage only the virtual machine and Container where this service is running, but not the other virtual machines and Containers on your server. The Parallels server itself will also remain unhurt. And if the service were running on the Parallels server, it would damage both the server and all virtual machines and Containers residing on it. Thus, you should make sure that you run only those services on the server that are really necessary for its proper functioning. Launch all additional services you need at the moment inside separate virtual machines and Containers. It can significantly improve your system safety.

Main Operations on Services and Processes

The ability to monitor and control processes and services in your system is essential because of the profound influence they have on the operation and performance of your whole system. The more you know about what each process or service is up to, the easier it will be to pinpoint and solve problems when they creep in.

The most common tasks associated with managing services running on the Parallels server or inside a virtual machine and Container are starting, stopping, enabling, and disabling a service. For example, you might need to start a service in order to use certain server-based applications, or you might need to stop or pause a service in order to perform testing or to troubleshoot a problem.

For `xinetd`-dependent services, you do not start and stop but enable and disable services. The services enabled in this way are started and stopped on the basis of the corresponding state of the `xinetd` daemon. Disabled services are not started whatever the `xinetd` state.

In Parallels Server Bare Metal, you can manage services on the Parallels server and inside Containers by means of special Linux command-line utilities. You can do it either locally or from any server connected on the network.

As for processes, such Parallels Server Bare Metal utilities as `vzps`, `vztop`, `vzpid` enable you to see what a process is doing and to control it. Sometimes, your system may experience problems such as slowness or instability, and using these utilities can help you improve your ability to track down the causes. It goes without saying that in Parallels Server Bare Metal you can perform all those operations on processes you can do in a normal system, for example, kill a process by sending a terminate signal to it.

Managing Processes and Services

In Parallels Server Bare Metal, services and processes can be managed using the following Parallels command line utilities:

- `vzps`
- `vzpid`
- `vztop`
- `vzsetxinetd`.

With their help, you can perform the following tasks:

- print the information about active processes on your Parallels server
- view the processes activity in real time
- change the mode of the services that can be either `xinetd`-dependent or standalone
- identify the Container ID where a process is running by the process ID

Note: In the current version of Parallels Server Bare Metal, you cannot use Parallels Server Bare Metal utilities for managing services and processes in virtual machines. However, you can log in to a particular virtual machine (e.g. via RDP to a Windows virtual machine and SSH to a Linux virtual machine) and manage its services and processes in the same way you would manage them on a standalone computer.

Viewing Active Processes and Services

The `vzps` utility provides certain additional functionality related to monitoring separate Containers running on the Parallels server. For example, you can use the `-E` switch with the `vzps` utility to:

- display the Container IDs where the processes are running
- view the processes running inside a particular Container

`vzps` prints the information about active processes on your Parallels server. When run without any options, `vzps` lists only those processes that are running on the current terminal. Below is an example output of the `vzps` run:

```
$ vzps
  PID TTY          TIME CMD
 4684 pts/1    00:00:00 bash
27107 pts/1    00:00:00 vzps
```

Currently, the only processes assigned to the user/terminal are the `bash` shell and the `vzps` command itself. In the output, the PID (Process ID), TTY, TIME, and CMD fields are contained. TTY denotes which terminal the process is running on, TIME shows how much CPU time the process has used, and CMD is the name of the command that started the process.

Note: The IDs of the processes running inside Containers and displayed by running the `vzps` command on the Parallels server does not coincide with the IDs of the same processes shown by running the `ps` command inside these Containers.

As you can see, the standard `vzps` command just lists the basics. To get more details about the processes running on your server, you will need to pass some command line arguments to `vzps`. For example, using the `aux` arguments with this command displays processes started by other users (`a`), processes with no terminal or one different from yours (`x`), the user who started the process and when it began (`u`). Besides, you can pass `vzps` the `-E` switch to sort the processes by the Container IDs where they are running.

```
# vzps aux -E
USER     PID %CPU %MEM   VSZ   RSS TTY     STAT START   TIME COMMAND
root       1  0.0  0.0  1516   128 ?       S    Jul14   0:37 init
root       5  0.0  0.0     0     0 ?       S    Jul14   0:03 [ubstatd]
root       6  0.0  0.0     0     0 ?       S    Jul14   3:20 [kswapd]
#27        7  0.0  0.0     0     0 ?       S    Jul14   0:00 [bdflush]
root       9  0.0  0.0     0     0 ?       S    Jul14   0:00 [kinoded]
root    1574  0.0  0.1   218   140 pts/4 S    09:30   0:00 -bash
```

There is a lot more information now. The fields USER, %CPU, %MEM, VSZ, RSS, STAT, and START have been added. Let us take a quick look at what they tell us.

The USER field shows you which user initiated the command. Many processes begin at system start time and often list root or some system account as the USER. Other processes are, of course, run by individuals.

The %CPU, %MEM, VSZ, and RSS fields all deal with system resources. First, you can see what percentage of the CPU the process is currently utilizing. Along with CPU utilization, you can see the current memory utilization and its VSZ (virtual memory size) and RSS (resident set size). VSZ is the amount of memory the program would take up if it were all in memory; RSS is the actual amount currently in memory. Knowing how much a process is currently eating will help determine if it is acting normally or has spun out of control.

You will notice a question mark in most of the TTY fields in the `vzps aux` output. This is because most of these programs were started at boot time and/or by initialization scripts. The controlling terminal does not exist for these processes; thus, the question mark. On the other hand, the `bash` command has a TTY value of pts/4. This is a command being run from a remote connection and has a terminal associated with it. This information is helpful for you when you have more than one connection open to the machine and want to determine which window a command is running in.

STAT shows the current status of a process. In our example, many are sleeping, indicated by an S in the STAT field. This simply means that they are waiting for something. It could be user input or the availability of system resources. The other most common status is R, meaning that it is currently running.

Note: For detailed information on all `vzps` parameters, output fields, states of processes, etc., please consult the `vzps` manual pages.

You can also use the `vzps` command to view the processes inside any running Container. The example below shows you how to display all active processes inside Container 101:

```
# vzps -E 101
CTID   PID TTY          TIME CMD
101 27173 ?        00:00:01 init
101 27545 ?        00:00:00 syslogd
101 27555 ?        00:00:00 sshd
101 27565 ?        00:00:00 xinetd
101 27576 ?        00:00:03 httpd
101 27583 ?        00:00:00 httpd
101 27584 ?        00:00:00 httpd
101 27587 ?        00:00:00 crond
101 27596 ?        00:00:00 saslauthd
```

Monitoring Processes in Real Time

The `vztop` utility is rather similar to `vzps` but is usually started full-screen and updates continuously with process information. This can help with programs that may infrequently cause problems and can be hard to see with `vzps`. Overall system information is also presented, which makes a nice place to start looking for problems.

The `vztop` utility can be run on the server just as the standard Linux `top` utility. The only features that distinguish the `vztop` utility from `top` are the following:

- `vztop` allows you to use the `-E` option that monitors only the processes belonging to the Container whose processes you want to display.

- You can use the `e` interactive command to temporarily view/hide the CTIDs where the processes are running.

- You can use the `E` interactive command to set the filter on the CTID field that helps you display only the processes belonging to the given Container.

The `vztop` utility usually has an output like the following:

```
# vztop -E 101
17:54:03  up 20 days, 23:37, 4 users, load average: 2.13, 1.89, 1.75
305 processes: 299 sleeping, 3 running, 3 zombie, 0 stopped
CPU0 states: 20.1% user 51.2% system  0.0% nice 0.0% iowait 28.1% idle
CPU1 states: 21.2% user 50.0% system  0.0% nice 0.0% iowait 28.1% idle
Mem: 1031088k av, 969340k used, 61748k free, 0k shrd, 256516k buff
       509264k active,             330948k inactive
Swap: 4056360k av,   17156k used, 4039204k free      192292k cached
CTID   PID USER PR  NI   VIRT   RES  SHR S %CPU %MEM  TIME+   COMMAND
101  27173 root 16  0   1616  604 1420 S  0.0  0.1  0:01.86 init
101  27545 root 16  0   1520  624 1356 S  0.0  0.1  0:00.34 syslogd
101  27555 root 25  0   4008 1700 3632 S  0.0  0.4  0:00.04 sshd
101  27565 root 25  0   2068  860 1740 S  0.0  0.2  0:00.05 xinetd
101  27576 root 16  0   7560 3180 6332 S  0.0  0.7  0:03.78 httpd
101  27587 root 16  0   2452 1036 1528 S  0.0  0.2  0:00.34 crond
101  27596 root 25  0   4048 1184 3704 S  0.0  0.2  0:00.01 saslauthd
```

As you can see, `vztop` provides an ongoing look at the processor activity in real time (the display is updated every 5 seconds by default, but you can change that with the `d` command-line option or the `s` interactive command). It displays a list of the most CPU-intensive tasks on the system and can provide an interactive interface for manipulating processes. It can sort the tasks by CPU usage, memory usage, and runtime. Specifying 101 after the `-E` option allows you to display only those processes that are running inside Container 101 only. Besides, most features can be selected by an interactive command, for example, the `e` and `E` commands described above.

Note: In the current version of Parallels Server Bare Metal, you cannot use the `vztop` utility for monitoring processes in virtual machines.

Changing Services Mode

xinetd is a service used to start and stop a variety of data communication services. xinetd starts on the Parallels server startup and waits for a connection request from a remote client that wants to connect to the server. There can be a number of remote clients in the network, and each of them can use different network protocols to establish connection to the server. In order not to run all network services responsible for a specific protocol, which will negatively influence the system performance, the system starts only the xinetd service. This service controls all other network services and, at the connection time, it starts the corresponding service to process this connection. In such a way, xinetd saves system resources allowing you to run only those network services in the system that are really needed at the moment.

The vzsetxinetd utility allows you to switch Container services between the standalone and xinetd mode. The services that can be either standalone or dependent on xinetd are sendmail, sshd, proftpd, and courier-imap. Whereas they are xinetd-dependent by default, in order to consume less resources, you may want to make them standalone due to the following reasons:

- The CPanel application does not recognize sshd if it is dependent on xinetd;
- sendmail does not process some rules correctly if it is dependent on xinetd;
- A number of control panel applications and some others are not able to manage xinetd-based services at all.

The courier-imapd, courier-imapds, courier-pop3d, and courier-pop3ds services are provided by the courier-imap service, thus vzsetxinetd can manage these services via the courier-imap service.

Let us assume that you wish to check the mode of the sendmail service and set it to standalone if it is in the xinetd mode. First, you should check the current status of the sendmail service. To this effect, type the following command in the command line:

```
# vzsetxinetd -s 222 sendmail
```

where 222 is the Container ID, sendmail denotes the name of the corresponding service, and the -s option gets the status of the sendmail service of the Container with ID 222. The output will tell you if this service has the standalone or xinetd mode:

```
sendmail is xinetd service
```

In our case it is in the xinetd mode. Now you can change the mode of the sendmail service to standalone. To make it standalone, type the following line:

```
# vzsetxinetd 222 sendmail off
sendmail is standalone service
```

where off specifies that the sendmail service should be set to the standalone mode. The output confirms that the sendmail service is now standalone.

For more information on the vzsetxinetd utility, consult the corresponding man pages or refer to the *Parallels Command Line Reference Guide*.

Notes:

1. You cannot use the vzsetxinetd utility to change the mode of the xinetd-dependent services in Containers where the Debian 3.0 OS template is installed.

2. In the current version of Parallels Server Bare Metal, you cannot use the `vzsetxinetd` utility for managing services in virtual machines.

Determining Container Identifier 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 Parallels Server Bare Metal, you can use the `vzpid` (retrieve process ID) utility to print the Container 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 Container number for each of the processes.

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

```
# vzpid 12
Pid     VEID    Name
12       101    init
```

In our example the process with the identifier 12 has the name 'init' and is running in the Container with ID 101.

Note: You can also display the Container ID where the corresponding process is running by using the `vzps` utility.

Starting, Stopping, and Restarting Services

You can manage (i.e. start, stop, and restart) services by using the command line. For example, you wish to start the `httpd` service. To do this, execute the following command:

```
[root@ct222 /]# service httpd start
```

where `service` is the standard Linux command, `httpd` denotes the name of the corresponding service, and `start` is the command that will launch this service. In order to check that the `httpd` service was successfully launched, you can either type the following Linux command:

```
[root@ct222 /]# service httpd status
```

or use the `vzps` utility when working on your server or the `ps` utility when working inside your Containers and passing them the `x` argument. The output will tell you if the `httpd` service is running in your system or not.

C H A P T E R   6

Managing Parallels Server Bare Metal Network

The given chapter familiarizes you with the Parallels Server Bare Metal network structure, enumerates Parallels networking components, and explains how to manage these components in your working environments. In particular, it provides the following information:

- How you can manage network adapters on the Parallels server.
- What Virtual Networks are and how you can manage them on the Parallels server.
- How to create virtual network adapters inside your virtual machines and Containers and configure their parameters.
- How to connect virtual machines and Containers to different networks.

In This Chapter

Managing Network Adapters on the Parallels Server

Network adapters installed on the Parallels server are used to provide virtual machines and Containers with access to each other and to external networks. During the installation, Parallels Server Bare Metal registers all physical and VLAN network adapters available on the server. In addition to that, it creates a number of virtual network adapters on the server. Once Parallels Server Bare Metal has been successfully installed, you can perform the following operations on network adapters:

- List the adapters currently available on the server.
- Create new VLAN adapters on the server.
- Connect adapters to Virtual Networks on the server.

Note: For more information on Virtual Networks, refer to **Managing Virtual Networks** (p. 138).

These operations are described in the following subsections in detail.

Listing Adapters

You can view the physical, virtual, and VLAN network adapters existing on your Parallels server using the `vznetcfg` utility. For example, you can execute the following command to list the available adapters:

```
# vznetcfg if list
Name       Type       Network ID     Addresses
eth0       nic        Bridged        10.30.18.41/16,dhcp
br2        bridge     Bridged
br1        bridge     Host-Only
br0        bridge     Shared
vnic1      vnic       Host-Only      10.37.131.2/24
vnic0      vnic       Shared         10.37.130.2/24
```

The information on adapters is presented in the table having the following columns:

Column Name	Description
Name	The adapter name.
Type	The type of the network adapter. It can be one of the following: • `nic` denotes a physical adapter installed on the Parallels server. • `vlan` stands for a VLAN adapter available on the Parallels server. • `vnic` denotes a virtual network adapter available on the Parallels server. By default, 2 virtual adapters are created during the Parallels Server Bare Metal installation: `vnic0` and `vnic1`. Besides, a new virtual adapter is automatically created on the server when you create a new Virtual Network. • `bridge` is a virtual bridge automatically created for each Virtual Network on the Parallels server.
Network ID	The ID of the Virtual Network where the network adapter is connected. Detailed information on Virtual Networks is provided in **Managing Virtual Networks** (p. 138).
Addresses	The IP address and subnet mask assigned to the network adapter. `dhcp` denotes that the adapter gets its network parameters from a DHCP server.

Creating VLAN Adapter

Parallels Server Bare Metal allows you to create new VLAN adapters on the Parallels server. You can use these adapters later on to connect your virtual machines and Containers to any of the available Virtual Networks (for more information on Virtual Networks, turn to **Managing Virtual Networks** (p. 138). VLAN adapters can be made using the `vznetcfg vlan add` command. To create a new VLAN adapter, you should specify the VLAN ID - an arbitrary integer number which will uniquely identify the virtual LAN among other VLANs on the server - and the physical network adapter on the server to which the VLAN is to be bound. For example, you can execute the following command to make a new VLAN adapter, associate it with a VLAN having the ID of 5 (i.e. with VLAN 5), and attach the VLAN adapter to the `eth0` physical adapter on the server:

```
# vznetcfg vlan add eth0 5
```

To check that the VLAN adapter has been successfully created, execute the following command:

```
# vznetcfg if list
Name        Type     Network ID    Addresses
eth0        nic                    192.168.0.150/22,dhcp
eth0.5      vlan
```

VLAN adapters can be easily identified by the `vlan` designation shown in the `Type` column of the command output. As you can see, only one VLAN adapter currently exists on the server. It is assigned the name of `eth0.5`. This name is generated automatically on the basis of the specified VLAN ID and the name of the physical adapter to which the VLAN adapter is tied.

