

AFFIDAVIT OF MINA CHING

1. I am a Records Request Processor at the Internet Archive. I make this declaration of my own personal knowledge.
2. The Internet Archive is a website that provides access to a digital library of Internet sites and other cultural artifacts in digital form. Like a paper library, we provide free access to researchers, historians, scholars, and the general public. The Internet Archive has partnered with and receives support from various institutions, including the Library of Congress.
3. The Internet Archive has created a service known as the Wayback Machine. The Wayback Machine makes it possible to browse more than 450 billion pages stored in the Internet Archive's web archive. Visitors to the Wayback Machine can search archives by URL (i.e., a website address). If archived records for a URL are available, the visitor will be presented with a display of available dates. The visitor may select one of those dates, and begin browsing an archived version of the Web. Links on archived files in the Wayback Machine point to other archived files (whether HTML pages or other file types), if any are found for the URL indicated by a given link. For instance, the Wayback Machine is designed such that when a visitor clicks on a hyperlink on an archived page that points to another URL, the visitor will be served the archived file found for the hyperlink's URL with the closest available date to the initial file containing the hyperlink.
4. The archived data made viewable and browseable by the Wayback Machine is obtained by use of web archiving software that automatically stores copies of files available via the Internet, each file preserved as it existed at a particular point in time.
5. The Internet Archive assigns a URL on its site to the archived files in the format `http://web.archive.org/web/[Year in yyyy][Month in mm][Day in dd][Time code in hh:mm:ss]/[Archived URL]` aka an "extended URL". Thus, the extended URL `http://web.archive.org/web/19970126045828/http://www.archive.org/` would be the URL for the record of the Internet Archive home page HTML file (`http://www.archive.org/`) archived on January 26, 1997 at 4:58 a.m. and 28 seconds (1997/01/26 at 04:58:28). The date indicated by an extended URL applies to a preserved instance of a file for a given URL, but not necessarily to any other files linked therein. Thus, in the case of a page constituted by a primary HTML file and other separate files (e.g., files with images, audio, multimedia, design elements, or other embedded content) linked within that primary HTML file, the primary HTML file and the other files will each have their own respective extended URLs and may not have been archived on the same dates.





6. Attached hereto as Exhibit A are true and accurate copies of browser screenshots of the Internet Archive's records of the archived files for the URLs and the dates specified in the attached coversheet of each screenshot.
7. I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

DATE: 11/14/2025



Mina Ching

Please see attached
All Purpose
Jurat form
for additional
Notary Events



EXHIBIT A

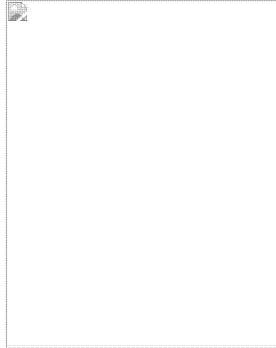


<https://web.archive.org/web/20080719083919/http://www.phonefactor.com/how-it-works>



PhoneFactor.com

How It Works



Two easy steps for PhoneFactor:

Step 1:

Enter your username & password.

Step 2:

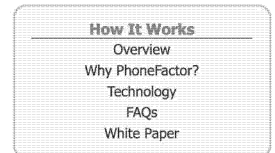
Instantly, you receive a phone call. Answer and press #.

That's It!

This simple process provides two factors of authentication through two separate channels (your computer and your phone service):

- Something you know - your password.
- Something you have - your telephone.

A hacker would have to know your password and have your telephone to login as you.



<https://web.archive.org/web/20080706161814/http://www.phonefactor.com/how-it-works/technology>



Demo
Download
Log In

1.877.NO.TOKEN
(1.877.668.6536)

PhoneFactor.com

Technology

Architecture

Agent

The PhoneFactor Agent runs on any Windows computer. It includes a configuration wizard that guides you through the setup process for any application you wish to use with PhoneFactor. You can import users, configure authentication settings, place test calls, and check the status of other agents on your network using this simple tool.

Web SDK

Instead of using the Agent to secure applications and manager users, the PhoneFactor SDK supports implementation with .NET, JAVA, Perl, Ruby, and PHP applications.

Online Management Portal

Use the web-based management portal from anywhere to install new agents, manage your existing agents, view usage reports, and enable additional modules.

PhoneFactor Service

The PhoneFactor Agent or SDK adds a second authentication step – a confirmation phone call – to your existing authentication process. If the username and password are correct, an SSL request is sent to one of the PhoneFactor data centers around the world. The data center calls the user, who confirms the login by answering and pressing the # key. Finally, it returns success or failure to the application.

Setup

PhoneFactor Agent

Step 1:

Download and install the PhoneFactor Agent.

Step 2:

Add users and their phone numbers.

That's It!

Note: Only the users you configure will get PhoneFactor calls when signing on.

Web SDK

Step 1:

Download the Web SDK for .Net, PHP, Perl, Ruby or Java.

Step 2:

Use the code samples to integrate PhoneFactor with you site's login or transaction process.

That's It!

Note: Users and their phone numbers will be maintained in your user database.



Demo PhoneFactor
See for yourself
how easy it is.



Get PhoneFactor
It's Free
Sign up now to get started.

How It Works

Overview

Why PhoneFactor?

Technology

FAQs

White Paper



© 2007-2008 Positive Networks, Inc.

[About](#) [News](#) [Sitemap](#) [Legal](#) [Support](#) [Customer Login](#)



<https://web.archive.org/web/20080705142441/http://www.phonefactor.com/how-it-works/white-paper>



Demo
Download
Log In

1.877.NO.TOKEN
(1.877.668.6536)

PhoneFactor.com

White Paper

Two-Factor Authentication Without Tokens: PhoneFactor™

For most companies, information security is a top priority. Demand for protecting data and employee confidentiality is only continuing to grow, especially in industries that require a regulatory-compliant environment. However, applying usernames and passwords for authentication is insufficient. While two-factor authentication is an effective security solution, traditional token-based systems have been difficult to implement and administer, leading to limited adoption. PhoneFactor, from Positive Networks, uses any mobile phone (or traditional phone) as the second authentication factor. Users do not need to carry an additional device, and there are no expensive tokens to manage. During login, PhoneFactor makes a call to the user's phone, confirming the authentication. This second factor - the possession of the phone itself - adds a significant additional layer of security. PhoneFactor can be set up in hours without the purchase of any hardware. This paper discusses the technical architecture of PhoneFactor, along with related security, deployment and integration issues.

Introduction

Authentication, which is the process by which a computer system positively identifies a user, is commonly considered to be one of the weakest links in modern computer security systems. Every day a new story emerges about an identity theft or a computer break-in due to stolen credentials. With the proliferation of network-based and online applications, the trend is only going to continue. Unfortunately, the dominant authentication system in production today is based on usernames and passwords. This relatively weak system is subject to a number of flaws, including notoriously poor user password choices, password harvesting via keylogging software, phishing and man-in-the-middle attacks, and others.

The most common solution to these authentication problems is to use a two-factor authentication system.

Two-factor authentication works by requiring both **something the user has** and **something the user knows**, as opposed to just something known (typically a password). The "something you have" is usually a piece of hardware that is impossible (or at least very difficult) to duplicate, and the "something you know" is typically a password or PIN. Two-factor authentication systems are secure because it is very difficult to obtain possession of both factors. Even if an attacker manages to learn the user's password, it is useless without also having physical possession of the token. Conversely, if the user happens to lose the physical token, the finder of that token won't be able to use it unless he or she can also guess the user's password.

