



Red Hat Reference Architecture Series

Red Hat Cloud Foundations

Deploying Private IaaS Clouds

Scott Collier, RHCA
Principal Software Engineer

Version 2.0
April 2011

refarch-feedback@redhat.com





1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

The following terms used in this publication are trademarks of other companies as follows:

- Linux is a registered trademark of Linus Torvalds.
- Red Hat, Red Hat Enterprise Virtualization, JBoss, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.
- Microsoft, Hyper-V, Windows are U.S. registered trademarks of Microsoft Corporation.
- UNIX is a registered trademark of The Open Group.
- Intel, the Intel logo and Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- IBM BladeCenter, WebSphere are trademarks of International Business Machines Corporation
- VMware is a registered trademark of VMware Incorporated
- Amazon and Amazon EC2 are registered Amazon trademarks
-

All other trademarks referenced herein are the property of their respective owners.

© 2011 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Distribution of modified versions of this document is prohibited without the explicit permission of Red Hat Inc.

Distribution of this work or derivative of this work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from Red Hat Inc.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

| | |
|--------------------------------------------------------------------------------------|--------------------|
| 1 Executive Summary..... | 6 |
| 2 Cloud Computing Standards..... | 8 |
| 2.1 Cloud Provider – Service Models..... | 9 |
| 3 Red Hat and Cloud Computing..... | 13 |
| 3.1 A Phased Approach to Cloud Computing..... | 13 |
| 3.2 Unlocking the Value of the Cloud..... | 14 |
| 3.3 Redefining the Cloud..... | 15 |
| 4 Red Hat Cloud Foundations Components..... | 16 |
| 4.1 Red Hat Enterprise Linux 5..... | 16 |
| 4.2 Red Hat Enterprise Linux 6..... | 16 |
| 4.3 Red Hat Enterprise Virtualization (RHEV) for Servers..... | 17 |
| 4.4 Red Hat Network (RHN) Satellite..... | 18 |
| 4.5 JBoss Enterprise Middleware..... | 18 |
| 4.5.1 JBoss Enterprise Application Platform (EAP)..... | 19 |
| 4.5.2 JBoss Operations Network (JON)..... | 20 |
| 4.5.3 Red Hat Enterprise MRG Grid..... | 21 |
| 5 RHCF Proof of Concept Configuration | 22 |
| 5.1 Operating Systems..... | 24 |
| 5.1.1 Applications and Tools..... | 25 |
| 5.2 Hardware..... | 26 |
| 5.2.1 Servers..... | 26 |
| 5.2.2 Storage | 27 |
| 6 Deploying Red Hat Cloud Foundations – Infrastructure Services..... | 28 |
| 6.1 Overview..... | 28 |
| 6.2 Download Software..... | 29 |
| 6.2.1 Download Scripts..... | 30 |
| 6.3 Deploy mgmt1 and Configure..... | 31 |
| 6.3.1 Set up the Logical Volumes for the Virtual Machines..... | 35 |
| 6.4 Deploy Satellite Virtual Machine and Install Satellite..... | 35 |