At any time, you can delete the `eth0.5` VLAN adapter and thus destroy VLAN 5 by issuing the following command:

```
# vznetcfg vlan del eth0.5
# vznetcfg if list
Name        Type     Network ID    Addresses
eth0        nic                    192.168.0.150/22,dhcp
```

Connecting an Adapter to a Virtual Network

Connecting an adapter on the physical server to a Virtual Network allows you to join all virtual machines and Containers included in the Virtual Network to the network where the corresponding adapter is connected.

Let us assume the following:

- The `eth1` physical adapter and the `vznetwork1` Virtual Network exist on the Parallels server. For information on how to create Virtual Networks, see **Creating a Virtual Network** (p. 139).

- The `eth1` physical adapter is connected to the local network.

- The `MyVM` virtual machine is connected to the `vznetwork1` Virtual Network. Detailed information on how to join virtual machines and Containers to Virtual Networks is given in **Connecting Containers to Virtual Networks** (p. 150) and **Connecting Virtual Machines to Virtual Networks** (p. 154).

To connect the `eth1` adapter to the `vznetwork1` Virtual Network and thus to join the `MyVM` virtual machine to the network behind `eth1`, run this command on the server:

```
# vznetcfg net addif vznetwork1 eth1
```

To check that the `eth0` physical adapter has been successfully added to the `vznetwork1` Virtual Network, you can execute the following command:

```
# vznetcfg if list
Name      Type     Network ID    Addresses
eth1      nic      vznetwork1    192.168.0.170/22,dhcp
...
```

As you can see, the `eth0` adapter is now joined to the `vznetwork1` Virtual Network, which means that the `MyVM` virtual machine whose virtual network adapters is connected to `vznetwork1` can access the local network behind `eth1`.

At any time you can disconnect the `eth1` physical adapter from the `vznetwork1` Virtual Network (and thus detach the `MyVM` virtual machine from the local network) by running the following command:

```
# vznetcfg net delif eth1
```

To check that the physical adapter has been successfully disconnected from `vznetwork1`, issue the following command:

```
# vznetcfg if list
Name      Type     Network ID    Addresses
eth1      nic                    192.168.0.170/22,dhcp
...
```

Managing Virtual Networks

A Virtual Network acts as a binding interface between a virtual network adapter inside a virtual machine and Container and the corresponding network adapter on the Parallels server, which allows you to include your virtual machines and Containers in different networks. In Parallels Server Bare Metal, you can manage Virtual Networks as follows:

- Create a new Virtual Network and remove an existing one.
- Configure the parameters of an existing Virtual Network.
- List the existing Virtual Networks.
- Delete a Virtual Network that you do not need any more.

These operations are described in the following subsections in detail.

Creating a Virtual Network

Virtual Networks serve as binding interfaces between the virtual network adapters inside virtual machines and Containers and the physical, VLAN, and virtual network adapters on the Parallels server. Using Virtual Networks, you can connect your virtual machines and Containers to different networks.

By default, Parallels Server Bare Metal creates the following Virtual Networks:

- *Bridged*. This Virtual Network is connected to one of the physical adapters on the Parallels server (as a rule, `eth0`) and provides virtual machines and Containers included in this Virtual Network with access to the network behind this physical adapter.

- *Shared*. This Virtual Network is connected to the `vnic0` virtual adapter on the Parallels server and allows virtual machines and Containers included in this Virtual Network to use the current network connections of your Parallels server.

- *Host-only*. This Virtual Network is connected to the `vnic1` virtual adapter on the Parallels server and allows a virtual machine and Container included in this Virtual Network to access only the Parallels server and the other virtual machines and Containers on this network.

You can also create your own Virtual Networks using the `prlsrvctl` or `vznetcfg` utility. For example, to make a new Virtual Network with the name of `vznetwork1`, you can issue one of the following commands:

```
# vznetcfg net new vznetwork1
```

or

```
# prlsrvctl net add vznetwork1
```

By default, both commands create host-only Virtual Networks. However, you can change their types using the `prlsrvctl` utility (see **Configuring Virtual Network Parameters** (p. 140) for details).

In the current version of Parallels Server Bare Metal, you can create

- 1 shared Virtual Network (it is automatically created on the Parallels server during the Parallels Server Bare Metal installation).

- 5 host-only Virtual Networks (1 host-only Virtual Network is automatically created on the Parallels server during the Parallels Server Bare Metal installation).

- One or more bridged Virtual Networks. The number Virtual Networks depends on the number of physical and VLAN adapters available on the Parallels server. One Virtual Network can be connected to only one physical or VLAN adapter.

Viewing Bridges

Each Virtual Network is associated with some bridge which is automatically made on the Parallels server during the Virtual Network creation and serves as the basis for the Virtual Network functioning. To find out what bridge is associated with what Virtual Network, you can run the following command:

```
# vznetcfg if list
Name         Type         Network ID     Addresses
eth0         nic          Bridged        10.30.18.41/16,dhcp
br3          bridge       vznetwork1
br2          bridge       Bridged
br1          bridge       Host-Only
```

```
br0          bridge      Shared
vnic1        vnic        Host-Only      10.37.131.2/24
vnic0        vnic        Shared         10.37.130.2/24
vnic2        vnic        vznetwork1     10.37.132.2/24
```

The bridges existing on the Parallels server are listed in the `Name` column and can be easily identified by the `br` prefix. For example, you can see that the `br3` bridge is currently associated with the `vznetwork1` Virtual Network.

Configuring Virtual Network Parameters

Parallels Server Bare Metal allows you to configure the following parameters for a Virtual Network:

- the name assigned to the Virtual Network
- the networking mode in which the Virtual Network is operating
- the description of the Virtual Network

All these operations can be performed using the `prlsrvctl` utility. Let us assume that you want to configure the `vznetwork1` Virtual Network. This Virtual Network is currently configured as a host-only network, attached to the `vnic2` adapter, and has the following description: `This is a host-only Virtual Network`. To change these parameters, you can execute the following command:

```
# prlsrvctl net set vznetwork1 -n psbm_network1 -t bridged --ifname eth1 -d
"This is a bridged Virtual Network"
```

This command sets the following parameters for the `vznetwork1` Virtual Network:

- Changes the Virtual Network name to `psbm_network1`.
- Changes the Virtual Network type to bridged. To do this, `prlsrvctl` detaches the Virtual Network from the `vnic2` adapter and connects it to the `eth1` physical adapter on the Parallels server.
- Changes the Virtual Network description to the following: `This is a bridged Virtual Network`. You can view this description in Parallels Management Console.

For more information on the `prlsrvctl` utility, refer to the *Parallels Command Line Reference Guide*.

Listing Virtual Networks

Sometimes, you may wish to list the Virtual Networks existing on the Parallels server. To do this, you can use either the `vznetcfg` or `prlsrvctl` utility.

Listing Virtual Networks With vznetcfg

To list the Virtual Networks on your server using the `vznetcfg` utility, execute the following command:

```
# vznetcfg net list
Network ID      Status      Master Interface  Slave Interfaces
Shared          active      vnic0
Host-Only       active      vnic1
Bridged         active      eth0
vznetwork1      active      vnic2
```

In the example above, 4 Virtual Networks - `vznetwork1` and 3 default Virtual Networks - exist on the Parallels server. The information on these Virtual Networks is presented in the table having the following columns:

Column Name	Description
Network ID	The ID assigned to the Virtual Network.
Status	Indicates the status of the Virtual Network. It can be one of the following: ▪ `active`: the Virtual Network is up and running. ▪ `configured`: the information on the Virtual Network is present in the `/etc/vz/vznet.conf` file on the server, but the bridge to which the Virtual Network is bound is down or absent from the server. **Note:** Detailed information on the `vznet.conf` file is given in the *Parallels Command Line Reference Guide*.
Master Interface	Displays the adapter on the server connected to the Virtual Network, if any.
Slave Interfaces	Lists the adapters in virtual machines and Containers joined to the Virtual Network, if any.

Listing Virtual Networks With prlsrvctl

You can also use the `prlsrvctl` utility to list the Virtual Networks existing on your server. To do this, run the following command:

```
# prlsrvctl net list
Network ID      Type        Bound To
Shared          shared      vnic0
Host-Only       host-only   vnic1
Bridged         bridged     eth0
vznetwork1      host-only   vnic2
```

This utility displays the following information on Virtual Networks:

Column Name	Description

Network ID	The name assigned to the Virtual Network.
Type	The networking mode set for the Virtual Network.
Bound To	The adapter on the Parallels server connected to the Virtual Networks, if any.

Deleting a Virtual Network

At any time, you can remove a Virtual Network that you do not need any more from the physical server. To do this, you can use both the `vznetcfg` and `prlsrvctl` utilities. For example, you can delete the `vznetwork1` Virtual Network by running one of the following commands:

```
# vznetcfg net del vznetwork1
```

or

```
# prlsrvctl net del vznetwork1
```

To check that `vznetwork1` has been successfully removed, execute one of these commands:

```
# vznetcfg net list
Network ID       Status        Master Interface  Slave Interfaces
Shared           active        vnic0
Host-Only        active        vnic1
Bridged          active        eth0
```

or

```
# prlsrvctl net list
Network ID       Type          Bound To
Shared           shared        vnic0
Host-Only        host-only     vnic1
Bridged          bridged       eth0
```

Note: Detailed information on the `vznetcfg` and `prlsrvctl` utilities is provided in the *Parallels Command Line Reference Guide* and their manual pages.

Managing Adapters in Containers

Parallels Server Bare Metal provides you with ample opportunities of configuring virtual network adapters inside Containers and including them in different network environments. The given section starts with the explanation of the two network modes - `venet0` and `veth` - in which any Container can operate and then shows you the way to perform the following operations:

- Create new virtual network adapters inside your Containers and delete existing ones.
- Configure the parameters of an existing virtual network adapter (e.g. assign an IP address to it).
- Join Container virtual network adapters to Virtual Networks, thus, connecting them to different networks.

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

Container Networking Modes

In Parallels Server Bare Metal, any Container can operate in one of the two operating modes:

- *venet0* mode
- *veth* mode

Detailed information on these operating modes is provided in the following subsections.

venet0 Mode

By default, all the Containers on the server are operating in the `venet0` mode, which means that they are connected among themselves and with the server using a virtual network adapter called `venet0`. The picture below provides an example of the network structure when all Containers (`Container #1`, `Container #2`, `Container #3`) are functioning in the `venet0` mode:
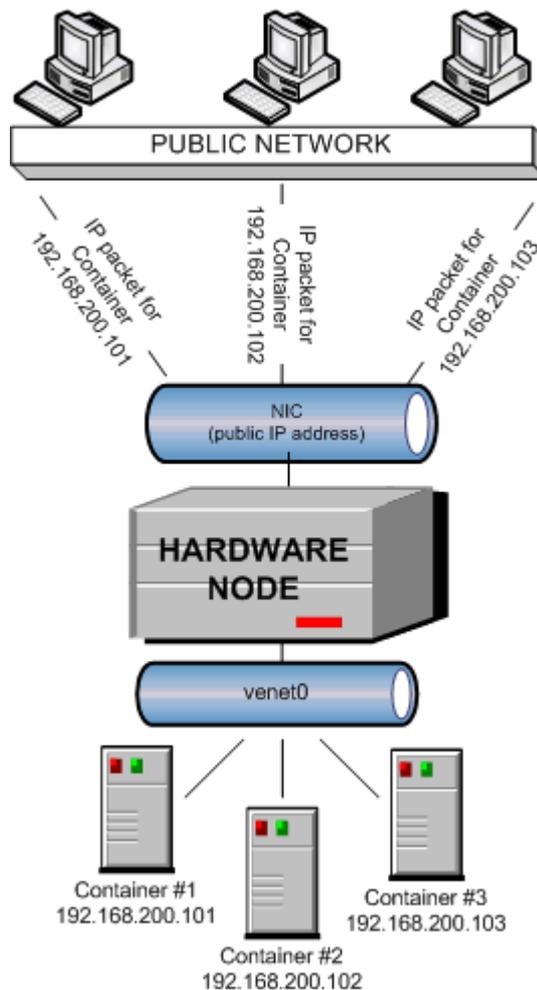
Figure 1: Networking - venet0 Mode

All Containers on the server use the `venet0` virtual adapter as the default gateway to send and receive data to/from other networks (shown as the *PUBLIC NETWORK* in the picture above). The procedure of handling incoming and outgoing IP packets may be described as follows:

- All IP packets from Containers operating in the `venet0` mode come to this adapter and are redirected through a public IP address of the server to the corresponding server on the public network.

- All IP packets coming from external networks and destined for Container IP addresses reach the public IP address of the server first and, afterwards, are sent through `venet0` to the IP addresses of the corresponding Containers.

The `venet0` adapter is also used to exchange the traffic among all the Containers hosted on the given server. All the network traffic of a Container is isolated from that of the other Containers, i.e. all Containers are protected from each other in the way that makes traffic snooping impossible.

veth Mode

You can also create special `veth` virtual adapters inside your Containers and make the Containers operate in the `veth` mode. The following figure represents an example of the network structure where all Containers (`Container#1` and `Container#2`) are operating in the `veth` mode:
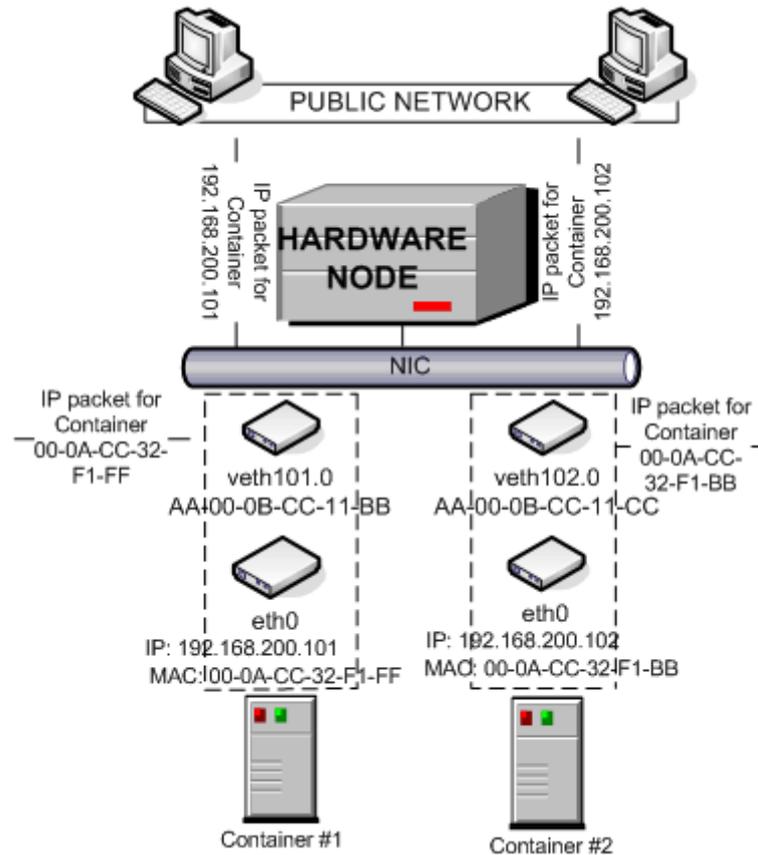
Figure 2: Networking - veth Mode

In the `veth` mode, a separate `veth` virtual adapter is created for each Container on the server. You are allowed to create several `veth` adapters for a Container. Any `veth` virtual adapter consists of two interfaces:

- An Ethernet interface inside the Container. This interface represents a counterpart of a physical network adapter installed on a standalone server. As any other physical adapter, it has a MAC address (e.g., `00-0A-CC-32-F1-FF` and `00-0A-CC-32-F1-BB`), can be assigned one or more IP addresses (e.g., `192.168.200.101` and `192.168.200.102`) and included in different network environments, etc. Refer to the **Configuring veth Adapter Parameters** section (p. 149) for detailed information on configuring Ethernet interfaces inside Containers.

- An Ethernet interface on the server. This interface is responsible for the adapter operation in the server context and mostly used to maintain the interaction and communication between the server and the Ethernet interface inside the Container. Each Ethernet interface on the server should be assigned a MAC address (e.g., `AA-00-0B-CC-11-BB` and `AA-00-0B-CC-11-CC`). Detailed information on how to manage Ethernet interfaces on the server is provided in the **Configuring veth Adapter Parameters** section (p. 149).

Both interfaces are closely linked to each other, which means that an IP packet entering one interface will always come out from the other one.