Security professionals have deployed various two-factor solutions, but no solution has widely displaced traditional username and password authentication. The industry has seen enterprise deployments of token-based systems from vendors such as RSA™ and VeriSign™, smartcard-based solutions, and various forms of biometric authentication. Each solution has significant drawbacks, historically leading to limited adoption by users. For online consumer logins, such as banking and financial websites, these limiting factors have dramatically limited two-factor deployments. These websites have instead opted to deploy security questions and image identification as an additional layer of security. PhoneFactor provides a strong, two-factor authentication service without the drawbacks. It uses the public telephone network for the second authentication factor, which allows PhoneFactor to be deployed rapidly and inexpensively while maintaining the advantages of a two-factor authentication system.

PhoneFactor Overview

PhoneFactor implements two-factor authentication using the user's telephone as the second factor of authentication. The authentication process begins as a normal login to the system, in which the user supplies a username and password. If the supplied credentials are valid, the system initiates a phone call to the user's registered phone number. The user then answers the phone and indicates whether or not the authentication should succeed by entering an optional PIN. Once the user acknowledges the authentication attempt via the phone call, the system completes the pending authentication and the login proceeds as normal. If the user did not request authentication, action can be taken to lock the user account and prevent an attack.

PhoneFactor's security rests on its use of the user's telephone as the second authentication factor. Telephones are extremely difficult to duplicate and phone numbers are extremely difficult to intercept. The combination of their phone, a physical possession, and a memorized password yields strong, two-factor authentication that provides minimal hassle to the user.

This architecture has a number of advantages over traditional two-factor systems. Most importantly, it doesn't require the user to carry an extra physical device. It instead leverages a device the user carries with them already. In addition, companies aren't excited about coordinating the logistics of issuing, mailing, RMAing, and servicing tokens. Most users already have mobile phones (there are about 1 billion active mobile subscriptions today), and they're going to carry them regardless. And most importantly, each device has a unique identifier that's almost impossible to copy: the phone number.

In addition to the advantage of using an existing device, PhoneFactor is also the only two-factor authentication system in the world that allows for instant attack detection. Every authentication attempt in which the attacker knows the user's username and password will generate a phone call to the (true) user. That user can immediately block their account and instantly notify the company's fraud department who can instantly take appropriate action.

The phone-based system also improves resilience to phishing attacks. Phishing attacks generally work by fooling the user into entering credentials into a fake authentication form. This is possible because users generally have no way to authenticate the form itself. While most two-factor solutions can't solve this problem, PhoneFactor can leverage its unique, out-of-band authentication call to prove to the user that it really is PhoneFactor calling. Other solutions must rely on application modification using technologies like



This is not a theoretical problem. A recent phishing attack used this technique against a bank whose customers used RSA's SecurID™ product for two-factor authentication.

PhoneFactor Implementation Details

PhoneFactor is implemented as a web service and either an Agent or a web plug-in. The web service is hosted by Positive Networks, and the Agent software runs on the customer's premises as a part of the site's authentication infrastructure. PhoneFactor integrates with a wide variety of off-the-shelf applications and IT technologies.

PhoneFactor doesn't require changes to existing applications, so integration is typically fast and easy. PhoneFactor adds a second, confirmatory authentication to existing one-factor authentication architectures. Integration with existing applications and infrastructures is done using one of several protocols. Most routers, RAS servers, VPN appliances, and other related hardware use the RADIUS protocol for authentication. RADIUS allows a central server to maintain a username and password database, allowing shared authentication databases and centralized account management. PhoneFactor plugs into the RADIUS protocol exchange by acting as a standard RADIUS proxy server. When a RADIUS request is made by an element to the PhoneFactor agent, it is first forwarded to the target RADIUS server. Once the authentication is successful, the agent triggers the secondary authentication phone call. If the user answers and approves the authentication, the user's authentication message is returned back to the original requesting device with an access-accept message. If the secondary authentication call fails, the requestor is notified of this fact and the entire authentication attempt fails.

Web-based applications running on Microsoft Internet Information Server integrate with PhoneFactor via a standard Microsoft IIS plug-in called an ISAPI filter. The filter implements both HTTP authentication support (basic and integrated) and form-based authentication (FBA) support. In the case of HTTP authentication, the browser supplies credentials with every request, and for requests that the web server authenticates successfully, the PhoneFactor filter triggers secondary phone-based authentication, in a similar manner. Authentication state is maintained by means of a session cookie.

In the case of form-based authentication (FBA) applications, PhoneFactor integrates using a combination of pre- and post-authentication. PhoneFactor never places a secondary authentication call to a user without first verifying that the primary authentication (using username and password) is successful. However, because there is no standard way to learn the status of FBA authentication after it takes place, and in order to avoid creating an unnecessary state in web applications, the ISAPI filter performs a pre-authentication step before submitting the form's contents to the server for processing. During pre-authentication, the user's primary credentials (username and password) are tested against the authentication database, which in most cases is either Microsoft Active Directory or the local Windows user account database. If this pre-authentication step indicates that the credentials are valid, the secondary authentication call is placed to the user, and if that is also successful, the form is then submitted to the application. Of course, by that time it is known that the login will be successful. This mechanism ensures the broadest possible compatibility with web applications.

Finally, PhoneFactor supports integration with the Windows authentication process at either the workstation level or at the domain level. This allows users to receive secondary authentication calls for Windows domain login, drive mapping, Exchange access, and so on.

PhoneFactor Agents are installed on various servers at the customer's site. The basic PhoneFactor package treats each server individually, and the available Extended Deployment Pack adds the ability to build sophisticated networks of PhoneFactor agents that share load and provide for fault tolerance.

Exactly which servers are used depends upon the types of authentication that are to be subjected to secondary confirmation by PhoneFactor. In general, the software is installed on each web server with a PhoneFactor-protected resource, and on each domain controller on a protected Windows domain. The software can be installed on any Windows computer when providing secondary authentication for RADIUS. In general, it's a good idea to plan appropriate levels of load balancing and redundancy into the deployment.

PhoneFactor runs as a Windows service and starts automatically on boot. Agents communicate with each other and with Positive Networks data centers using strong mutual authentication based on X.509 certificates and secure Windows RPC. All per-user data, including the actual list of PhoneFactor users, is stored on the Agents at the customer site, and is only sent to Positive Networks other than on a one-at-a-time basis to trigger a secondary authentication call.

PhoneFactor Web SDK

The PhoneFactor Web SDK offers integration with virtually any web application, and includes web plug-ins for .NET, Java, PHP, Ruby, and Perl among others. The web plug-in can be used to integrate into a web site's existing login or transaction processes and leverages the site's existing user database. Like the Agent, the web plug-in communicates with the web service hosted by Positive Networks, which places the authentication call and interprets the results.

PhoneFactor Implementation Security Considerations

The PhoneFactor service was developed from the ground up, using the latest in secure software design methodologies. The entire system includes strong, mutual authentication, and all network communications are encrypted using high strength cryptography algorithms. Industry-accepted cryptographic standards are used at every point in the design: authentication is based on X.509 certificates (both client and server), data transport is done using SSL or secure RPC (the same protocol used among domain controllers in Windows networks), and secure resources such as certificates and keys are stored using secure storage providers built into the operating system. Cryptographic-quality random numbers are used whenever randomness is needed.

The data architecture of PhoneFactor is designed to put administrators in total control of their authentication information. All per-user data is stored on the customer site, including the list of users enabled for PhoneFactor. The only information that is passed to Positive Networks during an authentication is the minimum necessary for appropriate auditing and for the placement of the secondary authentication call.