| | |
|-----------------------------------------------------------------------------------------|-----------|
| 6.4.1 Deploy the Satellite Virtual Machine..... | 35 |
| 6.4.2 Configure Iptables | 37 |
| 6.5 Create Kickstart Profiles and Activation Keys..... | 39 |
| 6.5.1 Create Activation Keys..... | 39 |
| 6.5.2 Create Kickstart Profiles..... | 40 |
| 6.6 Deploy DHCP / DNS Virtual Machine..... | 43 |
| 6.6.1 Configure DHCP..... | 43 |
| 6.6.2 Configure DNS..... | 44 |
| 6.7 Deploy Red Hat Enterprise Virtualization Platform..... | 45 |
| 6.7.1 Deploy RHEV-M Virtual Machine..... | 45 |
| 6.7.2 Install Red Hat Enterprise Virtualization Manager Software..... | 54 |
| 6.7.2.1 Prepare the Microsoft Windows 2008 R2 server for RHEV..... | 54 |
| 6.7.2.2 Disable “Internet Explorer Enhanced Security Configuration”..... | 55 |
| 6.7.2.3 Install Red Hat Enterprise Virtualization Manager..... | 55 |
| 6.8 Deploy the Red Hat Enterprise Virtualization Hypervisor..... | 55 |
| 6.8.1 Prepare the Environment to Deploy a RHEV Hypervisor | 55 |
| 6.8.2 Deploy the RHEV Hypervisor..... | 57 |
| 6.8.3 Approve RHEV Hypervisor..... | 57 |
| 6.9 Deploy the RHEL KVM Hypervisor..... | 57 |
| 6.9.1 Deploy Red Hat Enterprise Linux 5.6 Server with KVM | 57 |
| 6.10 Add the RHEV KVM Hypervisor in RHEV-M..... | 57 |
| 6.11 Configure RHEV Datacenter, Cluster, and Storage Domain | 59 |
| 6.12 Configure ISO Domain | 61 |
| 7 Deploy Tenant Virtual Machines..... | 65 |
| 7.1 Overview..... | 65 |
| 7.2 Deploy Red Hat Enterprise Linux 5.6 on RHEV VM Via PXE..... | 65 |
| 7.3 Deploy Red Hat Enterprise Linux 5.6 on RHEV VM Via ISO..... | 66 |
| 7.4 Deploy Red Hat Enterprise Linux 5.6 on RHEV via Template | 67 |
| 7.5 Deploy Microsoft Windows 2008 R2 VM Via ISO..... | 67 |
| 7.5.1 Sysprep Microsoft Windows VM and Create a Template..... | 69 |
| 8 Configure High Availability Environment..... | 70 |
| 8.1 Install Luci Virtual Machine..... | 70 |
| 8.2 Install Cluster nodes..... | 72 |
| 8.2.1 Install mgmt2, Configure Networking on mgmt2, Register mgmt1 with Satellite | 72 |
| 8.2.2 Set up the Cluster..... | 74 |
| 8.3 Cluster Virtual Machines..... | 77 |

| | |
|----------------------------------------------------------------------|------------|
| <u>9 Deploy and Scale Applications.....</u> | <u>79</u> |
| <u>9.1 Deploy Java Application.....</u> | <u>79</u> |
| <u>9.1.1 Configure GPG and Sign the javaApp package.....</u> | <u>80</u> |
| <u>9.1.2 Set up Software Channel on Satellite Server</u> | <u>81</u> |
| <u>9.1.3 Upload Application</u> | <u>81</u> |
| <u>9.1.4 Create RHN Activation Key for Custom Channel.....</u> | <u>82</u> |
| <u>9.1.5 Create a New Kickstart Profile.....</u> | <u>82</u> |
| <u>9.1.6 Deploy Virtual Machine with javaApp via PXE.....</u> | <u>83</u> |
| <u>9.1.7 Create a Template from the javaApp Virtual Machine.....</u> | <u>84</u> |
| <u>9.1.8 Scale the javaApp Virtual Machine.....</u> | <u>84</u> |
| <u>9.2 Deploy and Scale JBoss EAP Application</u> | <u>84</u> |
| <u>9.2.1 Provision JBoss ON Server on RHEV.....</u> | <u>86</u> |
| <u>9.3 Deploy JBoss Enterprise Application Platform.....</u> | <u>88</u> |
| <u>9.3.1 Scale JBoss Deployment.....</u> | <u>91</u> |
| <u>9.4 Deploy and Scale Applications – MRG Manager.....</u> | <u>93</u> |
| <u>10 Summary.....</u> | <u>103</u> |
| <u>11 Appendix A.....</u> | <u>105</u> |
| <u>11.1 Configuration Files.....</u> | <u>105</u> |
| <u>12 Appendix B Scripts.....</u> | <u>111</u> |



1 Executive Summary

Cloud computing is quickly becoming the platform of choice for users and businesses that want to reduce operating expenses and be able to scale resources rapidly. There are several other advantages of moving resources to the cloud such as eased automation, flexibility, mobility, resiliency, and redundancy.