Differences Between venet0 and veth Modes

The veth mode demonstrates the following differences as compared to the venet0 mode:

- Each of the Ethernet interfaces constituting a veth virtual adapter has a MAC address assigned to it while venet0 does not have any. Thanks to this fact:

  - Any Container can see all broadcast and multicast packets received from or sent to the selected network adapter on the server.

  - Using a veth virtual adapter inside a Container allows you to host a DHCP or Samba server inside this Container, etc.

- There is no more need to assign all network settings (IP addresses, subnet mask, gateway, etc.) to a Container from the Host OS. All network parameters can be set from inside the Container.

- veth adapters can be bridged among themselves and with other devices. If several veth adapters are united into a bridge, this bridge can be used to handle network traffic for the Containers whose veth adapters are included in the bridge.

- Due to the fact that veth adapters act as full members on the network (rather than 'hidden' beyond venet0), they are more prone to security vulnerabilities: traffic sniffing, IP address collisions, etc. Therefore, veth adapters are recommended to be used in trusted network environments only.

- The veth mode has poorer scalability than the venet0 mode. This is caused by the fact that any broadcast packet meant for any veth virtual network adapter is duplicated and transmitted to all available veth network adapters, which requires the CPU(s) on the server to process all the resulting broadcast packets and may noticeably degrade the system performance. So, we highly recommend that you create no more than 100 veth network adapters for every CPU on the server.

Creating and Deleting veth Network Adapters

By default, any Container on the Parallels server starts functioning in the `venet0` mode right after its creation. However, at any time you can create additional virtual adapters for your Container and set them to work in the `veth` mode. This can be done by using the `--netif_add` option of the `pctl set` command.

Let us assume that you wish to create a new virtual adapter with the name of `eth1` inside Container 101 and make it function in the `veth` mode. To do this, you can execute the following command :

```
# pctl set 101 --netif_add eth1 --save
Saved parameters for Container 101
```

The settings of the newly created virtual adapter are saved as the value of the `NETIF` parameter in the configuration file of Container 101 (`/etc/vz/conf/101.conf`). So, you can use the following command to display the parameters assigned to the `veth` network adapter inside Container 101:

```
# grep NETIF /etc/vz/conf/101.conf
NETIF="ifname=eth1,mac=00:10:41:F0:AA:B6,host_mac=00:18:51:A0:8A:D7"
```

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

- `ifname`: the name set for the `veth` Ethernet interface inside Container 101. You specified this name when creating the Container virtual network adapter. Usually, names of Ethernet interfaces inside Containers are set in the form of eth*Ad_N* where *Ad_N* denotes the index number of the created adapter (e.g. `eth0` or `eth1`); however, you can choose any other name you like and specify it during the virtual adapter creation.
- `mac`: the MAC address assigned to the `veth` Ethernet interface inside Container 101.
- `host_mac`: the MAC address assigned to the `veth` Ethernet interface on the Parallels server.

`ifname` is the only mandatory parameter that should be indicated when creating a Container virtual network adapter. All the other parameters are optional and generated by Parallels Server Bare Metal automatically, if not specified.

At any time, you can remove the `veth` virtual network adapter inside Container 101 by executing the following command:

```
# pctl set 101 --netif_del eth1 --save
Saved parameters for Container 101
# grep NETIF /etc/vz/conf/101.conf
NETIF=""
```

Configuring veth Adapter Parameters

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

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

Note: For detailed information on all parameters that can be configured for each default Container network adapter (i.e. for the adapter operating in the `venet0` mode), refer to **Performing Initial Configuration** (p. 30).

```
# pctl set 101 --ifname eth1 --ipadd 192.168.144.123 --save
Saved parameters for Container 101
```

This command will set an IP address of `192.168.144.123` for the `eth1` adapter inside Container 101. If you wish to use the Dynamic Host Configuration Protocol (DHCP) to make the `eth1` adapter of Container 101 automatically receive TCP/IP configuration settings, you can issue the following command instead:

```
# pctl set 101 --ifname eth1 --dhcp yes --save
Saved parameters for Container 101
```

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

```
# pctl set 101 --ifname eth1 --ipdel 192.168.144.123 --save
Saved parameters for Container 101
```

You can also delete all IP addresses set for Container 101 at once:

```
# pctl set 101 --ifname eth1 --ipdel all --save
Saved parameters for Container 101
```

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

▪   one or more DNS servers that the Container virtual adapter is supposed to use:

```
# pctl set 101 --ifname eth1 --nameserver 192.168.100.111 --save
Saved parameters for Container 101
```

▪   and a gateway to be used for routing the traffic of the Container virtual adapter:

```
# pctl set 101 --ifname eth1 --gateway 192.168.111.1 --save
Saved parameters for Container 101
```

Detailed information on all options which can be used with the `pctl set` command to manage Container adapter parameters is given in the *Parallels Command Line Reference Guide* and the `pctl` manual pages.

Connecting Containers to Virtual Networks

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

1   Creating a Virtual Network that will act as an intermediary between the `veth` adapters and the physical/VLAN adapter.

2   Connecting the `veth` virtual adapters you want to include in an Ethernet network/VLAN to the Virtual Network.

3   Joining the Virtual Network where the `veth` virtual adapters are included to the corresponding physical/VLAN adapter.

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

The process of creating new Virtual Networks and joining physical and VLAN adapters to these Virtual Network is described in the **Creating a Virtual Network** (p. 139) and **Connecting an Adapter to a Virtual Network** (p. 137) subsections, respectively. So, in the example below we assume the following:

▪   The `eth0` physical adapter and the `vznetwork1` Virtual Network exist on the server.

▪   The `eth0` physical adapter is connected to the local Ethernet network and to the `vznetwork1` Virtual Network.

▪   You want to connect Container 101 and Container 102 to the local Ethernet network.

To join Container 101 and 102 to the local Ethernet network behind the `eth0` adapter, you should connect these Containers to the `vznetwork1` Virtual Network. This can be done as follows:

1   Find out the name of the `veth` Ethernet interfaces inside Container 101 and 102:

```
# vzlist -a -o ctid,ifname
  CTID  IFNAME
  101  eth1
  102  eth0
  103  -
```

The command output shows that the `veth` Ethernet interfaces inside Container 101 and 102 have the names of `eth1` and `eth0`, respectively.

Note: To add a `veth` adapter to a Virtual Network, you must use the name of its Ethernet interface inside the Container.

2   Join the `veth` adapters to the `vznetwork1` Virtual Network:

▪   Add the `veth` adapter of Container 101 to the Virtual Network:

```
# pctl set 101 --ifname eth1 --network vznetwork1 --save
Saved parameters for Container 101
```

▪   Add the `veth` adapter of Container 102 to the Virtual Network:

```
# pctl set 102 --ifname eth0 --network vznetwork1 --save
```

```
Saved parameters for Container 102
```

After completing these tasks, Container 101 and Container 102 will be able to access any of the servers in the network where the `eth0` physical adapter is connected.

At any time, you can disconnect the `veth` virtual network adapters of Container 101 and 102 from the `vznetwork1` Virtual Network by executing the following commands:

- To disconnect the `veth` adapter of Container 101 from the Virtual Network:

```
# pctl set 101 --ifname eth1 --network "" --save
Saved parameters for Container 101
```

- To disconnect the `veth` adapter of Container 102 from the Virtual Network:

```
# pctl set 102 --ifname eth1 --network "" --save
Saved parameters for Container 102
```

Managing Adapters in Virtual Machines

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

- Create new virtual network adapters and delete existing ones.
- Configure the parameters of an existing virtual network adapter (e.g. assign an IP address to it).
- Join virtual network adapters to Virtual Networks.

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

Creating and Deleting Virtual Adapters

A virtual machine can have up to 16 virtual network adapters. Each adapter can be connected to a different network. Let us assume that you wish to create a new virtual adapter for the MyVM virtual machine. To do this, you can execute the following command :

```
# pctl set MyVM --device-add net
Creating net1 (+) type=shared iface='default' mac=XXXXXXXXXXX
The VM has been successfully configured.
```

To check that the network adapter (net1) has been successfully added to the virtual machine, run this command:

```
# pctl list --info MyVM
ID: {f3b3d134-f512-324b-b0b1-dbd642f5220b}
Name: Windows XP
...
  net0 (+) type=shared iface='default' mac=001C42566BCF
  net1 (+) type=shared iface='default' mac=001C42AF3D69
```

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

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

For the full of options that can be used when creating a new virtual network adapter, refer to the *Parallels Command Line Reference Guide*.

Configuring Virtual Adapter Parameters

Parallels Server Bare Metal allows you to configure the following parameters of virtual machine adapters:

Configuring the MAC Address

If you need for some reason to regenerate the current MAC address of a network adapter, you can use the following command:

```
# pctl set MyVM --device-set net1 --mac 00:1C:42:2D:74:00
Creating net1 (+) network=Bridged mac=001C422D7400
The VM has been successfully configured.
```

This command sets the MAC address of `00:1C:42:2D:74:00` MAC address for the `net1` adapter in the `MyVM` virtual machine. If do not know what MAC address to assign to your virtual adapter, you can make `pctl set` automatically generate a new MAC address. To do this, run the following command:

```
# pctl set MyVM --device-set net1 --mac auto
Creating net1 (+) network=Bridged mac=001C42C84F3E
The VM has been successfully configured.
```

Configuring the IP Parameters

As any other standalone server, each virtual machine must have a number of TCP/IP settings configured in the proper way to successfully operate on the network. These settings include:

- an IP address for each virtual network adapter inside the virtual machine
- the default gateways to be used by the virtual machine
- the default DNS servers to be used by the virtual machine

Usually, you define all these settings during the virtual machine creation. However, if you have not yet set any of the settings or want to modify any of them, you can use the `pctl set` command. For example, you can execute the following command to assign the IP address of `192.129.129.20` to the `net1` adapter in the `MyVM` virtual machine and set for it the default gateway with the IP address `192.129.129.1` and the DNS server with the IP address of `192.192.192.10`:

```
# pctl set MyVM --device-set net1 --ipadd 192.129.129.20 --gw 192.129.129.1 --
nameserver 192.192.192.10
```

Along with a static assignment of network parameters to a virtual adapter, you can make the adapter receive its TCP/IP settings automatically using the Dynamic Host Configuration Protocol (DHCP). For example, you can run this command to make the `net1` adapter in the `MyVM` virtual machine get its IP settings through DHCP:

```
# pctl set MyVM --device-set net1 --dhcp yes
Creating net1 (+) network=Bridged mac=001C42C84F3E
Enable automatic reconfiguration for this network adapter.
The VM has been successfully configured.
```

Detailed information on all options which can be used with the `pctl set` command to manage virtual machine adapter parameters is given in the *Parallels Command Line Reference Guide* and the `pctl` manual pages.

Connecting Virtual Machines to Virtual Networks

In Parallels Server Bare Metal, you can connect your virtual machines to Virtual Networks of the following types:

- *Bridged networks*. This type of Virtual Networks allows the virtual machine to use one of the physical server's network adapters, which makes it appear as a separate computer on the network the corresponding adapter belongs to.

- *Shared networks*. This type of Virtual Networks allows the virtual machine to use the current network connections of your Parallels server.

- *Host-only networks*. This type of Virtual Networks allows the virtual machine to access only the Parallels server and the virtual machines joined to this network.

To connect your virtual machines to any of these networks, use the `pctl set` command. For example, the following session shows you how to connect the `net0` adapter of the `MyVM` virtual machine to the `Bridged` Virtual Network (this is one of the default Virtual Networks created on the Parallels server during the Parallels Server Bare Metal installation).

Before connecting the `MyVM` virtual machine to the `Bridged` Virtual Network, you may wish to check the network adapter associated with this Virtual Network. You can do it, for example, using the following command:

```
# prlsrvctl net list
Network ID      Type          Bound To
Shared          shared        vnic0
Host-Only       host-only     vnic1
Bridged         bridged       eth0
vznetwork1      host-only     vnic2
```

From the command output, you can see that the `Bridged` Virtual Network is attached to the `eth0` physical adapter on the Parallels server. It means that, after connecting the `MyVM` virtual machine to the `Bridged` Virtual Network, the virtual machine will be able to access all the computers on the network where the `eth0` adapter is connected.

Now you can run the following command to join the `net0` adapter of the `MyVM` virtual machine to the `Bridged` Virtual Network:

```
# pctl set MyVM --device-set net0 --network Bridged
Creating net0 (+) network=Bridged mac=001C422D7493
The VM has been successfully configured.
```

C H A P T E R 7

Managing Licenses

The given chapter provides information on managing Parallels Server Bare Metal licenses. In particular, you will know how to view the current license status, to install a new license on your server or to update an existing one, to transfer the license from one server to another, etc.

In This Chapter

Installing the License

Depending on the way you have obtained your Parallels Server Bare Metal license, it can be installed on the Parallels server as follows:

- If you have obtained the license in the form of a product key, you can install it on the server using the -p option of the vzlicload command. For example, you can execute the following command to install the XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX product key:

```
# vzlicload -p 5BVMF2-560MM0-D28DQA-B59NTE-10H4HG
Processing product key "XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX"...
License VZSRV was loaded successfully
---
 1 of 1 licenses was loaded
```

> **Note:** You can also use the vzlicload utility to upgrade the license. For example, this may be necessary if your current license does not support using Parallels Virtual Automation for managing Parallels servers and their virtual machines and Containers.

- If you have obtained the license in the form of an activation code, you can install it on the server using the -a option of the vzlicupdate command. For example:

```
# vzlicupdate -a XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX
```

  where XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX is your activation code. When executed, vzlicupdate connects to the Parallels Key Authentication (KA) licensing server and transmits the specified activation code there. In its turn, the licensing server generates a license file, sends it back to the server from where the activation code has been dispatched, and automatically installs it on this server. So, before executing the aforementioned command, make sure that your Parallels server is connected to the Internet.

If you are activating your installation by means of an activation key, you must have an active Internet connection to successfully complete the license installation. Otherwise, you will be presented with the corresponding warning message informing you of the steps you have to take to activate your license. As a rule, these steps include the following:

1   Visiting the http://www.parallels.com/en/support/virtuozzo/activate web page and activating the license manually.

2   Providing the following information on this web page:

- In the Product Code field, specify your license activation code.

- In the HWID field, provide the ID of your server.

- In the Enter following digits field, type the digits displayed next to this field.

3   Clicking the ACTIVATE LICENSE button.

If you have entered the correct information on the Virtuozzo License Activation page, you will be provided with a link to a license file that you should download to and install on the server. For example, you can run this command to install the obtained license file

```
# vzlicload -f /etc/vzlicense
```

This command will install the license file with the name of vzlicense on your server.

Updating the Current License

In Parallels Server Bare Metal, you can use the `vzlicupdate` utility to update the license currently installed on the Parallels server. When executed, the utility tries to connect to the Parallels Key Authentication (KA) server and to retrieve a new license and install it on the server. To update your license, do the following:

1 Make sure that the Parallels server where you wish to update the license is connected to the Internet.

2 Execute the following command on the server:

```
# vzlicupdate
Start updating license [6E62.3D01.6BEC.E8D7.CE42.4517.68CB.E102]
...
```

By default, `vzlicupdate` tries to access the KA server having the hostname of `ka.parallels.com`. However, you can explicitly specify what KA server to use using the `--server` option:

```
# vzlicupdate --server ka.server.com
```

In this case, the `vzlicupdate` utility will try to connect to the KA server with the hostname of `ka.server.com`, to get a new license from this server, and to install it on the server where `vzlicupdate` has been executed.

Transferring the License to Another Server

Sometimes, you may wish to transfer licenses from one Parallels server (*source server*) to another (*destination server*). For example, this may be the case if the server where the license is installed starts experiencing problems or requires the hardware upgrade.

The procedure of transferring a license from one Parallels server to another depends on the license type and can be one of the following:

▪ If you have activated your Parallels Server Bare Metal installation by means of a product key, you can transfer the installed license from the source to the destination server as follows:

   ▪ Remove the installed license from the source server (e.g. using the `vzlicload -r product_key` command).

   ▪ Log in to the destination server.

   ▪ Install the product key on the destination server. Detailed information on how to install Parallels Server Bare Metal licenses is provided in **Installing a License** (p. 156).