Administration of the system can be delegated to others via the PhoneFactor website, allowing clean sharing of management responsibilities while retaining a complete audit trail.

PhoneFactor As a Two-Factor Authentication System

The security of PhoneFactor lies in its ability to strongly authenticate users based on proof that they know a secret (their password) and are in physical possession of a unique physical device (their phone). While it may



secret (their password) and are in physical possession of a unique physical device (their phone). While it may be possible (or even easy) for an attacker to gain access to a user's login credentials, it is usually a much more difficult problem to obtain that same user's phone.

Password theft is rampant on the Internet, and the proliferation of botnets and phishing sites means that stolen credentials will probably be a fact of life for years to come. PhoneFactor neutralizes the threat of password theft, since phones are not vulnerable to being stolen in the same ways that passwords are. While mobile phones can theoretically be diverted to third parties by phone companies, individual attackers have a much harder time gaining access to a phone number. Modern mobile phone networks make use of some form of encryption or other link-level security. Mobile phone "cloning" attacks have been described in academic papers, but the techniques they describe are aimed more at unauthorized access to the network rather than taking over a particular phone number, and in any case, it is generally accepted that phone number hijacking is beyond the capabilities of most attackers.

But beyond securing a given single user from password theft attacks, PhoneFactor dramatically increases the difficulty of mass password theft attacks that are becoming so common with phishing sites and keyloggers. There's virtually no chance that an attacker will be able to use or sell a list of authentication credentials that are protected by PhoneFactor.

When the High Strength Authentication Pack is added, PhoneFactor can be configured in either Standard Mode or PIN Mode. Standard Mode requires only that the user answer the phone and press a button to prove that the authentication is expected and approved. In PIN mode, users are required to enter a PIN that functions essentially as a third authentication factor. This makes it even more difficult for an attacker that comes into possession of the user's phone to get logged in.

Existing single-step authentication solutions are often vulnerable to phishing attacks, where the user is tricked into supplying credentials to an attacker impersonating a legitimate application. Most existing two-factor solutions are vulnerable to this attack as well, because users have no way to know if the system they're communicating with is the genuine system or a man-in-the-middle. PhoneFactor solves this problem as well, using either a user-supplied voice message (similar to recording a voice mail greeting) that only the legitimate PhoneFactor server could have, or by having the end user select a random authentication word for the PhoneFactor server to repeat to the user at every authentication call. The latter system is similar to the SiteKey™ system in use by many financial websites.

Comparisons With Other Systems

Two-factor systems have been on the market for years, and the basic concepts have been refined over that time such that the basic security of two-factor has been well established. However, every other system produced to date has had serious drawbacks that have prevented widespread deployment. Due to its unique phone-based architecture, PhoneFactor overcomes these drawbacks and sets the stage for easy, widespread deployment of two-factor technology for the first time. One of the most common existing two-factor systems is based on hardware tokens that generate a pseudo-random sequence of digits. The most popular of these systems is RSA Data Security's SecurID™ system. While these systems provide an additional level of security over legacy one-factor authentication systems, they bring with them several drawbacks. Nobody likes token management, and users don't enjoy waiting to receive tokens (or replacement tokens). Companies don't like inventorying, mailing, and accounting for tokens. Furthermore, the logistics of token distribution becomes painful quickly as the number of users grows.

Tokens bring other operational issues as well. Most systems require some form of token synchronization, and not all applications support the token synchronization protocol. Synchronization often turns into a helpdesk call, and regardless, it represents yet another manual process that companies must manage. Because token-based systems require users to change their behavior substantially, significant training is needed. Users sometimes have a hard time remembering which order the PIN and the token digits are entered, and training users to "wait for the bars" is difficult. Some systems even require administrators to modify applications before they will work, invoking all of the change control difficulties associated with non-standard vendor software.

Finally, tokens of this kind are typically expensive and time-limited. Add to that the expense of buying server hardware and software, training IT personnel to manage the system, the various costs associated with rollout/training, and you have a very expensive solution.

Another common two-factor solution involves the use of smartcards. Smartcards are credit card-sized tokens that have an embedded private key that is protected by a PIN or password. This private key positively identifies the user to the system. Like tokens, users are required to carry around a new object that they didn't have before. Cards must be mailed, handled; RMA'd, and so on, creating similar logistical problems. And, since very few modern computers (and virtually no older systems) have built-in smartcard readers, an additional piece of hardware, together with drivers (and associated platform dependencies), must be distributed to users. Smartcards present lockout risks – most cards deactivate themselves after a certain number of failed attempts, and require physical replacement – leading to increased IT management time, and in most cases require regular updating to stay current. Finally, few applications have native support for smartcard technology, and those that do often have narrow support for operating system versions, card reader models, and so on. Adding smartcard support to applications is often difficult.

Biometric authentication is another form of two-factor authentication that is occasionally deployed, although currently its use is typically only seen in ultra high security applications (e.g. the defense industry). Biometric devices are a mixed bag when it comes to availability, compatibility, price, and security. For example, one well-known vendor distributes a fingerprint reader that uses an insecure data path to transmit biometric data back to the computer for analysis, making it vulnerable to the very keylogging attacks it tries to avoid. In any case, biometric data cannot easily be changed in case it is compromised (e.g. fingerprint checksums are stolen, etc.), and integration is even more challenging and less available than with smartcards. Training is a significant issue and logistics are perhaps more difficult than any other solution. The authenticator has to obtain the biometric data from the user the first time. Finally, one of the biggest reasons few companies employ biometric authentication is the cost of the hardware.

PhoneFactor generally does not suffer from any of the above challenges. Because users already have phones, there is no hardware distribution. If a user's phone is lost, that user has the responsibility of replacing it, not the company. All that is required for provisioning is getting users' phone numbers, and often, that data is already available in a company directory. Because PhoneFactor adds a secondary authentication call to an existing authentication system, it doesn't require application-specific changes. It is compatible with a broad range of applications and uses standard, documented interfaces to add the secondary authentication step, thereby helping to ensure future compatibility. User training is simple; as long as a user is told to expect a phone call during login, the rest is self-explanatory. At worst, the training is explained during the authentication call itself. There are never synchronization or lock-out issues, and the user's computer needs no extra hardware or drivers. Because of how it integrates, PhoneFactor natively supports clients running on



Macintosh, Linux and PDAs. As long as the server uses an industry-standard authentication method like RADIUS, HTTP authentication, FBA, or Windows authentication, PhoneFactor can support clients running on virtually any platform.

Conclusion

PhoneFactor enables new classes of users to experience the benefits of strong authentication without placing undue burdens on the company or on the budget. Rollout is fast and maintenance is easy. And, PhoneFactor is backed by Positive Networks, a company with extensive experience in managed security. Usernames and passwords no longer provide enough security in many environments. For the first time, PhoneFactor will allow companies to roll out two-factor authentication to anyone, using any application, and provide companies the many benefits of strong authentication.

**Demo PhoneFactor**
See for yourself
how easy it is.

**Get PhoneFactor**
It's Free
Sign up now to get started.

How It Works
[Overview](#)
[Why PhoneFactor?](#)
[Technology](#)
[FAQs](#)
[White Paper](#)



© 2007-2008 Positive Networks, Inc.

[About](#) [News](#) [Sitemap](#) [Legal](#) [Support](#) [Customer Login](#)



https://web.archive.org/web/20141025112001/https://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-intro-virt-history.html



1.1 What is the History of Virtualization?

Chapter 1 Introduction to Virtualization

[Prev](#)

[Next](#)

1.1 What is the History of Virtualization?

The concept of virtualization is generally believed to have its origins in the mainframe days in the late 1960s and early 1970s, when IBM invested a lot of time and effort in developing robust time-sharing solutions. Time-sharing refers to the shared usage of computer resources among a large group of users, aiming to increase the efficiency of both the users and the expensive computer resources they share. This model represented a major breakthrough in computer technology: the cost of providing computing capability dropped considerably and it became possible for organizations, and even individuals, to use a computer without actually owning one. Similar reasons are driving virtualization for industry standard computing today: the capacity in a single server is so large that it is almost impossible for most workloads to effectively use it. The best way to improve resource utilization, and at the same time simplify data center management, is through virtualization.

Data centers today use virtualization techniques to provide abstraction from the physical hardware, create large aggregated pools of logical resources consisting of CPUs, memory, disks, file storage, applications, networking, and offer those resources to users or customers in the form of agile, scalable, consolidated virtual machines. Even though the technology and use cases have evolved, the core meaning of virtualization remains the same: to enable a computing environment to run multiple independent systems at the same time.

[Prev](#)

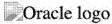
[Up](#)

[Next](#)

Chapter 1 Introduction to Virtualization

[Home](#)

1.2 Why use Virtualization?



Copyright © 2014, Oracle and/or its affiliates. All rights reserved. [Legal Notices](#)



https://web.archive.org/web/20141024221019/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-intro-virt-reasons.html



1.2 Why use Virtualization?

There are many different good reasons for companies and organizations to invest in virtualization today, but it is probably safe to assume that financial motivation is number one on the list: virtualization can save a lot of money. Below is an overview of the key benefits of virtualization.

Table 1.1 Reasons to Use Virtualization

Item	Description
Resource optimization	Today's enterprise level computer resources are so powerful that they often have excess capacity. By virtualizing the hardware and allocating parts of it based on the real needs of users and applications, the available computing power, storage space and network bandwidth can be used much more effectively. Computers no longer need to be idle or performing below their capabilities because there are fewer connected users, or because the hosted application happens to be less demanding than the server can handle. Virtual machines offer software developers isolated, constrained, test environments. Rather than purchasing dedicated physical hardware, virtual machines can be created on the existing hardware. Because each virtual machine is independent and isolated from all the other servers, programmers can run software without having to worry about affecting other applications, or external components affecting the execution of their code.
Consolidation	It is common practice to dedicate individual computers to a single application. If several applications only use a small amount of processing power, the administrator can consolidate several computers into one server running multiple virtual environments. For organizations that own hundreds or thousands of servers, consolidation can dramatically reduce the need for floor space, HVAC, A/C power, and co-location resources. This means the cost of ownership is reduced significantly, since less physical servers and floor and rack space are required, which in turn leads to less heat and power consumption, and ultimately a smaller carbon footprint.
Maximizing Uptime	Agility is all about being able to respond to changing requirements as quickly and flexibly as possible. Virtualization brings new opportunities to data center administration, allowing users to enjoy: • Guaranteed uptime of servers and applications; speedy disaster recovery if large scale failures do occur. • Instant deployment of new virtual machines or even aggregated pools of virtual machines via template images. • Elasticity, that is, resource provisioning when and where required instead of keeping the entire data center in an <i>always-on</i> state. • Reconfiguration of running computing environments without impacting the users.
Automatically Protect Applications from Server Failure	Server virtualization provides a way to implement redundancy without purchasing additional hardware. Redundancy, in the sense of running the same application on multiple servers, is a safety measure: if for any reason a server fails, another server running the same application takes over, thereby minimizing the interruption in service. This kind of redundancy works in two ways when applied to virtual machines: • If one virtual system fails, another virtual system takes over. • By running the redundant virtual machines on separate physical hardware you can also provide better protection against physical hardware failure.
Easily Migrate Workloads as Needs Change	Migration refers to moving a server environment from one place to another. With most virtualization solutions it is possible to <i>move</i> a virtual machine from one physical machine in the environment to another. With physical servers this was originally possible only if both physical machines ran on the same hardware, operating system and processor. In the virtual world, a server can be migrated between physical hosts with entirely different hardware configurations. Migration is typically used to improve reliability and availability: in case of hardware failure the guest system can be moved to a healthy server with limited downtime, if any. It is also useful if a virtual machine needs to scale beyond the physical capabilities of the current host and must be relocated to physical hardware with better performance.
Protect Investment in Existing, Legacy Systems	Server hardware will eventually become obsolete, and switching from one system to another can be difficult. In order to continue offering the services provided by these legacy systems, you can run it as a virtual machine on new, modern hardware, while the legacy system itself still behaves as if it were running on the same legacy hardware. From an application perspective, nothing has changed. In fact, its performance may well benefit from the newer underlying hardware. This gives the organization the time to transition to new processes without worrying about hardware issues, particularly in situations where the manufacturer of the legacy hardware no longer exists or cannot fix broken equipment.



https://web.archive.org/web/20141025200004/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-hypervisor.html



1.3 What are Hypervisors?

If virtualization is defined as enabling multiple operating systems to run on a single *host computer*, then the essential component in the virtualization stack is the *hypervisor*. This hypervisor, also called Virtual Machine Monitor (VMM), creates a virtual platform on the host computer, on top of which multiple *guest* operating systems are executed and monitored. This way, multiple operating systems, which are either multiple instances of the same operating system, or different operating systems, can share the hardware resources offered by the host.

Hypervisors are commonly classified as one of these two types, as show in [Table 1.2, “Hypervisor Types”](#).

Table 1.2 Hypervisor Types

Classification	Characteristics and Description
Type 1: <i>native</i> or <i>bare metal</i>	Native hypervisors are software systems that run directly on the host's hardware to control the hardware, and to monitor the guest operating systems. Consequently, the guest operating system runs on a separate level above the hypervisor. Examples of this classic implementation of virtual machine architecture are Oracle VM, Microsoft Hyper-V, VMware ESXi and Xen.
Type 2: <i>hosted</i>	Hosted hypervisors are designed to run within a traditional operating system. In other words, a hosted hypervisor adds a distinct software layer on top of the host operating system, and the guest operating system becomes a third software level above the hardware. A well-known example of a hosted hypervisor is Oracle VM VirtualBox. Others include VMware Server and Workstation, Microsoft Virtual PC, KVM, <i>QEMU</i> and Parallels.

https://web.archive.org/web/20141024221025/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-ovm-arch.html



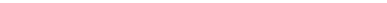
Oracle VM is a platform that provides a fully equipped environment with all the latest benefits of virtualization technology. Oracle VM enables you to deploy operating systems and application software within a supported virtualization environment. Oracle VM insulates users and administrators from the underlying virtualization technology and allows daily operations to be conducted using goal-oriented GUI interfaces. The components of Oracle VM are shown in Figure 2.1, “Oracle VM Architecture”.

[illegible]

- **Client Applications:** Various user interfaces to Oracle VM Manager are provided, either via the graphical user interface (GUI) accessible using a web-browser; the command line interface (CLI) accessible using an SSH client; custom built applications or scripts that use the Web Services API (WS-API); or external applications, such as Oracle Enterprise Manager, or legacy utility scripts that may still make use of the legacy API over TCPS on port 54322. The legacy API is due to be deprecated in the near future and applications that are using it must be updated to use the new Web Services API instead. All communications with Oracle VM Manager are secured using either a key or certificate based technology.
- **Oracle VM Manager:** Used to manage *Oracle VM Servers*, virtual machines, and resources. It is comprised of a number of subcomponents, include a web browser-based user interface; and a command line interface (CLI) allowing you to manage your infrastructure directly from the command line either via external scripts or by running manual command sequences. Each of these interfaces runs as a separate application to the Oracle VM Manager core and interfaces with this using the Web Services API.