▪ If you have activated your Parallels Server Bare Metal installation by means of an activation code, you can use the `vzlicupdate` utility to move licenses between Parallels servers. For example, to transfer a license that has been installed using the `XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX` activation code, do the following:

   1. Ascertain that the source server is shut down, or the license is removed from this server.

2. Make sure that the destination server is up and connected to the Internet.

3. Log in to the destination server (e.g. via `ssh`).

4. Execute the following command on the destination server:

```
# vzlicupdate -t -a XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX
```

When executed, `vzlicupdate` sends the activation code to the Parallels KA server, thus informing the server of its intention to transfer the license to a new Parallels server. The KA server verifies the received code, generates a new license file, sends it back to the destination server, and installs it there.

You can check that the license transferal has completed successfully using the `vzlicview` utility. For example:

```
# vzlicview
Show installed licenses...
VZSRV
        status="ACTIVE"
        version=4.0
        serial="XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX"
        expiration="05/01/2009 23:59:59"
         ...
```

Detailed information on the `vzlicview` utility and its output is provided in **Viewing Current License** .

Viewing the Current License

The given subsection familiarizes you with the way to view the information on the license installed on your Parallels server.

Viewing the License

In Parallels Server Bare Metal, you can use the `vzlicview` utility to view the information on the installed license and find out its current status. When executed, this utility processes the license currently installed on the Parallels server and prints the license contents along with its status. A sample output of `vzlicview` is given below:

```
# vzlicview
Show installed licenses
VZSRV
       status="ACTIVE"
       version=4.0
       serial="XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX"
       expiration="12/01/2006 23:59:59"
       graceperiod=86400 (86400)
       key_number="PSBM.00000001.0000"
       cpu_total=64 (1)
       ct_total=8200 (1)
       max_vzmcpmc_users=128
       max_pim_users=260
       platform="Any"
       product="PSBM"
       vzpp_allowed=1
       backup_mgmt_allowed=1
       workflow_mgmt_allowed=1
       vzagent_allowed=1
      nr_vms=10
       architecture="Any"
```

The command output shows the full information about the license. The main license parameters are listed in the following table:

Column Name	Description
status	The license status. The information on all possible license statuses is provided in License Statuses (p. 160).
version	The version of Parallels Server Bare Metal with which the license is compatible.
serial	The license serial number.
expiration	The license expiration date, if it is time-limited.
graceperiod	The period during which Parallels Server Bare Metal continues functioning after your license has expired, in seconds.
key_number	The number under which the license is registered on the Parallels Key Authentication server.
cpu_total	The total number of central processor units (CPUs) which can be installed on the Parallels server.
ct_total	The total number of virtual machines and Containers which can simultaneously run on the Parallels server.
max_vzmc_users	The number of users able to simultaneously connect to the server using Parallels Management Console.
max_vzcc_users	The number of users able to simultaneously connect to the server using Parallels Virtual Automation (formerly, Parallels Infrastructure Manager).
platform	The operating system with which the license is compatible.

`product`	The product name for which the license has been issued.
`vzpp_allowed`	Indicates whether you can manage virtual machines and Containers using Parallels Power Panel: ▪ `1`: the 'Parallels Power Panel' functionality is enabled ▪ `0`: the 'Parallels Power Panel' functionality is disabled
`backup_mgmt_allowed`	Indicates whether the 'backup' functionality is enabled for the given server: ▪ `1`: the 'backup' functionality is enabled ▪ `0`: the 'backup' functionality is disabled
`workflow_mgmt_allowed`	Indicates whether the 'virtual machine and Container requesting' functionality is enabled for the given server: ▪ `1`: the 'virtual machine and Container requesting' functionality is enabled ▪ `0`: the 'virtual machine and Container requesting' functionality is disabled
`vzagent_allowed`	Indicates whether you are allowed to use the Parallels Agent functionality on the given server: ▪ `1`: the Parallels Agent functionality is enabled ▪ `0`: the Parallels Agent functionality is disabled
`nr_vms`	The number of virtual machines which can simultaneously run on the Parallels server.
`architecture`	The system architecture with which the license is compatible.

License Statuses

When viewing information on your license, pay special attention to the license status that can be one of the following:

ACTIVE	The license installed on the server is valid and active.
VALID	The license the utility parses is valid and can be installed on the server.
EXPIRED	The license has expired and, therefore, could not be installed on the server.
GRACED	The license has been successfully installed on the server; however, it has expired and is currently on the grace period (i.e. it is active till the end of the grace period).
INVALID	The license is invalid (for example, because of the server architecture mismatch) or corrupted.

C H A P T E R   8

Keeping Your System Up To Date

This chapter explains the ways to keep your Parallels server up to date. The components you need to take care of are the following:

- Parallels Server Bare Metal software
- virtual machines and Containers created on the Parallels server

In This Chapter

Updating Parallels Server Bare Metal Software

Parallels Server Bare Metal is constantly developing: there appear new versions of the Parallels Server Bare Metal core and of existing utilities, OS and application templates are perfected, new templates and utilities are also added from time to time. Thus, Parallels Server Bare Metal may sometimes be repackaged to include the latest changes in any of its parts. As these changes grow in number, new product versions are released.

Parallels Server Bare Metal provides a special utility - `vzup2date` - allowing you to easily and quickly update your Parallels server. The main components that need to be updated are the following:

- Parallels Server Bare Metal system software (packages built by Parallels)
- Parallels Server Bare Metal templates

`vzup2date` is intended to relieve Parallels Server Bare Metal administrators of the necessity to manually update existing Parallels Server Bare Metal installations. It provides a single information channel for learning if updated product versions are available. In other words, a regular launching of this utility helps ensure that you always have the latest version of Parallels Server Bare Metal installed.

The `vzup2date` utility can be launched in two modes:

- Graphical mode. In this mode, you use a special wizard to update either the Parallels Server Bare Metal system files or templates depending on the options passed to `vzup2date`.
- Command line mode containing two submodes:
  - the *batch* submode
  - the *messages* submode

  In comparison to the graphical mode, the command line mode provides more possibilities for the Parallels Server Bare Metal updates management (e.g. the ability to use special filters while selecting updates for your system).

Both modes are described in the following subsections in detail.

Updating in Graphical Mode

In the graphical mode, the `vzup2date` utility can be launched in two submodes. If invoked without any parameters or with the `-s` switch, it is supposed to check and, if necessary, download and install Parallels Server Bare Metal system files. On the other hand, specifying the `-z` option when invoking the utility tells it to perform the same operations for OS and application EZ templates. There is no single interface for checking system files and templates at once, as these operations are different in nature. Therefore, you should consecutively call the `vzup2date` utility with and without the `-z` option, if you wish to check for all available system and template updates.

Note: You can explicitly specify that the `vzup2date` utility is to be run in the graphical mode by passing the `-m interactive` switch to it.

The `vzup2date` utility is implemented as a wizard, the first few steps of which are common for both modes. After you launch the utility from the command line, you will be presented with the Welcome screen. In this window, you can do one of the following:

▪ Click the Next button to connect to the Parallels default repository.

▪ Click the Configure button to display the current settings used to connect to the repository housing Parallels Server Bare Metal updated packages and templates and to configure it, if necessary:
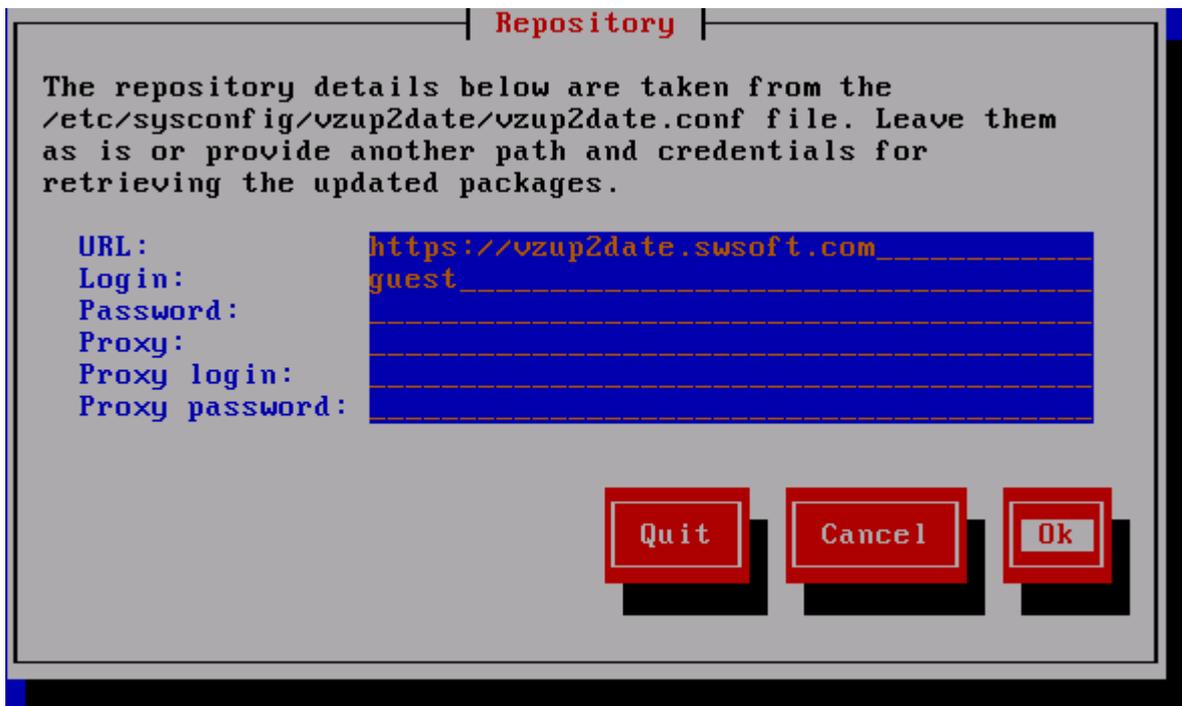
The information on this screen is taken from the /etc/sysconfig/vzup2date/vzup2date.conf file on the Parallels server. If you wish to change this information and save the changes to the configuration file, enter the correct settings into the fields provided, and press OK.

Once you press Next in the Welcome window, the utility will try to connect to the specified repository (either the Parallels default repository or your own one) and, if the connection is successful, display the next screen, which will vary depending on the mode of the vzup2date invocation. First, we will describe the mode of updating Parallels Server Bare Metal system files and then proceed with updating your EZ templates.

Note: The vzup2date utility might see that the selected update includes an updated version of the vzup2date utility itself. In this case you will first have to perform an update of this utility and then to re-launch it and select the desired Parallels Server Bare Metal system update once again.

Updating System Files

After the `vzup2date` utility has checked the repository and found any updates, you are presented the following window:

This window displays the list of updates that can be installed on your Parallels server. If you want to update to the latest Parallels Server Bare Metal core and utilities versions, just press Next on this screen, and the `vzup2date` utility will download and install them asking your confirmation before each action.

On the other hand, if you have a reason not to install the latest updates for both the Parallels Server Bare Metal core and utilities, press Customize. In this case, you will be able to choose whether to perform customization on the Parallels Server Bare Metal core or utilities. This step will be skipped if updates are currently available either only for the core or only for utilities. On the next step, you will be asked to choose the desired core or utilities updates, in case there are many.

```
┤ Virtuozzo tools updates ├

Select the Virtuozzo tools update you wish to apply.
Marking a certain update will automatically have all the
previous updates also installed.

        ( ) Do not install any updates
        ( ) Virtuozzo tools update 4.0.0-245
        ( ) Virtuozzo tools update 4.0.0-246
        ( ) Virtuozzo tools update 4.0.0-247
        ( ) Virtuozzo tools update 4.0.0-248
        (*) Virtuozzo tools update 4.0.0-249


        Quit      Back      Select      Info
```

Notice that the bottommost update includes the functionality of all the other updates. You can select any of the intermediary updates and press Select to go back to the List of Selected Updates screen and read the information on this update. You will be able to perform customization more than once until you finally decide on the set of updates to be applied and press Next.

Downloading and installing the necessary updates is straightforward.

Updating EZ Templates

Updating EZ templates consists in updating one or more EZ templates configuration files located in the /vz/template/<*os_name*>/<*os_version*>/<*arch*>/config directory on the Parallels server and takes place if you have launched the vzup2date utility with the -z option. The first few steps of the wizard were described in the **Updating in Graphical Mode** subsection (p. 163). As soon as you press **Next** in the **Welcome...** window, the utility will try to connect to the EZ templates repository (either the Parallels default repository or your own one) and, if the connection is successful, display the **EZ Templates Selection** window listing all EZ templates that have one or more updates available or that are not installed on your server at all.

```
                 ┤ EZ templates selection ├

  Below is a list of OS distributions for which new OS and
  application EZ templates are available.  Mark those you
  wish to install or update and press Next.

                  [ ] Suse 10.0 i386 (EZ)
                  [ ] Suse 10.1 i386 (EZ)
                  [ ] Suse 10.2 i386 (EZ)
                  [ ] Suse 10.3 i386 (EZ)
                  [ ] Ubuntu 5.10 i386 (EZ)
                  [ ] Ubuntu 6.06 i386 (EZ)
                  [ ] Ubuntu 6.10 i386 (EZ)
                  [ ] Ubuntu 7.04 i386 (EZ)


          Quit        Back        Next        Customize
```
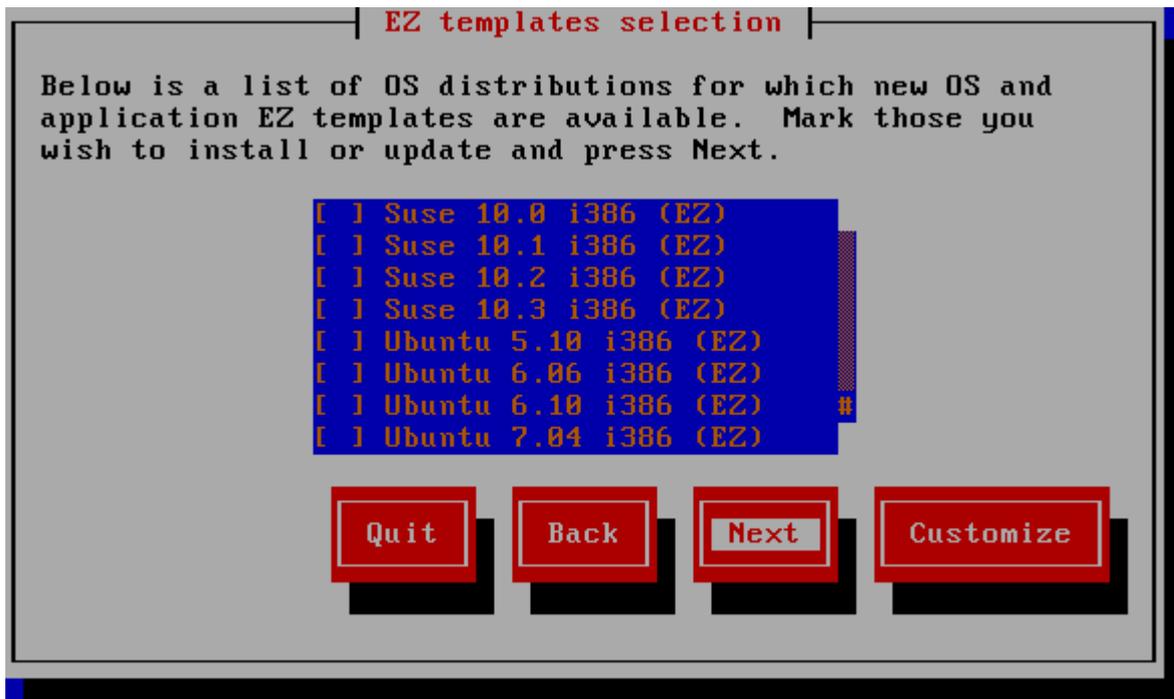
In this window, you can do one of the following:

- If you wish to download and install all available EZ templates/template updates for a certain Linux distribution, select this distribution by placing the cursor beside it and pressing the space bar on your keyboard; then click **Next**.

- If you wish only certain EZ templates of the corresponding Linux distribution to be installed/updated on the Parallels server, place the cursor beside this distribution and press F2 on your keyboard. You will be presented with the **Templates selection** window where you can select the corresponding EZ templates.

After choosing the right EZ templates, click the Select button to close the displayed window, and then click Next to proceed with the wizard.

Note: New application EZ templates for a Linux distribution can be installed on the Parallels server only if the corresponding OS EZ template is already installed on this server.

In the next step, you can review the EZ templates/template updates you selected on the previous step and scheduled for downloading and installing on your server. If you are not satisfied with the chosen templates/template updates, click the Back button to return to the previous step and modify the set of templates; otherwise, click Next to start downloading the templates/template updates.