Oracle VM Manager communicates with each Oracle VM Server via the Oracle VM Agent, using XML-RPC over HTTPS on port 8899. Actions on servers that are initiated within Oracle VM Manager are triggered using this method. The Oracle VM Agent on each Oracle VM Server is equally able to send notifications, statistics and event information back to Oracle VM Manager. Actions within Oracle VM Manager triggered by Oracle VM Agent are achieved using the Web Services API exposed by Oracle VM Manager and are secured using HTTPS.

Document Id: B47E45D0-C1D1-11F0-946E-27B54F7F29B4
OnlineNotary.net



Page 20/34

- **Oracle VM Manager Database:** Used by Oracle VM Manager core to store and track configuration, status changes and events. Oracle VM Manager uses a MySQL Enterprise database that is bundled in the installer and which runs on the same host where Oracle VM Manager is installed. The database is configured for the exclusive use of Oracle VM Manager and must not be used by any other applications. The database is automatically backed up on a regular schedule, and facilities are provided to perform manual backups as well.
- **Oracle VM Server:** A managed virtualization environment providing a lightweight, secure, server platform which runs virtual machines, also known as domains. At least one Oracle VM Server is required, but several are needed to take advantage of clustering.

Oracle VM Server is installed on a bare metal computer, and contains the *Oracle VM Agent* to manage communication with Oracle VM Manager. *dom0* is an abbreviation for *domain zero*, the management or *control domain* with privileged access to the hardware and device drivers. *DomU* is an unprivileged *domain* with no direct access to the hardware or device drivers. A user-domain (domU) is started and managed on an Oracle VM Server by dom0.

On x86-based systems, Oracle VM Server is based upon an updated version of the underlying Xen *hypervisor* technology, and includes Oracle VM Agent. It also includes a Linux kernel with support for a broad array of devices, file systems, and software RAID volume management. The Linux kernel is run as dom0 to manage one or more domU virtual machines, each of which could be Linux, Oracle Solaris, or Microsoft Windows™.

In contrast, Oracle VM Server for SPARC takes advantage of the hypervisor that is already included within the SPARC firmware, alongside the Oracle VM Agent for SPARC. The default Oracle Solaris operating system is usually promoted to act as the primary domain, which is equivalent to dom0 on x86 systems. Once the primary domain is in place, it can be used to create and manage further domains running different versions of the Oracle Solaris operating system.

Groups of Oracle VM Servers are usually clustered together to create server pools. This allows Oracle VM Manager to handle load balancing and failover for high-availability environments. Virtual machines run within a server pool and can be easily moved between the different servers that make up a server pool. Server pools also provide logical separation of servers and virtual machines. Server pools are required entities within the Oracle VM infrastructure, even if they consist of only one server.

Each Oracle VM Server maintains its own Berkeley Database, used to store local configuration and runtime information. This allows the Oracle VM Server to continue to function normally, even if Oracle VM Manager becomes unavailable for a period. Where Oracle VM Servers are clustered together, a separate cluster database, stored in the server pool file system, is shared between the servers. This allows the server pool to continue to provide clustering features, such as High Availability, even if Oracle VM Manager is unavailable.

- **External Shared Storage:** Provides storage for a variety of purposes and is required to enable high-availability options afforded through clustering. Storage discovery and management is achieved using the Oracle VM Manager, which then interacts with Oracle VM Servers via the storage connect framework to then interact with storage components. This process is discussed in more detail in [Section 3.4, “How do Different Storage Types Connect?”](#). Oracle VM provides support for a variety of external storage types including NFS, iSCSI and Fibre Channel.

[Prev](#)

Chapter 2 Oracle VM Overview

 Oracle logo

[Up](#)

[Home](#)

[Next](#)

2.2 What is Oracle VM Server?

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. [Legal Notices](#)



https://web.archive.org/web/20141024221028/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-ovm-server-overview.html



2.2 What is Oracle VM Server?

- [2.2.1 What Hypervisors are used in Oracle VM?](#)
- [2.2.2 What are Domains?](#)
- [2.2.3 What is Oracle VM Agent?](#)
- [2.2.4 What are Server Pools?](#)

Oracle VM Server can be installed on either x86 or SPARC hardware platforms. Since these platforms are fundamentally dissimilar, the hypervisor used on each platform is different. Oracle VM helps to abstract the hypervisor further by providing a user interface that facilitates the same logical actions across different server types. Much of this abstraction is provided by the Oracle VM Agent that is installed on each Oracle VM Server in the environment.

In this section, the way in which components run on these different hardware platforms and their roles are described in more detail. For more information on the different hardware platforms that are supported and the requirements to run an Oracle VM Server, refer to the [Oracle VM Installation and Upgrade Guide](#).

More detailed coverage of Oracle VM Server is provided in [Chapter 6, Understanding Server Pools and Oracle VM Servers](#).

2.2.1 What Hypervisors are used in Oracle VM?

- [2.2.1.1 Oracle VM Server for SPARC](#)
- [2.2.1.2 Xen™](#)

The *hypervisor* present on each *Oracle VM Server* is an extremely small-footprint virtual machine manager and scheduler. It is designed so that it is the only fully privileged entity in the system. It controls only the most basic resources of the system, including CPU and memory usage, privilege checks, and hardware interrupts.

2.2.1.1 Oracle VM Server for SPARC

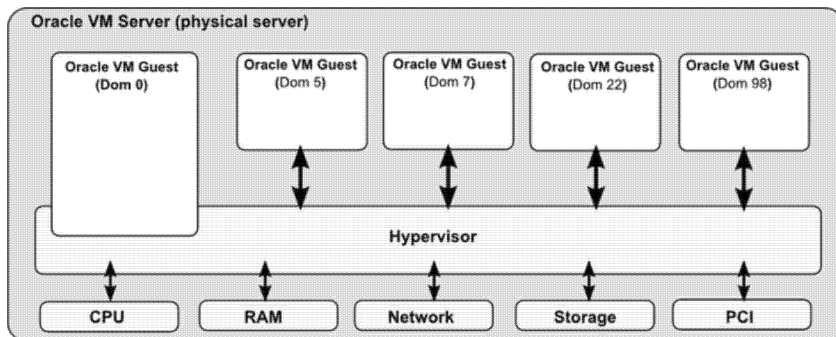
On SPARC systems, the SPARC hypervisor is built into the SPARC firmware and is generally referred to as the *Logical Domains* Manager (LDM). If Oracle VM Server for SPARC has not been installed, the default operating system runs on top of the hypervisor transparently. When Oracle VM Server for SPARC is installed, the default operating system becomes the primary *domain* and tools are provided for the primary domain to manage how resources and hardware are allocated via the hypervisor to other domains.

As with the Xen hypervisor, each virtual machine is securely executed on a single computer and runs its own *guest* Oracle Solaris operating system. The SPARC hypervisor provides a broader range of virtualization features than the Xen hypervisor, due to the nature and design of SPARC hardware.

2.2.1.2 Xen™

Oracle VM makes use of Xen technology, when running on x86 servers, taking advantage of the Xen hypervisor. The Xen hypervisor is a small, lightweight bare metal hypervisor for x86-compatible computers. The Xen hypervisor securely executes multiple virtual machines on one host computer. Each virtual machine runs in its own domain and has its own *guest* operating system with almost native performance. A primary management domain, called dom0, also runs as a guest on top of the hypervisor. See [Section 2.2.2, "What are Domains?"](#) for more information on domains. The Xen hypervisor was originally created by researchers at Cambridge University, and derived from work done on the Linux kernel.

Figure 2.2 Oracle VM Server Hypervisor and Guest Domains



[Click to view larger image ovm_arch_hypervisor.png](#)

2.2.2 What are Domains?

- [2.2.2.1 Management Domain \(dom0\)](#)
- [2.2.2.2 Domains \(domU\)](#)

The terms "domain", "guest" and "virtual machine" are often used interchangeably, but they have subtle differences. A *domain* is a configurable set of resources, including memory, virtual CPUs, network devices and disk devices, in which virtual machines run. A *domain* or *virtual machine* is granted virtual resources and can be started, stopped and restarted independently of other domains or the host server itself. A *guest* is a virtualized operating system running within a *domain*. Multiple guests can run on the same Oracle VM Server, each within its own domain.

2.2.2.1 Management Domain (dom0)

Most of the responsibility of hardware detection in an Oracle VM Server environment is passed to the management domain, referred to as *domain zero* (or *dom0*). On x86-based servers, the dom0 kernel is actually a small-footprint Linux kernel with support for a broad array of devices, file systems, and software RAID and volume management. In Oracle VM Server, the dom0 is tasked with providing a view of the system hardware available to the hypervisor. It also allows you to interact directly with the hypervisor to control access to resources as well as performing various tasks such as creating, destroying and controlling *guest* operating systems, and presenting those guests with a set of common virtual hardware. Other domains never interact with dom0 directly. Their requirements are handled by the hypervisor itself. Dom0 only provides a means to administer the hypervisor.

On SPARC-based servers, the management domain, usually referred to as the primary service domain or control domain, is created when the *logical domains* manager is installed. If installed on an existing server that is not already configured for logical domains, the current Operating System automatically gets promoted to primary domain status. The primary domain runs an



Oracle Solaris kernel and is responsible for the creation and management of all other domains. It is also responsible for providing access to virtualized hardware resources.

On systems running Oracle VM Server for SPARC, aside from the concepts of a "management domain" and of "guest domains", similar to "user domains", there are a variety of other domain types that can run alongside the management domain. For instance, it is possible to set up multiple "[service domains](#)" that can act as network switches and virtual disk servers. Not all of these domain types are configurable using the Oracle VM Manager. Most importantly, it is possible to add a secondary service domain to your servers that can take over disk and network I/O in the case where the primary domain must be restarted. This allows running guest virtual machines to continue to access these resources even while the primary domain is unavailable. This topic is discussed in more detail within [Configuring a Secondary Service Domain on SPARC](#) in the [Oracle VM Administrator's Guide](#).

2.2.2.2 Domains (domU)

Guest operating systems each have their own management domain called a "user domain", abbreviated to "domU". These domains are unprivileged domains with no direct access to the hardware or device drivers. Each *domU* is started alongside *dom0* running on Oracle VM Server.

Note

Under Oracle VM Server for SPARC these domains are usually referred to as "guest domains".

2.2.3 What is Oracle VM Agent?

The Oracle VM Agent is a daemon that runs within *dom0* on each Oracle VM Server instance. Its primary role is to facilitate communication between Oracle VM Server and Oracle VM Manager. The daemon listens for connections from Oracle VM Manager on the TCP port 8899, and implements a messaging facility that allows Oracle VM Manager to connect to each Oracle VM Server instance and exchange information required for the efficient running of the entire Oracle VM infrastructure.

Oracle VM Agent is responsible for carrying out all of the configuration changes required on an Oracle VM Server instance, in accordance with the messages that are sent to it by Oracle VM Manager. This means that when a networking change is implemented in Oracle VM Manager, it is the Oracle VM Agent that reconfigures the server to cater for the change. Equally, if new storage is discovered and presented to a server in Oracle VM Manager, it is the Oracle VM Agent that handles the actual mount process required on the server.

Oracle VM Agent is also responsible for starting and stopping virtual machines as required by Oracle VM Manager. For this reason, the actual implementation of Oracle VM Agent differs significantly between different hardware platforms, even though actions within Oracle VM Manager are consistent across platforms.

For security reasons, the Oracle VM Agent must authenticate any system attempting to connect to it on port 8899. When an Oracle VM Server is not configured for any particular Oracle VM Manager it is in an unowned state. In this state, an Oracle VM Manager must take ownership of the server before it is able to communicate with the Oracle VM Agent. During the process where the Oracle VM Manager instance takes ownership of a server, the Oracle VM Manager authenticates using a password configured for the Oracle VM Agent. This password is exchanged over a connection that is secured using an SSL certificate. Once the Oracle VM Agent has authenticated Oracle VM Manager, an SSL key-certificate pair is set up to authenticate and encrypt all future communications between that Oracle VM Manager instance and the Oracle VM Agent. At this point, no other Oracle VM Manager instance or application can take control of the Oracle VM Server via the Oracle VM Agent. If you wish to allow another Oracle VM Manager instance to take ownership of the server, the original Oracle VM Manager instance must release ownership first.

There are also times when Oracle VM Agent must initiate a connection to Oracle VM Manager to signal an event or to provide statistical information. This is achieved using the Web Services API exposed by Oracle VM Manager via HTTPS on TCP port 7002.

Oracle VM Agent also maintains its own log files on the Oracle VM Server that can be used for debugging issues on a particular server instance or for auditing purposes.

Oracle VM Agent is discussed in more detail in [Chapter 6, Understanding Server Pools and Oracle VM Servers](#).

2.2.4 What are Server Pools?

A [server pool](#) is a required entity in Oracle VM, even if it contains a single Oracle VM Server. In practice, several Oracle VM Servers will form a server pool, and an Oracle VM environment may contain one or several server pools. Server pools are typically clustered, although an unclustered server pool is also possible.

Server pools have shared access to storage repositories and exchange and store vital cluster information in the server pool file system. In a server pool, a [Master server](#) is elected, which is responsible for centralized communication with the Oracle VM Manager. If necessary, any other member of the server pool can take over the Master role. Even in case of server failure, the server pool remains accessible at its virtual IP address, which all servers in the pool share. Since server pools have shared access to storage repositories, live migration of virtual machines is possible for load balancing or for scheduled maintenance, so that a virtual machine can be moved from one Oracle VM Server to another without an interruption of service.

Within a clustered server pool, virtual machines have high availability or HA. The clustering technology used within Oracle VM can take care of monitoring the status of all of the Oracle VM Servers belonging to the server pool. If a pool member disappears for whatever reason, its virtual machines can be recovered and brought back up on another Oracle VM Server because all necessary resources are available on shared storage.

A detailed explanation of the purpose of a server pool and the technology used within one is provided in [Chapter 6, Understanding Server Pools and Oracle VM Servers](#).

[Prev](#)

[Up](#)

[Next](#)

2.1 What is the Oracle VM Architecture?

[Home](#)

2.3 What is Oracle VM Manager?

 Oracle logo

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. [Legal Notices](#)



https://web.archive.org/web/20141025120741/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-ovm-manager-overview.html



2.3 What is Oracle VM Manager?

[2.3.1 Oracle VM Manager Server and API](#)

[2.3.2 Oracle VM Database Repository](#)

[2.3.3 Oracle VM Manager Web-Based User Interface](#)

[2.3.4 Oracle VM Manager Command Line Interface](#)

[2.3.5 Oracle VM Utilities](#)

Management for the Oracle VM environment is provided by *Oracle VM Manager*, a transaction-based framework that also includes an integrated database, a web-based management user interface and a command line interface. Since Oracle VM can continue to function normally even during periods of downtime for the Oracle VM Manager, high-availability and redundancy are not generally required for Oracle VM Manager.