After the EZ templates/templates have been successfully downloaded to the server, the Installing EZ template window is displayed.

```
┤ EZ templates download confirmation ├

  Below is a list of EZ templates to be downloaded.

Ubuntu 7.04 i386 (EZ)
    Ubuntu 7.04 x86 (4.0.0-6.swsoft)              7.9KB
    Cyrus-imap (4.0.0-2.swsoft)                   3.2KB
    Devel (4.0.0-1.swsoft)                        3.7KB
    Imp (4.0.0-1.swsoft)                          2.8KB   #
    Mailman (4.0.0-1.swsoft)                      2.8KB
    Mod_perl (4.0.0-1.swsoft)                     2.8KB
    Mysql (4.0.0-1.swsoft)                        3.2KB


                        Quit      Back      Next
```
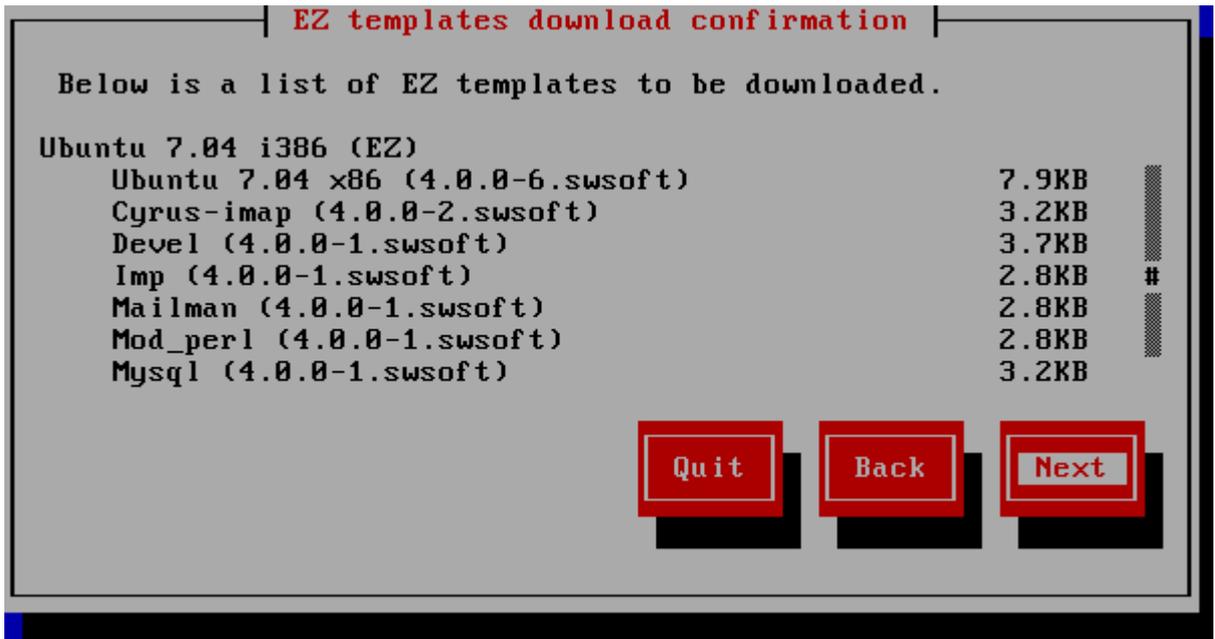
In this window, you can view the templates/template updates ready to be installed on your server. If you are installing a new OS EZ template/OS EZ template update, you can select the Run vzpkg cache after installation option and specify whether to cache the corresponding OS EZ template/template update right after its installation on the server or to do it at a later time. By default, all OS EZ templates are just installed on the Parallels without being cached. However, you can select the provided check box and schedule your OS EZ template/template update for caching. Clicking Next starts installing the EZ templates on the server. By the time the wizard finishes, you should have updated OS and application templates on your system.

Updating in Command Line Mode

Another way of updating your Parallels Server Bare Metal system files and templates is to run the `vzup2date` utility in the command line mode and to pass the corresponding commands, switches, and options to it. While executing `vzup2date` in the command line mode, you can choose between the batch and messages submodes. Both submodes can be used to update either the Parallels Server Bare Metal system files or EZ templates and have the identical syntax. However, the output produced by these commands is different. The messages submode output is less user friendly than that of the batch submode and is mostly suitable for machine processing.

To run the `vzup2date` utility in the command line mode, you can use either the `-m batch` switch or the `-m messages` switch intended for executing `vzup2date` in the batch and messages submodes, respectively.

Let us assume that you wish to update Parallels Server Bare Metal system files by installing the latest core in the batch submode. To do this, you can issue the following command on the Parallels server:

```
# vzup2date -m batch install --core
```

This will check the Parallels Server Bare Metal repository for the latest core updates and, in the case of finding any, download and install them on your server.

To update your Parallels Server Bare Metal installation, you may need to edit the `/etc/sysconfig/vzup2date/vzup2date.conf` file to specify the repository from where the updates are to be downloaded or configure a number of other parameters. Detailed information on the `vzup2date.conf` file is provided in the *Parallels Command Line Reference Guide*.

You can also execute the `vzup2date` utility in the batch mode to update the EZ templates installed on the Parallels server. For example, you can issue the following command

```
# vzup2date -t -m batch install --all-os
```

to update all OS templates installed on your server. Detailed information on all options that can be passed to the `vzup2date` utility is given in the *Parallels Command Line Reference Guide*.

Note: To perform the aforementioned operations in the messages submode, you should pass the `-m messages` option to the `vzup2date` utility instead of `-m batch`.

Updating Software In Virtual Machines

To keep software in your virtual machines up to date, you can use the same means you would use on standalone computers running the corresponding operating systems:

- In Linux-based virtual machines, you can use the native Linux updaters (`up2date`, `yum`, or `yast`).
- In Windows-based virtual machines, you can use the native Windows updaters (e.g. the Windows Update tool).

You should regularly run these updaters to ensure that your system has the latest updates and fixes (including security patches) installed. For more information on native updaters, refer to the documentation shipped with your operating system.

Updating Containers

Parallels Server Bare Metal provides two facilities to keep your Containers up to date. These facilities include:

- Updating EZ templates software packages inside a particular Container by means of the `vzpkg` utility. Using this facility, you can keep any of the Containers existing on your Parallels server up to date.
- Updating caches of the OS EZ templates installed on the Parallels server. This facility allows you to create new Containers already having the latest software packages installed.

Updating EZ Template Packages Inside a Container

Parallels Server Bare Metal allows you to update packages of the OS EZ template a Container is based on and of any application EZ templates applied to the Container. You can do it by using the `vzpkg update` utility. Assuming that Container 101 is based on the `redhat-el5-x86` OS EZ template, you can issue the following command to update all packages included in this template:

```
# vzpkg update 101 redhat-el5-x86
...
  Updating: httpd                  ##################### [1/4]
  Updating: vzdev                  ##################### [2/4]
  Cleanup : vzdev                  ##################### [3/4]
  Cleanup : httpd                  ##################### [4/4]

Updated: httpd.i386 0:2.0.54-10.2 vzdev.noarch 0:1.0-4.swsoft
Complete!
Updated:
 httpd                  i386       0:2.0.54-10.2
 vzdev                  noarch     0:1.0-4.swsoft
```

Notes:

1. Updating EZ templates is supported for running Containers only.

2. If you are going to update the cache of a commercial OS EZ template (e.g. Red Hat Enterprise Server 5 or SLES 10), you should first update software packages in the remote repository used to handle this OS EZ template and then proceed with updating the EZ template cache. Detailed information on how to manage repositories for commercial Linux distributions is provided in the *Parallels Server Bare Metal Templates Management Guide*.

As you can see from the example above, the `httpd` and `vzdev` applications have been updated for the `redhat-el5-x86` OS EZ template. If you wish to update all EZ templates (including the OS EZ template) inside Container 101 at once, execute this command:

```
# vzpkg update 101
...
Running Transaction
  Updating  : hwdata                  #################### [1/2]
  Cleanup   : hwdata                  #################### [2/2]

Updated: hwdata.noarch 0:1.0-3.swsoft
Complete!
Updated:
 hwdata                  noarch     0:0.158.1-1
```

In the example above, only the `hwdata` package inside Container 101 was out of date and updated to the latest version.

Updating OS EZ Template Caches

With the release of new updates for the corresponding Linux distribution, the created OS EZ template cache can become obsolete. Parallels Server Bare Metal allows you to quickly update your OS EZ template caches using the `vzpkg update cache` command.

Note: If you are going to update the cache of a commercial OS EZ template (e.g. Red Hat Enterprise Server 5 or SLES 10), you should first update software packages in the remote repository used to handle this OS EZ template and then proceed with updating the EZ template cache. Detailed information on how to manage repositories for commercial Linux distributions is provided in the *Parallels Command Line Reference Guide*.

When executed, `vzpkg update cache` checks the `cache` directory in the template area (by default, the template area is located in `/vz/template`) on the Parallels server and updates all existing tarballs in this directory. However, you can explicitly indicate the tarball for what OS EZ template should be updated by specifying the OS EZ template name. For example, to update the tarball for the `fedora-core-8-x86` OS EZ template, you should issue the following command:

```
# vzpkg update cache fedora-core-8-x86
Loading "rpm2vzrpm" plugin
Setting up Update Process
Setting up repositories
base0           100% |=========================|  951 B    00:00
base1           100% |=========================|  951 B    00:00
base2           100% |=========================|  951 B    00:00
base3           100% |=========================|  951 B    00:00
...
```

Upon the `vzpkg update cache` execution, the old tarball is renamed by receiving the `-old` suffix (e.g. `fedora-core-8-x86.tar.gz-old`):

```
# ls /vz/template/cache
fedora-core-8-x86.tar.gz  fedora-core-8-x86.tar.gz-old
```

You can also pass the `-f` option to `vzpkg update cache` to remove an existing tar archive and create a new one instead of it.

If the `vzpkg update cache` command does not find a tarball for one or several OS EZ templates installed on the server, it creates tar archives of the corresponding OS EZ templates and puts them to the `/vz/template/cache` directory.

Advanced Tasks

This chapter describes those tasks that are intended for advanced system administrators who would like to obtain deeper knowledge about Parallels Containers capabilities.

In This Chapter

Configuring Capabilities

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

This section documents the tasks that can be achieved using per-Container capabilities in Parallels Server Bare Metal and all configurable capabilities.

Creating VZFS Symlinks Inside a Container

Normally it is impossible to create a VZFS symlink from a Container. The ability to create VZFS symlinks presents a serious security concern explained further in this subsection. However, there may be a situation when you need such an ability, for example, for testing created templates or creating VZFS mounts.

A VZFS symlink is a symbolic link starting with four slashes. You can see VZFS symlinks in the private area of any Container, as is illustrated below:

```
# ls -l /vz/private/101/root/bin/bash
lrwxr-xr-x   1 root   root   37 Jul  9  2008 \
/vz/private/101/root/bin/bash -> \
////redhat-as4/bash-3.0-19.2/bin/bash
```

VZFS symlinks have no special meaning if the private area is not mounted over VZFS (to the Container root directory). If it is, then instead of a VZFS symlink the users inside the Container will see the file located in the template directory (in this particular case, /vz/template/redhat-as4/bash-3.0-19.2/bin/bash) instead of the VZFS symlink.

If you try to create a VZFS symlink inside the Container, you will get an error:

```
[root@ct101 root]# ln -s ////redhat-as4/bash-3.0-19-2/bin/bash .
ln: creating symbolic link `./bash' to \
`////redhat-as4/bash-3.0-19.2/bin/bash': Invalid argument
```

The reason for this restriction is security considerations. If an intruder can correctly guess where the template area (defined by the TEMPLATE variable in the global configuration file /etc/sysconfig/vz) is located, he/she can access any file on the server provided the path to the file is guessed correctly. However, in case it is necessary to allow the VZFS symlinks creation inside a Container, it is possible to make use of the sys_rawio capability:

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

After restarting the Container, you can unpack VZRPMs inside the Container or simply create VZFS symlinks:

```
# ssh root@ct101
root@ct101's password:
Last login: Mon Oct 28 23:25:58 2008 from 10.100.40.18
[root@ct101 root]# rpm2cpio bash-3.0-19.2.i386.vz.rpm | cpio -id
94 blocks
[root@ct101 root]# ls -l bin/bash
-rwxr-xr-x    1 root     root       519964 Oct 29 23:35 bin/bash
[root@ct101 root]# ln -s ////redhat-as4/bash-3.0-19.2/bin/bash .
[root@ct101 root]# ls -l bash
-rwxrwxrwx    1 root     root       519964 Oct 29 23:35 bash
```

As you can see both VZFS symlinks look like regular files for Container users. If you need to unpack and work on symlinks themselves, you have to create a Container that has a directory bind-mounted over a regular file system such as EXT2FS, EXT3FS or ReiserFS.

Remember that assigning this capability to non-trusted Containers can lead to compromising the server. The session below shows how a malicious Container administrator can get a copy of the server password database files:

```
[root@ct101 root]# ln -s ////../../etc/passwd .
[root@ct101 root]# ln -s ////../../etc/shadow .
```

```
[root@ct101 root]# ls -l
total 3
-rwxrwxrwx    1 root      root           1252 Oct 29 23:56 passwd
-rwxrwxrwx    1 root      root            823 Oct 29 23:56 shadow
```

While there is no easy way to substitute the password files on the server, a malicious Container administrator could run a dictionary attack against the obtained files.

Available Capabilities for Container

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

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

Capabilities Defined by POSIX Draft

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

Linux-Specific Capabilities

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

Creating Customized Containers

If you wish to run one or several customized applications inside your Containers and the number of such Containers is relatively large, you may think of a way to automate the process of creating Containers that already have a number of applications installed and tuned to meet your demands. So, you do not need to manually install and customize your applications every time you create a new Container.

Parallels Server Bare Metal allows you to create customized Containers having a certain set of customized applications installed inside them right after their creation in one of the following ways:

- By making a customized base OS EZ template and using it as the basis for your Containers.
- By making a non-base OS EZ template and using it as the basis for your Containers.
- By making a customized application EZ template, adding it to a new configuration sample file, and using this sample file as the basis for your Containers.

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

Using Customized OS EZ Template

Let us first start with making a customized base OS EZ template which can then be used to create Containers with a set of application already tuned to meet your demands. To make such a template, you should perform the following operations:

1 Create a metafile that will serve as the basis for your customized base OS EZ template.

> **Notes:**
>
> 1. Detailed information on how to create metafiles is given in the *Parallels Server 4 Bare MetalTemplates Management Guide*.
>
> 2. While creating a metafile for your new OS EZ template, you should make sure that the value of either the %osname parameter or the %version parameter in the metafile differs from the names or versions of all base OS EZ templates installed on the server. So, if the base RHEL4 OS EZ template is already installed on your server, these values cannot be simultaneously set to redhat and as4.

2 Create one or more scripts that will be executed on different stages of the OS EZ template lifecycle and customize your applications to meet your needs. For example, you can create a postinstall script with the name of post_install.bash and make it perform a number of customization operations on some application included in the OS EZ template after installing this application inside your Container.

3 Create a customized OS EZ template by running the vzmktmpl utility and passing the corresponding options to it. So, you can use the --post-install option and specify the path to the post_install.bash script from the example above to make an OS EZ template that will customize your application after installing it inside your Container.

> **Note:** The full list of options allowing you to specify what scripts are to be executed on what stage of the EZ template lifecycle is provided in the **vzmktmpl** subsection of the *Parallels Command Line Reference Guide*.

4 Install the customized OS EZ template on the server by running the rpm -i command.

5 Cache the created OS EZ template by running the vzpkg create cache command. Detailed information on how you can do it is provided in the *Parallels Server 4 Bare Metal Templates Management Guide*.

6 Create a Container based on the OS EZ template.

For example, to create a Container which will run Red Hat Enterprise Linux 4 (RHEL 4) and have the customized mysql and apache applications installed inside it right after its creation, you can do the following:

1 Create a metafile for the RHEL 4 OS EZ template, name it, for example, rhel_4_customized.metafile, and save in the /root/rhel4 directory on the server.

2 Make a script that will perform a number of custom operations after applying the mysql and apache application EZ templates to the Container, and name it post_install.bash.