Since Oracle VM Manager is a multi-user application with a variety of different interfaces that allow components to be configured and managed, it is possible that two users may attempt to perform a configuration change on the same element at the same time. In this situation, Oracle VM Manager processes the first configuration change that reaches its queue, and temporarily locks the element for any other changes. The second configuration change fails because the information that it has is stale. Usually an error is returned for the configuration change that fails. This error notifies the user that the object was locked while fetching. To work around this problem, the user that receives the error must refresh the view of the object that was being updated and resubmit the changes.

When describing the architecture of Oracle VM Manager, it is best to understand that it is composed of various inter-related components. The fundamental components consist of the Oracle VM Manager Server and the Oracle VM Manager Database. All other Oracle VM Manager components interface with the Oracle VM Manager Server via an API.

2.3.1 Oracle VM Manager Server and API

The Oracle VM Manager Server is a Java-based Oracle *WebLogic* 12c application that is usually installed on a dedicated system. On x86 platforms, Oracle VM Manager is installed on Oracle Linux.

Oracle VM Manager Server is sometimes referred to as the Oracle VM Manager core. It is used to provide a means to configure, monitor and manage the Oracle VM environment. It does this largely by communicating directly with the Oracle VM Agent on each Oracle VM Server in the environment. It also stores all configuration information and event data within the Oracle VM Manager database.

Oracle VM Manager Server provides an API that allows other components to interface with it. As of Oracle VM Release 3.3.1, this is entirely implemented as a Web Services API, providing both SOAP and REST interfaces to the Oracle VM Manager. The Oracle VM Manager Web Interface and Oracle VM Manager Command Line Interface both make use of this API to interact directly with Oracle VM Manager.

The Web Services API exposed by Oracle VM Manager, makes it possible to programmatically set up and manage an Oracle VM environment. This API is introduced in considerable detail in the *Oracle VM Web Services API Developer's Guide*.

2.3.2 Oracle VM Database Repository

The Oracle VM Manager database is used by the Oracle VM Manager Server to store configuration information and event data. The actual database management server is a MySQL Enterprise database that is bundled with the Oracle VM Manager installer.

The MySQL database is set up using bundled database software included and licensed for use with Oracle VM Manager. The database management server is installed on the same system as Oracle VM Manager and is intended only for use by this application. Particular configuration parameters specific to the requirements of Oracle VM Manager are applied to this installation.

Details on the installation of the database are covered in the *Oracle VM Installation and Upgrade Guide*, while database backups are explained in more detail in *Oracle VM Administrator's Guide*.

2.3.3 Oracle VM Manager Web-Based User Interface

The Oracle VM Manager Web Interface is a web-based application built on top of Oracle WebLogic 12c and Application Development Framework (ADF). It interfaces with Oracle VM Manager Server using the Web Services API. The Oracle VM Manager Web Interface is installed on the same system as Oracle VM Manager Server and shares the same Oracle WebLogic Server space as the Oracle VM Manager core application, but is a distinct application. By default, the graphical user interface is accessed over HTTPS on the TCP port 7002.

The Oracle VM Manager Web Interface, its configuration and usage are described in detail in the *Oracle VM Manager User's Guide*.

2.3.4 Oracle VM Manager Command Line Interface

The Oracle VM Manager Command Line Interface is a Java-based daemon that runs on the system where Oracle VM Manager is installed, and interfaces with the Oracle VM Manager Server using the Web Services API. Access to the CLI is provided via an SSH connection available on a customizable TCP port, set to TCP 10000 by default. Any SSH client can be used to access the CLI, which provides its own shell and supports SSH key-based authentication.

The Oracle VM Manager Command Line Interface enables administrators to manage the entire Oracle VM environment from the shell, and facilitates programmatic setup and control of an environment through the use of simple shell or Expect scripts. Every action that is available via the Oracle VM Manager Web Interface is also available via the CLI.

As an independent component of the Oracle VM Manager suite of applications, the Oracle VM Manager Command Line Interface can be disabled if it is not required within an environment.

Full usage of the Oracle VM Manager Command Line Interface, including syntax and examples, is covered in the *Oracle VM Manager Command Line Interface User's Guide*.

2.3.5 Oracle VM Utilities

The Oracle VM Utilities are a set of largely unsupported Java applications designed to allow administrators to perform actions against Oracle VM Manager from the command line that may not be easily achieved using the officially provided interfaces. In general, most of the tools provided with the Oracle VM Utilities have been superseded by the Oracle VM Manager Command Line Interface, however some tools such as Oracle VM Hostd For Metrics Messaging provide facilities that are not otherwise available.

The Oracle VM Utilities and other assorted tools that assist with the configuration and management of Oracle VM Manager are discussed in the *Oracle VM Administrator's Guide*.





https://web.archive.org/web/20141012224904/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-vm.html



Chapter 7 Understanding Virtual Machines

Table of Contents

[7.1 What are Virtualization Modes or Domain Types?](#)
[7.2 How is a Guest OS Installed on a Virtual Machine?](#)
[7.3 Where are Virtual Machine Resources Stored?](#)
[7.4 What Networking is Available for Virtual Machines?](#)
[7.5 How are Virtual Machines Created?](#)
[7.6 How does Cloning Work?](#)
[7.7 How Can a Virtual Machine be Moved or Migrated?](#)
[7.8 Editing Virtual Machine Parameters](#)
[7.9 Sending Messages to Virtual Machines](#)
[7.10 Accessing the Virtual Machine Console](#)
[7.11 How Can I Protect Virtual Machines?](#)
[7.12 How are Virtual CPUs Allocated to Virtual Machines](#)
[7.13 How are Huge Pages Enabled for Virtual Machines?](#)
[7.14 Setting Hard Partitioning for Virtual Machine CPUs](#)
[7.15 How are Virtual Machines Backed Up?](#)

The purpose of Oracle VM is to provide an infrastructure that virtual machines are able to run on, and which is easy to manage, configure and maintain. In this chapter we explore the different types of virtual machines that are supported; how virtual machines work within an Oracle VM environment; how virtual machines can be created, moved and migrated; and how to provision resources for a virtual machine.

The terms *domain*, *guest* and virtual machine are often used interchangeably, but they have subtle differences. A domain is a configurable set of resources, including memory, virtual CPUs, network devices and disk devices, in which virtual machines run. A domain is granted virtual resources and can be started, stopped and restarted independently of other domains and of the host server itself. A guest is a virtualized operating system running within a domain. A guest operating system may be *paravirtualized*, *hardware virtualized*, or hardware virtualized with paravirtualized drivers. A description of these different virtualization modes is provided in [Section 7.1, “What are Virtualization Modes or Domain Types?”](#).

Multiple guests can run on the same [Oracle VM Server](#). A virtual machine is a guest operating system and its associated application software. For the sake of simplicity, we use the term virtual machine to encompass domain, guest and virtual machine. They are synonymous with each other and may be used interchangeably.

An operating system installed in a virtual machine is known as a guest operating system. Oracle VM supports a variety of guest operating systems including Linux, Oracle Solaris and Microsoft Windows™. For a list of the supported guest operating systems, see the [Oracle VM Release Notes](#). Guest operating system installation is described in [Section 7.2, “How is a Guest OS Installed on a Virtual Machine?”](#).

You can use Oracle VM Manager to create virtual machines using:

- ISO files in a repository (hardware virtualized only).
- Mounted ISO files on an NFS, HTTP or FTP server (paravirtualized only).
- Virtual machine templates (by cloning a template).
- Existing virtual machine (by cloning the virtual machine).
- Virtual machine assemblies.

Virtual machines require most of these installation resources to be in a storage repository, managed by Oracle VM Manager, with the exception of mounted ISO files for paravirtualized guests. See [Section 7.3, “Where are Virtual Machine Resources Stored?”](#) for information on how these resources are stored within a repository.

When you create a virtual machine that requires network connectivity, or a paravirtualized machine which requires network connectivity to perform the operating system install, you generate some virtual network interfaces that can be bridged to a network defined on the Oracle VM Server. More information on provisioning networking for your virtual machines is provided in [Section 7.4, “What Networking is Available for Virtual Machines?”](#).

In cases where you require multiple virtual machines that are identical in make up, you might consider cloning a virtual machine or template. This process is described in more detail in [Section 7.6, “How does Cloning Work?”](#). Equally, you may need to move a virtual machine from one server or server pool to another, this process is known as migration and is discussed in [Section 7.7, “How Can a Virtual Machine be Moved or Migrated?”](#)

Oracle VM provides a messaging facility that allows messages to be sent backwards and forwards between Oracle VM Manager and a guest operating system. This makes it possible to automate configuration actions when a virtual machine boots, to obtain information about the internal status of a virtual machine, and to trigger events within a virtual machine based on changes within the Oracle VM environment. This messaging facility is described in some detail in [Section 7.9, “Sending Messages to Virtual Machines”](#).

In some cases, usually due to software licensing requirements, you may require that virtual machines are *hard partitioned* using CPU pinning. This feature allows you to pin a virtual machine to a specific physical CPU to prevent it from running on alternate systems. This is an advanced feature and is described in [Section 7.14, “Setting Hard Partitioning for Virtual Machine CPUs”](#).



https://web.archive.org/web/20141026070305/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-vm-modes.html



7.1 What are Virtualization Modes or Domain Types?

Virtual machines may run in one of two main modes, *paravirtualized (PVM)* or *hardware virtualized machine (HVM)*. In paravirtualized mode, the kernel of the guest operating system is recompiled to be made aware of the virtual environment. This allows the paravirtualized guest to run at near native speed, since most memory, disk and network accesses are optimized for maximum performance.

If support for hardware virtualization is available (either Intel® VT or AMD-V™), the guest operating system may run completely unmodified. This hardware virtualized machine is carefully monitored and trapped by *Oracle VM Server* when any instruction is executed which would violate the isolation with other guests or dom0. In the current implementation, there may be performance penalty for certain types of guest and access types, but hardware virtualization also allows many Microsoft Windows operating systems and legacy operating systems to run unmodified.

The third virtualization mode is a hardware virtualized machine with paravirtualized drivers (PVHVM). This mode is identical to a hardware virtualized machine, but with additional paravirtualized drivers installed in the guest's operating system to improve virtual machine performance.

Oracle recommends you create paravirtualized virtual machines if possible, as the performance of a paravirtualized virtual machine is often superior to that of a hardware virtualized machine.

There are a number of virtual machine virtualization modes, or *domain* types, as shown in [Table 7.1, "Domain Types"](#). When you create a virtual machine you must decide which mode to use.

Table 7.1 Domain Types

Domain Type	Description
Hardware virtualized machine (HVM)	Hardware virtualization, or fully virtualized. When you create an HVM guest, you must supply an ISO file in a repository from which to create the virtual machine. See Section 7.3, "Where are Virtual Machine Resources Stored?" for information on storing an ISO file in a repository. To create HVM guests, you may need to activate the hardware virtualization in the BIOS of the server on which you install the Oracle VM Server.
Hardware virtualized, with paravirtualized drivers (PVHVM)	Identical to HVM, but with additional paravirtualized drivers for improved performance of the virtual machine. See Converting to Paravirtualized Guests or Installing Paravirtualized Drivers for more information about using paravirtualized drivers. This Domain Type is used to run Microsoft Windows guest operating systems with an acceptable performance level.
Paravirtualized (PVM)	When you create a PVM guest, you must supply a location for the mounted ISO file from which to create the virtual machine. Before you create the virtual machine using the paravirtualized method, mount the ISO file on an NFS share, or HTTP or FTP server. You supply the location of the mounted ISO file during the creation process. For information on creating a mounted ISO file, see Section 7.2, "How is a Guest OS Installed on a Virtual Machine?" .
Oracle VM Server for SPARC (OVM/SPARC)	Since virtualization on SPARC platforms is significantly different to that achieved within x86 environments, Oracle VM distinguishes a separate virtualization mode specifically to indicate that the SPARC hypervisor is to be used. When creating a virtual machine that is to run in a server pool consisting of Oracle VM Servers for SPARC, this mode is used during configuration.

Note

If you want to create a PVHVM or PVM, make sure all disks the virtual machine is configured to use are paravirtual devices. An explanation and description of the steps to properly configure paravirtual disks is discussed in [Paravirtualized Guest Disk Devices are Not Recognized](#) in the *Oracle VM Administrator's Guide*.

Converting to Paravirtualized Guests or Installing Paravirtualized Drivers

For optimized performance, you can install *paravirtualized* drivers on *hardware virtualized machines*. Paravirtual drivers are optimized and improve the performance of the operating system in a virtual machine. These drivers enable high performance throughput of I/O operations in *guest* operating systems on top of the *Oracle VM Server* hosts.

Creating hardware virtualized machine machines may require that you install paravirtual drivers for your hardware on the guest operating system.

The Oracle Solaris 10 and Oracle Solaris 11 operating system runs as a hardware virtual machine (HVM), which requires HVM support (Intel® VT or AMD-V™) on the underlying hardware platform. By default, Oracle Solaris 10 or Oracle Solaris 11 operating system already has the required paravirtualized drivers installed as part of the operating system.

You can continue using HVM guest, but leverage the paravirtualized I/O drivers. For more information, see the Oracle Support document 757719.1 titled *Comparison of Guest Virtualization Modes; HVM, PVM and PVHVM. Configuration, Mode Validation and Conversion to PVHVM* on the My Oracle Support web site at:

<https://support.oracle.com/epmos/faces/DocumentDisplay?id=757719.1>.

To install the paravirtual drivers for Microsoft Windows operating systems (Oracle VM Paravirtual Drivers for Microsoft Windows), see the *Oracle VM Paravirtual Drivers Installation Guide*.

https://web.archive.org/web/20141026065837/http://docs.oracle.com/cd/E50245_01/E50249/html/vmcon-repo.html



Copyright © 2014, Oracle and/or its affiliates. All rights reserved. [Legal Notices](#)

JURAT ATTACHMENT

A notary public or other officer completing this certificate verifies only the identity of the individual who signed the document to which this certificate is attached, and not the truthfulness, accuracy, or validity of that document.

STATE OF Texas }

COUNTY OF Harris }

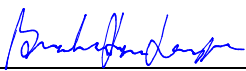
The foregoing instrument was subscribed and sworn before me this date of
11/14/2025 by Mina Ching

This notarial act was an online notarization.



(Notary Seal)

Document Notarized using a Live Audio-Video Connection

Notary's Signature 

Registration No.: 135513465

Commission Expiration Date: April 29, 2029