3 Copy the script to the /root/rhel4 directory on the server.

4 Execute the following command on the server to create the RHEL 4 OS EZ template:

```
# vzmktmpl /root/rhel4/rhel_4_customized.metafile \
```

```
            --post-install /root/rhel4/post_install.bash
```

This command will create an OS EZ template for RHEL 4 and put it to the `/root` directory (e.g. `/root/redhat_customized-as4-x86-ez-4.0.0-1.swsoft.noarch.rpm`).

5   Install the resulting OS EZ template on the server:

```
# rpm -i /root/redhat_customized-as4-x86-ez-4.0.0-1.swsoft.noarch.rpm
```

6   Cache the installed OS EZ template:

```
# vzpkg create cache redhat_customized-as-x86
...
Complete!
Packing cache file redhat_customized-as4-x86.tar.gz ...
Cache file redhat_customized-as4-x86.tar.gz [14M] created.
```

7   Create Container 101 on the basis of the new OS EZ template:

```
# pctl create 101 --ostemplate redhat_customized-as4-x86
                   --config basic
Creating Container private area (redhat_customized-as4-x86)
Container is mounted
Postcreate action done
Container is unmounted
Container private area was created
Delete port redirection
Adding port redirection to Container(1): 4643 8443
```

So, you have just created Container 101 having the customized `mysql` and `apache` applications installed inside it.

Using EZ OS Template Set

Another way of creating customized Containers is to make a non-base OS EZ template (also known as an OS EZ template set) differing from the corresponding base OS EZ template in the number of packages included in this template. For example, if you wish your Container to run Red Hat Enterprise Linux 4 and to function as a Linux-based server only, you can create the `redhat-as4-x86-server` OS EZ template set and include only those packages in it that are needed for performing main server tasks. So, you can specify packages to be used for setting up file and print sharing and exclude all the packages for graphical interfaces (GNOME and KDE).

To create a non-base OS EZ template, you should complete the following tasks:

1 Create a metafile that will serve as the basis for your non-base OS EZ template. Any metafile for this kind of EZ template must contain the following information:

- `%osname`: the name of the Linux distribution for which you are creating the OS EZ template set. This name must correspond to that specified in the base OS EZ template. For example, if you are creating an OS template set of the base OS EZ template for RHEL 4, you must set the value of this parameter to `redhat`.

- `%osver`: the version of the Linux distribution specified as the value of the `%osname` parameter. This name must correspond to that specified in the base OS EZ template. For example, if you are creating an OS template set of the base OS EZ template for RHEL 4, you must set the value of this parameter to `as4`.

- `%osarch`: the system architecture where the EZ template is to be run. This name must correspond to that specified in the base OS EZ template. For example, if you are creating an OS template set of the base OS EZ template for RHEL 4, you must set the value of this parameter to `x86`.

- `%setname`: the name to be assigned to your non-base OS EZ template. You can specify any name you like for your OS template set:

  a This name will be added to the name of the base OS EZ template after the indication of the architecture where the OS EZ template is to be run. For example, if you are creating an OS template set of the base OS EZ template for RHEL 4 which is supposed to run on x86 platforms, the name of your non-base OS EZ template should look like the following - `redhat-as4-x86-`*Template_Name*`-ez-1.0-1.noarch.rpm` - where *Template_Name* is the name you specify as the value of the `%setname` parameter.

  b This name will also be assigned to the directory which will store the meta data of your non-base OS EZ template after the template installation on the server. For example, it will have the name of `/vz/template/redhat/as4/x86/config/os/my_non_base_template/` after you set the value of this parameter to `my_non_base_template`, created a non-base OS EZ template for RHEL 4, and installed it on the server.

- `%packages`: a list of RPM packages to be included in the non-base OS EZ template. This parameter allows you to specify what applications will be present inside your Containers based on this OS EZ template set right after their installation. The names of the packages listed as the value of this parameter must correspond to the names of real RPM packages (without indicating the package version, release, architecture, and the `.rpm` extension) that are stored in the repository used for managing your EZ templates.

Note: You can also specify a number of additional parameters in your metafile. For example, you may wish to add one or several extra packages to your OS EZ template set which are not available in the repository used to handle the packages for the corresponding base OS EZ template. For this purpose, you will have to specify the `%mirrorlist` parameter providing information on the repository where these extra packages are kept. Detailed information on all parameters you can set in metafiles is given in the *Parallels Command Line Reference Guide*.

2   You can also (although you do not have to) create a number of scripts that will be executed on different stages of the non-base OS EZ template lifecycle and customize your applications to meet your demands. The path to these scripts should then be specified after the corresponding options while creating your OS template set. For example, you can create a preinstall script with the name of `pre_install.bash` and make it perform a number of customization operations on some application included in the non-base OS EZ template before installing this application in your Container.

Note: If there are no scripts for a non-base OS EZ template, the scripts available for the corresponding base OS EZ template will be executed.

3   Create the non-base OS EZ template by running the `vzmktmpl` utility and passing the corresponding options to it, if needed. So, if you created one or several scripts in the previous step, you can use special options and specify the path to these scripts during the command execution. For example, you can use the `--pre-install` option and specify the path to the `pre_install.bash` script to make an OS EZ template that will customize your application before installing it inside your Container.

Note: The full list of options allowing you to specify what scripts are to be executed on what stage of the EZ template lifecycle is provided in the **vzmktmpl** subsection of the *Parallels Command Line Reference Guide*.

4   Install the non-base OS EZ template on the server using the `rpm -i` command.

5   Cache the created OS EZ template by running the `vzpkg create cache` command. Detailed information on how you can do it is provided in the *Parallels Server 4 Bare Metal Templates Management Guide*.

6   Create a Container based on the OS EZ template.

Using Customized Application Template

If the number of customized applications inside your Containers is relatively small, you can also use the following way of creating customized Containers:

1 Create a metafile that will serve as the basis for your customized application EZ template.

> **Note:** Detailed information on how to create metafile is given in the **Creating Metafile for EZ Template** subsection of the *Parallels Server 4 Bare Metal Templates Management Guide*.

2 Create one or more scripts that will be executed on different stages of the application EZ template lifecycle and customize your applications to meet your demands. For example, you can create a postinstall script with the name of `post_install.bash` and make it perform a number of customization operations on your application after installing this application in your Container.

3 Create a customized application EZ template by running the `vzmktmpl` utility and passing the corresponding options to it. So, you can use the `--post-install` option and specify the path to the `post_install.bash` script from the example above to customize your application in accordance with your needs after installing it in your Container.

> **Note:** The full list of options allowing you to specify what scripts are to be executed on what stage of the EZ template lifecycle is provided in the **vzmktmpl** subsection of the *Parallels Command Line Reference Guide*.

4 Install the customized EZ template on the server using the `rpm -i` command.

5 Create a new Container configuration sample file and include the customized EZ template in this file. Detailed information on Container configuration sample files is provided in the **Managing Container Resources Configuration** section (p. 118).

6 Create a customized Container on the basis of the configuration sample.

The following example demonstrates how to create Container 101 which will run Red Hat Enterprise Linux 4 and have the customized `mysql` application installed inside it right after its creation:

1 Create a metafile for the `mysql` application, name it `mysql.metafile`, and save in the `/usr/mysql` directory on the server.

2 Make a script that will perform a number of custom operations after applying the `mysql` EZ template to the Container, and name it `post_install.bash`.

3 Copy the script to the `/usr/mysql` directory on the server.

4 Execute the following command on the server to create the `mysql` EZ template:

```
# vzmktmpl /usr/mysql/mysql.metafile \
           --post-install /usr/mysql/post_install.bash
```

This command will create an EZ template for the `mysql` application and put it to the `/root` directory (e.g. `/root/mysql-redhat-as4-x86-ez-4.0.0-1.swsoft.noarch.rpm`).

5 Install the `mysql` EZ template on the server:

```
# rpm -ihv /root/mysql-redhat-as4-x86-ez-4.0.0-1.swsoft.noarch.rpm
```

6 Create a new Container configuration sample file and add the `mysql` EZ template to a list of templates that will be installed in Containers created on the basis of this configuration sample file.

7 Create Container 101 by using the `pctl create` command and the `mysql` sample file:

```
# pctl create 101 --ostemplate redhat-as4-x86 --config mysql
Creating Container private area (redhat-as4-x86)
Container is mounted
Postcreate action done
Container is unmounted
Container private area was created
Delete port redirection
Adding port redirection to Container(1): 4643 8443
```

So, you have just created Container 101 having the customized `mysql` application installed inside it.

Changing System Time From Container

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

Normally only the server 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 server itself, since flaws in the software can lead to compromising all Containers on this server. Thus, if you plan to use NTP, you should create a special Container for it and configure it to have the sys_time capability. The example below illustrates configuring such a Container:

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

The output of the above command warns you that pctl cannot apply changes in the capabilities to a running Container. The Container has to be restarted before changes take effect:

```
# pctl stop 101; pctl start 101
Stopping Container ...
Container was stopped
Container is unmounted
Starting Container ...
Container is mounted
Adding IP address(es): 192.168.1.101
Hostname for Container set: Container101
Container start in progress...
# ssh root@ct101
root@ct101's password:
Last login: Mon Feb 28 23:25:58 2007 from 10.100.40.18
[root@ct101 root]# date
Mon Feb 28 23:31:57 EST 2007
[root@ct101 root]# date 10291300
Tue Feb 29 13:00:00 EST 2007
[root@ct101 root]# date
Tue Feb 29 13:00:02 EST 2007
[root@ct101 root]# logout
Connection to Container101 closed.
# date
Tue Feb 29 13:01:31 EST 2007
```

The command session above shows the way to change the system time from Container 101. The changes will affect all the Containers and the server itself. It is not advisable to have more than one Container 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).

Obtaining Server ID From Inside a Container

The default Parallels Server Bare Metal installation does not allow users inside a Container to obtain any information specific to the Parallels server the Container is running on. The reason is that no Container shall have knowledge about the corresponding server. A Container can be transparently migrated to another server, and if this Container runs any applications depending on the particular server, these applications might fail after the migration.

In some situations, however, you need to provide a unique server ID to some applications. For example, you might want to license your application per server. In this case, after the migration your customer will need to re-apply the license for your application.

Parallels Server Bare Metal provides access to the unique server ID via the `/proc/vz/hwid` file. The default Parallels Server Bare Metal installation makes this file accessible to Containers from 1 to 100 (i.e. Containers with reserved IDs). It is possible to change this range in the global configuration file (`vz.conf`). For example, this is the way to make the file visible in Containers from 1 to 1000:

```
# vi /etc/vz/vz.conf
VZPRIVRANGE="1 1000"
# pctl exec 101 cat /proc/vz/hwid
0C3A.14CB.391B.6B69.02C9.4022.3E2F.CAF6
```

The above example illustrates accessing the server ID from Container 101.

Enabling VPN for Container

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

- Make sure the `tun.o` module is already loaded before Parallels Server Bare Metal is started:

```
# lsmod
```

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

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

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

```
# pctl exec 101 mkdir -p /dev/net
# pctl exec 101 mknod /dev/net/tun c 10 200
# pctl 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/>`.

Managing Server Resources Parameters

Parallels Server Bare Metal allows you to configure a number of resource management parameters defining the amount of resources to be allocated to the Parallels server. These parameters include all standard UBC parameters (VMGUARPAGES, KMEMSIZE, OOMGUARPAGES, etc.) as well as the ONBOOT parameter.

You can edit any of these parameters in the /etc/vz/conf/0.conf file on the server using your favorite text editor (for example, vi or emacs) or using the pctl set command and specifying 0 after this command. For example:

```
# pctl set 0 --kmemsize 12211840:14359296 --save
Saved parameters for Container 0
```

This command sets both the barrier and limit values of unswappable kernel memory (in bytes) which can be allocated to internal kernel structures of the processes on the server. The specified parameter values will be in force until the server restart. If you wish these values to be applied to the server on its next booting, you should additionally set the ONBOOT parameter in the /etc/vz/conf/0.conf file to yes. This can be done in one of the following ways:

- Passing the --onboot option to the pctl set command:

```
# pctl set 0 --onboot yes
Saved parameters for Container 0
```

- Editing the /etc/vz/conf/0.conf file with your favorite text editor (e.g. vi) and setting the value of the ONBOOT parameter in this file to yes.

Note: Detailed information on all resource parameters that can be changed in respect of your Parallels server is provided in the *Parallels Command Line Reference Guide*.

If you have made a number of changes to server resource management parameters and wish to reset them to the values specified in the /etc/vz/conf/0.conf file, you can run this command:

```
# pctl set 0 --reset_ub
UBC limits were set successfully
```

Setting Immutable and Append Flags for Container Files and Directories

You can use standard Linux utilities - `chattr` and `lsattr` - to set extra flags for files and directories inside your Containers and to query their status, respectively. Currently, two of these extra flags - 'append' and 'immutable' - are supported. For example, you can execute the following command to set the 'immutable' flag for the `/root/MyFile` file inside Container 101:

```
[root@ct101 root] chattr +i /root/MyFile
```

To check that the 'immutable' flag has been successfully set, use the following command:

```
[root@ct101 root] lsattr /root/MyFile
----i-------- /root/MyFile
```

Note: For detailed information on the `chattr` and `lsattr` utilities, see their manual pages.

Customizing /proc/meminfo Output Inside Container

The /proc/meminfo virtual file allows you to view the information about memory usage (both physical and swap) on the system. In the current version of Parallels Server Bare Metal, you can customize the output of this file inside a particular Container and set it to one of the following modes:

- *Non-virtualized*. In this case running the cat /proc/meminfo command inside a Container will display the information about the physical memory on the server (total, used, free, shared, etc.), in kilobytes.

- *Virtualized in pages*. Setting the /proc/meminfo output to this mode allows you to specify what amount of total memory (in kilobytes) will be displayed while running the cat /proc/meminfo command inside this or that Container.

- *Virtualized in privvmpages*. Setting the /proc/meminfo output to this mode also allows you to arbitrarily specify the amount of total memory (in kilobytes) to be displayed while running the cat /proc/meminfo command inside this or that Container. As distinct from the previous mode, the amount of memory to be shown in this mode is calculated on the basis of the value of the PRIVVMPAGES parameter set in the Container configuration file.

Notes:

1. Enabling this or that mode for a Container does not exert any influence on the real resources allocation to the Container. It is only used to modify the way the /proc/meminfo output will look inside this Container.

2. The output of the /proc/meminfo file cannot be customized if the new SLM functionality is enabled on the Parallels server. In this case, the cat /proc/meminfo command executed inside a Container always displays the amount of memory set for this Container using the --slmmemorylimit option of the pctl set command.

During the Parallels Server Bare Metal installation, the output of the /proc/meminfo virtual file is set to the *'non-virtualized'* mode, i.e. running the cat /proc/meminfo command inside any Container will show the information about the memory usage on the Parallels server. You can use the --meminfo option with the pctl set command to switch between different modes:

- To set the output of /proc/meminfo inside Container 101 to the *'virtualized in pages'* mode, issue the following command:

```
# pctl set 101 --meminfo pages:2000 --save
```

  The amount of memory that will be displayed by running the cat /proc/meminfo command inside Container 101 is defined by the data specified after the --meminfo option:

  - pages tells the pctl set command that you wish to enable the *'virtualized in pages'* mode for the /proc/meminfo output and simultaneously denotes the units of measurement to be used for setting the amount of memory (e.g. 4-Kb pages for Containers running 32-bit operating systems).

- 200 denotes the number of pages to be shown in the /proc/meminfo output.

In our case the /proc/meminfo output inside Container 101 may look like the following:

```
# pctl exec 101 cat /proc/meminfo
MemTotal:             8000 kB
MemFree:              5140 kB
LowTotal:             8000 kB
LowFree:           5140 kB
Buffers:                 0 kB
Cached:                  0 kB
SwapCached:              0 kB
HighTotal:               0 kB
HighFree:                0 kB
...
```

When working in this mode, keep in mind the following:

- The specified amount of memory (in our case it is 8000 Kb) is always shown in the MemTotal and LowTotal fields of the cat /proc/meminfo output.

- The values in the MemFree and LowFree fields are calculated automatically by the system.

- All the other fields in the command output have the values set to 0.

- To set the output of /proc/meminfo inside Container 101 to the '*virtualized in privvmpages*' mode, execute the following command:

```
# pctl set 101 --meminfo privvmpages:3 --save
```

The amount of memory that will be displayed by running the cat /proc/meminfo command inside Container 101 is calculated using the following formulas:

- *Privvmpages_Value* * 3 * 4Kb if Container 101 is running a 32-bit operating system (OS) or an OS for x86-64 processors and

- *Privvmpages_Value* * 3 * 16Kb if Container 101 is running an OS for IA-64 processors

where *Privvmpages_Value* denotes the value of the PRIVVMPAGES parameter set in the Container configuration file and 3 is an arbitrary integer coefficient which you can modify to increase/decrease the amount of memory in the /proc/meminfo output. Assuming that the privvmpages parameter for Container 101 is set to 10000, your output may look as follows:

```
# pctl exec 101 cat /proc/meminfo
MemTotal:            120000 kB
MemFree:             78248 kB
LowTotal:            120000 kB
LowFree:             78248 kB
Buffers:             0 kB
Cached:                      0 kB
SwapCached:                  0 kB
HighTotal:                   0 kB
HighFree:                    0 kB
...
```

As can be seen from the example above, the displayed records comply with the same rules as the records in the '*virtualized in pages*' mode.

- To revert the output of /proc/meminfo to the default mode, execute the following command:

```
# pctl set 101 --meminfo none --save
```

Note: If the value specified after the `--meminfo` option exceeds the total amount of memory available on the Parallels server, the `cat /proc/meminfo` command executed inside a Container will display the information about the total physical memory on this server.

The `--save` flag in the commands above saves all the parameters to the Container configuration file. If you do not want the applied changes to persist, you can omit the `--save` option and the applied changes will be valid only till the Container shutdown.

Loading iptables Modules

The given section provides information on how you can manage `iptables` modules on the server and inside particular Containers.

Loading iptables Modules to Parallels Server

You can configure a list of `iptables` modules that will be loaded on the Parallels server after its startup by editing the `/etc/vz/vz.conf` file. The `IPTABLES` parameter in this file determines the `iptables` modules that will additionally be loaded to the server during the Parallels Server Bare Metal startup. For example, you can indicate the following `iptables` modules as the value of this parameter to have them automatically loaded to your Parallels server after the Parallels Server Bare Metal startup:

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

All the specified modules will be loaded on the server startup once you restart it.

Loading iptables Modules to Particular Containers

The list of `iptables` modules that are loaded to a Container by default is determined by the `iptables` modules loaded on the server at the moment of the Container startup. For example, if your server has the `ipt_REJECT`, `ipt_tos`, `ipt_limit`, `ipt_multiport`, and `iptable_filter` modules loaded, any Containers on this server will also have these `iptables` modules loaded after their startup.

However, Parallels Server Bare Metal allows you to prevent certain modules from being loaded inside a Container on its startup, even if they are loaded on the server itself. The full list of such `iptables` modules is listed below:

- `ip_table`
- `ip6_table`
- `iptable_filter`
- `ip6table_filter`
- `iptable_mangle`
- `ip6table_mangle`
- `ip_conntrack`
- `ip_conntrack_ftp`
- `ip_conntrack_irc`
- `iptable_nat`
- `ip_nat_ftp`
- `ip_nat_irc`

To forbid the usage of any of the aforementioned `iptables` modules inside a Container, you should explicitly indicate the names of the modules you wish to be loaded to the Container as the value of the `IPTABLES` parameter in the Container configuration file (`/etc/vz/conf/<CT_ID>.conf`) or by using the `pctl` command. For example:

```
# pctl set 101 --iptables ip_table --iptables iptable_filter --iptables
ip_conntrack --iptables iptable_nat --iptables iptable_mangle --save
```

This command will tell Parallels Server Bare Metal to:

- load the `ip_table`, `iptable_filter`, `ip_conntrack`, `iptable_nat`, and `iptable_mangle` modules to Container 101 if they are loaded on the server during the Container startup
- forbid the usage of all the other `iptables` modules listed above (i.e. `ip6_table`, `ip6table_filter`, `ip6table_mangle`, `ip_conntrack_ftp`, `ip_conntrack_irc`, `ip_nat_ftp`, `ip_nat_irc`) inside Container 101 even if they are loaded on the server during the Container startup

This information will also be saved in the Container configuration file thanks to the `--save` option.

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

Creating Configuration Files for New Linux Distributions

Distribution configuration files are used to distinguish among Containers running different Linux versions and to determine what scripts should be executed when performing the relevant Container-related operations (e.g. assigning a new IP address to the Container). Detailed information on distributions configurations files is provided in the **Linux Distribution Configuration Files** subsection of the *Parallels Command Line Reference Guide*.

All Linux distributions shipped with Parallels Server Bare Metal have their own configuration files located in the `/etc/vz/conf/dists` directory on the Parallels server. However, you may wish to create your own distribution configuration files to support new Linux versions released. Let us assume that you wish your Containers to run the CentOS 5 Linux distribution and, therefore, have to make the `centos-5.conf` distribution configuration file to define what scripts are to be executed while performing major tasks with Containers running this Linux version. To do this:

1 In the Container configuration file (with the name of `/etc/vz/conf/`*CT_ID*`.conf`), specify `centos-5` as the value of the `DISTRIBUTION` variable (for example, `DISTRIBUTION="centos-5"`).

2 Create the `centos-5.conf` configuration file in the `/etc/vz/conf/dists` directory. The easiest way to do it is copy one of the existing configuration files by executing the following command in the `/etc/vz/conf/dists` directory:

```
# cp fedora.conf centos-5.config
```

In the example above, we assume that the `fedora.conf` file is present in the `/etc/vz/conf/dists` directory on the Parallels server. In case it is not, you may use any other distribution configuration file available on your server.

3 Open the `centos.conf` file for editing with the help of any text editor:

```
# vi centos-5.conf
```

4 In the `centos-5.conf` file, go to the first entry and, in the right part of the entry, specify the name of the script you wish to be run on issuing the `pctl` command with the parameter specified in the left part of the entry. For example, if you wish the script to be executed while assigning a new IP address to your Container and the script has the `my_centos_script` name, your entry should look as follows:

```
ADD_IP=my_centos_script-add_ip.sh
```

Note: The information on all acceptable parameters and their description are provided in the *Parallels Command Line Reference Guide*.

5 Repeat **Step 4** for all entries in the file.

6 Place the scripts for the new Linux distribution to the `/etc/vz/conf/dists/scripts` directory on the Node. Make sure the names of these scripts coincide with those specified in the `centos-5.conf` file.

C H A P T E R   1 0

Troubleshooting

This chapter provides the information about those problems that may occur during your work with Parallels Server Bare Metal and suggests the ways to solve them, including getting technical support from Parallels.

In This Chapter

General Considerations

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

- Make sure a valid license is always loaded on the server. If your license has expired and the grace period is over, all the virtual machines and Containers on your server will be stopped.

- You should always remember where you are currently located in your terminal. Check it periodically using the `pwd`, `hostname`, `ifconfig`, `cat /proc/vz/veinfo` commands. One and the same command executed inside a virtual machine and Container and on the server can lead to very different results. You can also set up the `PS1` environment variable to show the full path in the `bash` prompt. To do this, add these lines to `/root/.bash_profile`:

```
PS1="[\u@\h \w]$ "
export PS1
```

- If the server slows down, use `vmstat`, `ps` (`ps axfw`), `dmesg`, `top` (`vztop`) to find out what is happening, never reboot the machine without investigation. If no thinking helps restore the normal operation, use the Alt+SysRq sequences to dump the memory (`showMem`) and processes (`showPc`).

- If the server was incorrectly brought down, on its next startup all the partitions will be checked and quota recalculated for each Container, which dramatically increases the startup time.

- Do not run any binary or script that belongs to a Container directly from the server, for example, do not ever do that:

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

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

- Do not use init scripts on the server. An init script may use `killall` to stop a service, which means that all similar processes will be killed in all Containers. You can check `/var/run/Service.pid` and kill the correspondent process explicitly.

- You must be able to detect any rootkit inside a Container. It is recommended to use the `chkrootkit` package for detection (you can download the latest version from www.chkrootkit.org), or at least run

```
rpm -Va|grep "S.5"
```

to check up if the MD5 sum has changed for any RPM file.

You can also run `nmap`, for example:

```
# nmap -p 1-65535 192.168.0.1

Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.0.1):
(The 65531 ports scanned but not shown below are in
  state: closed)
Port       State        Service
21/tcp     open         ftp
22/tcp     open         ssh
80/tcp     open         http
111/tcp    open         sunrpc
```

```
Nmap run completed -- 1 IP address (1 host up) scanned
  in 169 seconds
```

to check if any ports are open that should normally be closed.

That could however be a problem to remove a rootkit from a Container and make sure it is 100% removed. If you're not sure, create a new Container for that customer and migrate his/her sites and mail there.

- Check the `/var/log/` directory on the server to find out what is happening on the system. There are a number of log files that are maintained by the system and Parallels Server Bare Metal (the `boot.log`, `messages`, etc.), but other services and programs may also put their own log files here depending on your distribution of Linux and the services and applications that you are running. For example, there may be logs associated with running a mail server (the `maillog` file), automatic tasks (the `cron` file), and others. However, the first place to look into when you are troubleshooting is the `/var/log/messages` log file. It contains the boot messages when the system came up as well as other status messages as the system runs. Errors with I/O, networking, and other general system errors are reported in this file. So, we recommend that you read to the `messages` log file first and then proceed with the other files from the `/var/log/` directory.

- Subscribe to bug tracking lists. You should keep track of new public DoS tools or remote exploits for the software and install them into Containers or at servers.

- When using `iptables`, there is a simple rule for Chains usage to help protect both the server and its Containers:
  - use INPUT, OUTPUT to filter packets that come in/out the server
  - use FORWARD to filter packets that are designated for Containers

Kernel Troubleshooting

Using ALT+SYSRQ Keyboard Sequences

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

```
SysRq: unRaw Boot Sync Unmount showPc showTasks showMem loglevel0-8 tErm kIll
killalL Calls Oops
```

This output shows you what ALT+SYSRQ sequences you may use for performing this or that command. The capital letters in the command names identify the sequence. Thus, if there are any troubles with the machine and you're about to reboot it, please 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 filesystems

ALT+SYSRQ+L to kill all processes

ALT+SYSRQ+U try to unmount once again

ALT+SYSRQ+B to reboot

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

Saving Kernel Fault (OOPS)

You can use the following command to check for the kernel messages that should be reported to Parallels Server Bare Metal developers:

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

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

```
Sep 26 11:41:12 kernel: Linux version 2.6.18-8.1.1.el5.028stab043.1
(root@rhel5-32-build) (gcc version 4.1.1 20061011 (Red Hat 4.1.1-30)) #1 SMP
Wed Aug 29 11:51:58 MSK 2007
```

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

```
Aug 25 08:27:46 boar BUG: unable to handle kernel NULL pointer dereference at
virtual address 00000038
Aug 25 08:27:46 boar printing eip:
Aug 25 08:27:46 boar f0ce6507
Aug 25 08:27:46 boar *pde = 00003001
Aug 25 08:27:46 boar Oops: 0000 [#1]
Aug 25 08:27:46 boar SMP
Aug 25 08:27:46 boar last sysfs file:
Aug 25 08:27:46 boar Modules linked in: snapapi26(U) bridge(U) slm_dmprst(U)
ip_vzredir(U) vzredir(U) vzcompat(U) vzrst(U) i
p_nat(U) vzcpt(U) ip_conntrack(U) nfnetlink(U) vzfs(U) vzlinkdev(U)
vzethdev(U) vzevent(U) vzlist(U) vznet(U) vzstat(U) vzmo
n(U) xt_tcpudp(U) ip_vznetstat(U) vznetstat(U) iptable_mangle(U)
iptable_filter(U) ip_tables(U) slm_kill(U) slm_nofork(U) slm_core(U)
slm_skill(U) slm_if(U) vztable(U) vzdquota(U) vzdev(U) autofs4(U) hidp(U)
rfcomm(U) l2cap(U) bluetooth(U) sunrpc(U) ipv6(U) xt_length(U) ipt_ttl(U)
xt_tcpmss(U) ipt_TCPMSS(U) xt_multiport(U) xt_limit(U) ipt_tos(U)
ipt_REJECT(U) x_tables(U) video(U) sbs(U) i2c_ec(U) button(U) battery(U)
asus_acpi(U) ac(U) lp(U) floppy(U) sg(U) pcspkr(U) i2c_piix4(U) e100(U)
parport_pc(U) i2c_core(U) parport(U) cpqphp(U) eepro100(U) mii(U) serio_raw(U)
ide_cd(U) cdrom(U) ahci(U) libata(U) dm_snapshot
(U) dm_zero(U) dm_mirror(U) dm_mod(U) megaraid(U) sym53c8xx(U)
scsi_transport_spi(U) sd_mod(U) scsi_mod(U) ext3(U) jbd(U) ehci_hcd(U)
ohci_hcd(U) uhci_hcd(U)
Aug 25 08:27:46 boar CPU: 1, VCPU: -1.1
Aug 25 08:27:46 boar EIP: 0060:[<f0ce6507>] Tainted: P VLI
Aug 25 08:27:46 boar EFLAGS: 00010246 (2.6.18-028stab043.1-ent #1)
Aug 25 08:27:46 boar EIP is at clone_endio+0x29/0xc6 [dm_mod]
Aug 25 08:27:46 boar eax: 00000010   ebx: 00000001   ecx: 00000000   edx:
00000000
Aug 25 08:27:46 boar esi: 00000000   edi: b6f52920   ebp: c1a8dbc0   esp:
0b483e38
Aug 25 08:27:46 boar ds: 007b   es: 007b   ss: 0068
Aug 25 08:27:46 boar Process swapper (pid: 0, veid: 0, ti=0b482000
task=05e3f2b0 task.ti=0b482000)
Aug 25 08:27:46 boar Stack: 0b52caa0 00000001 00000000 b6f52920
00000000f0ce64de 00000000 02478825
Aug 25 08:27:46 boar 00000000 c18a8620 b6f52920 271e1a8c 024ca03800000000
00000000 00000000
Aug 25 08:27:46 boar 00000000 00000000 c18a3c00 00000202 c189e89400000006
00000000 05cb7200
Aug 25 08:27:46 boar Call Trace:
Aug 25 08:27:46 boar [<f0ce64de>] clone_endio+0x0/0xc6 [dm_mod]
Aug 25 08:27:46 boar [<02478825>] bio_endio+0x50/0x55
Aug 25 08:27:46 boar [<024ca038>] __end_that_request_first+0x185/0x47c
Aug 25 08:27:46 boar [<f0c711eb>] scsi_end_request+0x1a/0xa9 [scsi_mod]
Aug 25 08:27:46 boar [<02458f04>] mempool_free+0x5f/0x63
```

```
Aug 25 08:27:46 boar
Aug 25 08:27:46 boar [<f0c713c3>] scsi_io_completion+0x149/0x2f3 [scsi_mod]
Aug 25 08:27:46 boar [<f0c333b9>] sd_rw_intr+0x1f1/0x21b [sd_mod]
Aug 25 08:27:46 boar [<f0c6d3b9>] scsi_finish_command+0x73/0x77 [scsi_mod]
Aug 25 08:27:46 boar [<024cbfa2>] blk_done_softirq+0x4d/0x58
Aug 25 08:27:46 boar [<02426452>] __do_softirq+0x84/0x109
Aug 25 08:27:46 boar [<0242650d>] do_softirq+0x36/0x3a
Aug 25 08:27:46 boar [<024050b7>] do_IRQ+0xad/0xb6
Aug 25 08:27:46 boar [<024023fa>] default_idle+0x0/0x59
Aug 25 08:27:46 boar [<0240242b>] default_idle+0x31/0x59
Aug 25 08:27:46 boar [<024024b1>] cpu_idle+0x5e/0x74
Aug 25 08:27:46 boar ========================
Aug 25 08:27:46 boar Code: 5d c3 55 57 89 c7 56 89 ce 53 bb 01 00 00 00 83 ec
0c 8b 68 3c 83 7f 20 00 8b 45 00 8b 00 89 44 24 04 8b 45 04 89 04 24 8b 40 04
<8b> 40 28 89 44 24 08 0f 85 86 00 00 00 f6 47 10 01 75 0a 85 c9
Aug 25 08:27:46 boar EIP: [<f0ce6507>] clone_endio+0x29/0xc6 [dm_mod]
SS:ESP0068:0b483e38
Aug 25 08:27:46 boar Kernel panic - not syncing: Fatal exception in interrupt
```

All you need is to put the oops into a file and then send this file as part of your problem report to the Parallels support team.

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, 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 Container Management

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

Failure to Create a Container

An attempt to create a new Container fails. There is a message on the system console: `Cached package set XXX version YYY not found.`

Solution 1

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

Solution 2

The Container private area might not be pre-cached. In this case the `vzpkgcache` utility shall be used. Issue the command:

```
vzpkgcache
```

The utility looks for the OS templates installed on the server and caches those that are not cached. After this, try to create a Container once again.

Failure to Start a Container

An attempt to start a Container 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 `pctl set --save` command (see Performing Initial Configuration (p. 30) for instructions). Try to start the Container once again.

Solution 2

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

```
pctl set CT_ID --ipadd IP_addr --save
```

where *CT_ID* represents the Container numeric identifier and *IP_addr* represents an actual IP address.

Solution 3

Poor UBC parameters might prevent the Container from starting. Try to validate the Container configuration (see Validating Container Configuration (p. 121)). See what configuration parameters have caused the error and set appropriate values using the `pctl set --save` command.

Solution 4

The Container might have used all its disk quota (either disk space or disk inodes). Check the Container disk quota (see the Managing Disk Quotas section (p. 92) and Chapter 4 for details) and increase the quota parameters if needed (see Setting Up Per-Container Disk Quota Parameters (p. 95)).

Solution 5

Run the `vzfsutil` utility to make sure that the VZFS symlinks inside the Container work correctly. For example:

```
vzfsutil --call -t /vz/template /vz/private/<CT_ID>
```

The complete reference on the `vzfsutil` utility is provided in the *Parallels Command Line Reference Guide*.

Solution 6

The Container administrator might have inadvertently modified, replaced, or deleted any file that is part of an application or OS template, which has brought about the Container malfunction. In this case, restore the file(s) with the `pctl recover` command (see the Reinstalling Container section (p. 67) for details).

Solution 7

Restore the latest operable copy of the Container by means of the `vzrestore` utility (see the Managing virtual machine and Container Backups section (p. 40) for details).

Failure to Access Container From Network

Solution 1

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

```
# grep IP_ADDRESS /etc/vz/conf/<CT_ID>.conf
IP_ADDRESS="10.0.186.101"
```

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

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

Solution 2

Make sure the routing to the Container is properly configured. Containers can use the default router for your network, or you may configure the server as rooter for its Containers.

Failure to Log In to Container

The Container starts successfully, but you cannot log in.

Solution 1

You are trying to connect via SSH, but access is denied. Probably you have not set the password of the `root` user yet or there is no such user. In this case, use the `pctl set --userpasswd` command. For example, for Container 101 you might issue the following command:

```
# pctl set 101 --userpasswd root:secret
```

Solution 2

Check forwarding settings by issuing the following command:

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

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

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

Getting Technical Support

Preparing and Sending Questions to Technical Support

In most cases, the support team must rely on the customer's observations and communications with the customer to diagnose and solve the problem. Therefore, the detailed problem report is extremely important. You can submit a support report by visiting the http://www.parallels.com/en/support/virtuozzo/request/ web page and filling in the Online Support Form. When describing the problem, please do mention the following:

- symptoms of the problem
- when the problem began including the circumstances of the failure
- any changes you made to your system
- other information that may be relevant to your situation (e.g. the installation method)
- specific hardware devices that may be relevant to your problem

You can also make use of the Parallels Helpdesk support tool. To do this:

1   Follow the https://helpdesk.parallels.com/ link.

2   Register with the Parallels Helpdesk (if you have not done so before) by clicking the Get Access to Parallels RT link on the Helpdesk login page and following the instructions provided on the Activate Your Support Account screen.

3   Log in to the Helpdesk using the received credentials.

4   At the top of the RT At Glance screen, select the Parallels Server Bare Metal component your problem relates to on the drop-down menu, and click the New Ticket in button:

5   On the Create New Ticket screen, fill in the appropriate fields, describe your problem, and click the Create button to make a new support ticket.

Another way of getting help is to directly call us or visit one of our offices. The information about phone numbers, contact people and office addresses is available on the contact pages at http://www.parallels.com/en/contact and http://www.parallels.com/en/support/phone/.

Submitting Problem Report to Technical Support

Parallels Server Bare Metal is shipped with a special utility - vzreport - allowing you to compile a detailed report if you have any problems and to automatically send it to the Parallels support team. After receiving your report, the support team will closely examine your problem and make its best to solve it as quickly as possible.

vzreport has two modes of execution — full screen and command line. By default, the utility starts in the full screen mode. However, you can force the utility to run in the command line mode by specifying any option containing your contact information (e.g. -n denoting your name) or the problem report description (e.g. -m used to provide additional information on your problem). Detailed information on all the options that can be passed to vzreport in the command line is provided in the *Parallels Command Line Reference Guide*.

After running the vzreport utility in the full screen mode, the Problem Report Wizard is opened, which will guide you through a number of steps asking you to provide the necessary information to generate a problem report. On the Welcome screen, just click Next to proceed with the wizard. You will be presented with the following window:
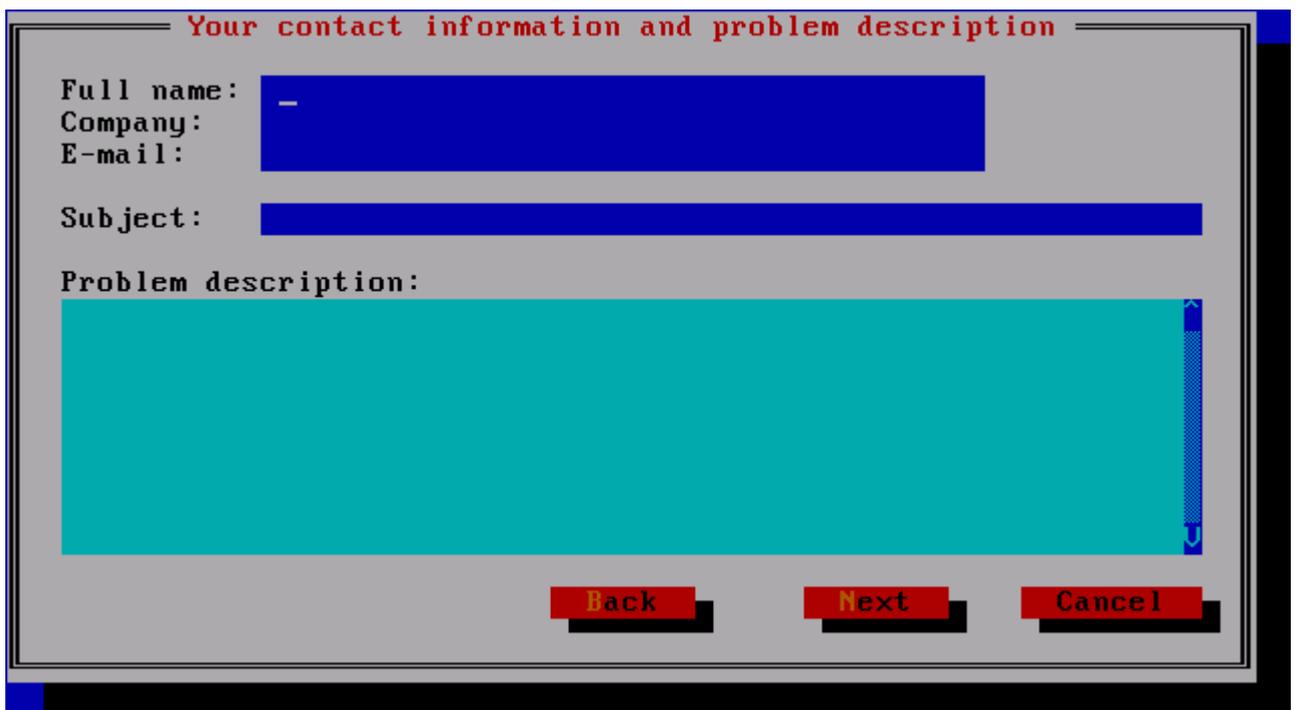
In this window, you should enter your name, e-mail, and the name of your company into the corresponding fields. Make sure that you type a valid e-mail address. Otherwise, the Parallels support team will not be able to contact you. In the Subject field, specify what problem you encountered. You can also provide additional information in the Problem description field which, in your opinion, can help solve the problem.

Clicking Next in the Your contact information and issue description window starts collecting Parallels Server Bare Metal logs and the information on your system and network settings into a special file. You can view the progress in the Gathering Information window. This file will be sent to the Parallels support team upon the completion of the wizard. The file does not contain any private information!

After the utility has gathered all the necessary information on your server, the **Submit report** window is displayed:
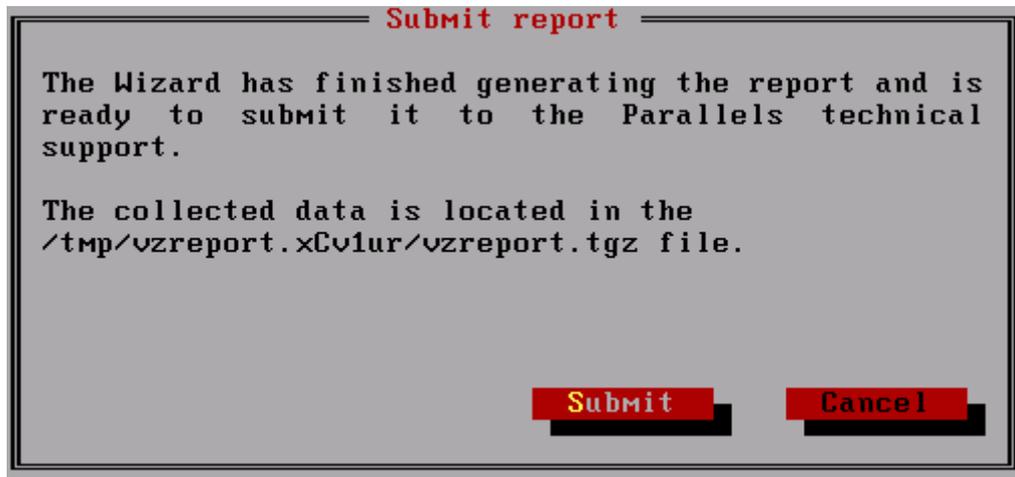
In this window you can do one of the following:

- Click the **Submit** button to send your problem report to the Parallels technical support team. The report is dispatched directly to Parallels by using the HTTP protocol and port 80. However, if you use an HTTP proxy server for handling all your HTTP requests and wish your problem report to be sent via this server, you should specify the hostname or IP address of the server in the /etc/vz/vz.conf configuration file on the server as the value of the HTTP_PROXY parameter. After the problem report has been successfully sent to the Parallels support, the **Congratulations** window is displayed informing you:

  - Of the ID assigned to your report. Use this ID every time you communicate with the Parallels support via e-mail or the Parallels Helpdesk support tool

  - That an e-mail message providing you with detailed information on your problem report has been sent to the e-mail address you specified in the **E-mail** field of the **Your contact information and issue description** window.

Click the **Cancel** button if you do not wish to dispatch the problem report to the support team at the moment. You can do it later on by manually sending the generated zip file to the Parallels support team. The full path to this file is indicated in the **Submit report** window.

Establishing Secure Channel to Parallels Support

Parallels Server Bare Metal provides you with a special tool - *Support Tunnel* - which allows you to establish a private secure channel to the Parallels support team server. After establishing such a channel, the support team will be able to quickly and securely connect to your Parallels server and diagnose and solve your problem. The secure connection to your server is achieved through a Virtual Private Network (VPN) created between the Parallels support team server and your server.

To start using the *Virtuozzo Support Tunnel* tool:

- Make sure the `openvpn` (version 2.0 and above) and `vzvpn` packages are installed on your server. These packages are automatically installed during the Parallels Server Bare Metal installation.
- Make sure that port 80 is opened on the server.
- Edit the `/etc/vzvpn/vzvpn.conf` file to specify the correct parameters for your proxy server, if you use any. Detailed information on these parameters is given in the **vzvpn Configuration File** subsection of the *Parallels Command Line Reference Guide*.

After you have completed the tasks above and in case you encountered a problem, you can do the following to get assistance from the Parallels support:

1 Obtain a special certificate from Parallels which will uniquely identify you as a Parallels Server Bare Metal user. Certificates are issued by Parallels in the form of files and should be installed on your server by issuing the `vzvpn.sh key-install` *certificate* command where *certificate* denotes the name of the certificate file obtained from Parallels. You can get a certificate in one of the following ways:

  - Visit the http://www.parallels.com/en/support/virtuozzo/certificates web site, fill up the **Request Secure Virtuozzo Support Tunnel Certificate** form, and click the **Submit** button. After a while, a certificate will be generated and sent to the email address you provided in the **Request Secure Virtuozzo Support Tunnel Certificate** form.

  - Contact the Parallels support team via e-mail or by telephone and ask for a valid certificate.

2 After you are ready with the certificate installation, make sure your server is connected to the Internet.

3 On the server, execute the `/etc/init.d/vzvpn.sh start` command to establish a VPN between your server and the Parallels support server.

4 Contact the Parallels support team (by telephone or via e-mail) and inform them of the problem you encountered. You should also mention that you have launched the *Virtuozzo Support Tunnel* tool and established a VPN to the Parallels support server.

5 After that, the Parallels support team will connect to your server by using the secure VPN established, closely examine your problem, and make its best to solve the problem as quickly as possible.

Notes:

1. `Support Tunnel` is implemented as a standard Linux service running in the background of your system. Therefore, to have this service running after your server reboot, you should set it to the `autoboot` mode or start it manually again by executing the `/etc/init.d/vzvpn start` command.

2. To close the VPN session with the Parallels support server, you should issue the `/etc/init.d/vzvpn stop` command on the server.

Glossary

Application template. A template used to install a set of applications in *Containers*. See also *Template*.

Container (or *regular Container*). 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. Containers share one *Parallels server* and one OS kernel. However, they are isolated from each other. A Container is a kind of 'sandbox' for processes and users.

Guest operating system (Guest OS). An operating system installed inside a virtual machine and Container. It can be any of the supported Windows, Linux, or Mac operating systems.

Hardware virtualization. A virtualization technology allowing you to virtualize physical servers at the hardware level. Hardware virtualization provides the necessary environment for creating and managing Parallels virtual machines.

Operating system virtualization (or *OS virtualization*). A virtualization technology allowing you to virtualize physical servers at the operating system (kernel) level. OS virtualization provides the necessary environment for creating and managing Parallels Containers.

OS template (or *Operating System template*). A template used to create new *Containers* with a pre-installed operating system. See also *Template*.

Package set. See *Template*.

Parallels Management Console. A Parallels Server Bare Metal management and monitoring tool with graphical user interface. Parallels Management Console is cross–platform and can run on Microsoft Windows, Linux, and Mac computers.

Parallels Server. A hardware virtualization solution that enables you to efficiently use your physical server's hardware resources by sharing them between multiple virtual machines created on this server.

Parallels server (or *physical server* or *server*). A server where the Parallels Server Bare Metal software is installed for hosting Parallels virtual machines and Containers. Sometimes, it is marked as Container 0.

Parallels Server Bare Metal license. A special license that you should install on the physical server to be able to start using Parallels Server Bare Metal. Every physical server must have its own license installed.

Parallels Virtuozzo Containers for Linux. An operating system virtualization solution allowing you to create multiple isolated Containers on a single physical server to share hardware, licenses, and management effort with maximum efficiency.

Private area. A part of the file system storing *Container* files that are not shared with other *Containers*.

Template (or *package set*). A set of original application files (packages) repackaged for mounting over Virtuozzo File System. There are two types of templates. OS Templates are used to create new *Containers* with a pre-installed operating system. Application templates are used to install an application or a set of applications in *Containers*.

UBC. An abbreviation of *User Beancounter*.

User Beancounter. The subsystem of the Parallels Server Bare Metal software for managing *Container* memory and some system-related resources.

Virtual Environment (or *VE*). An obsolete designation of a *Container*.

Virtuozzo File System (*VZFS*). A virtual file system for mounting to Container private areas. VZFS symlinks are seen as real files inside *Containers*.

Virtual machine (VM). A computer emulated by Parallels Server Bare Metal. Like a Container, a virtual machine is functionally identical to an isolated standalone computer, with its own IP addresses, processes, files, its own users database, its own configuration files, its own applications, system libraries, and so on. However, as distinct from Containers, virtual machines run their own operating systems rather than sharing one operating system kernel.

Index

W

Z