Even though cloud computing is in the early stages, there are different types of cloud solutions available to businesses today. Private clouds allow businesses to take advantage of cloud technologies while remaining on a private network. Public clouds allow businesses to leverage pay-as-you-go external resources. In addition to that, using a public cloud reduces the need to evaluate new hardware, deploy the infrastructure or support it. Hybrid clouds allow the best of both public and private cloud computing models because it allows reallocation of traffic to public clouds when it makes sense to do so.

Cloud computing provides several different services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These standards are defined by NIST (National Institute for Standards and Technology)¹ and are discussed in detail within this paper.

Red Hat Cloud Foundations (RHCF) provide the software infrastructure required to build public, private and hybrid clouds by utilizing open source technologies that are available today. Some of the key technologies that RHCF utilizes are Red Hat Enterprise Linux (RHEL), Red Hat Enterprise Virtualization (RHEV) which serves as the foundation of the virtualized hosts and Messaging, Realtime and Grid (MRG) which provides a framework for distributing workloads in a high speed, efficient manner. RHCF also utilizes Red Hat Network Satellite technologies to manage parts of the infrastructure and components of the JBoss suite as a middleware platform used for scaling applications. RHCF utilizes many other technologies that are going to be discussed in greater detail in the following sections.



Figure 1.1: CloudForms

Red Hat CloudForms, currently in beta, brings a wide range of additional capabilities to organizations building private and hybrid IaaS clouds. Red Hat CloudForms provides advanced features such as Self Service, Resource Abstraction and pooling, creation and management of images, and complete Application Life Cycle Management. These capabilities are additive to current products and can be added incrementally to Red Hat Cloud Foundations while fully preserving that investment. Red Hat Cloud Foundations is therefore a great way to get started with cloud computing today.

This paper takes all the required Red Hat Cloud Foundation technologies and presents best practices for tying them together. The entire cloud stack is covered from the Red Hat Enterprise Linux operating system to the applications that scale on top of it.



2 Cloud Computing Standards

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The following definitions have been proposed by NISTⁱⁱ

A **cloud consumer** can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

Resource Pooling serves multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify a location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

Elasticity enables the ability to automatically and quickly scale out resources and rapidly release those resources to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Cloud systems automatically control and optimize resources used by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). This model of Measured Service allows resource usage to be monitored, controlled, and reported providing transparency for both the provider and consumer of the used service.

Infrastructure as a Service (IaaS) is where the cloud consumer generally has broad freedom to choose the operating system and development environment to be hosted. Security provisions beyond the basic infrastructure are carried out mainly by the cloud consumer.

When using the IaaS model, the cloud provider maintains the storage, networking and the hosting environment for the virtual machines. This eliminates the need for cloud consumers to evaluate, maintain and support the hardware infrastructure.

Platform as a Service (PaaS) gives the Cloud Consumer control over applications and application environment settings of the platform. Security provisions are split between the cloud provider and the cloud consumer. This scenario would be beneficial for organizations that need to develop, test, deploy and manage the application life cycle.

Software as a Service (SaaS) abstracts the management and control of the underlying cloud infrastructure and individual applications, except for preference selections and limited administrative settings. Security provisions are carried out mainly by the cloud provider. An example usage model here might be a cloud consumer that simply needs access to cloud

resources for business operations. In this scenario, the cloud provider installs, manages, maintains and supports the software, the cloud consumer just uses it.

2.1 Cloud Provider – Service Models

A **public cloud** provides infrastructure that is made available to the general public or a large industry group and is owned by an organization providing cloud services. Some of the services that public cloud providers host are storage, email, social applications, business applications and web application platforms. In the public cloud model, the cloud consumer does not maintain any of the infrastructure required to run the services. See **Figure 2.1: Public Cloud** for an example public cloud model.

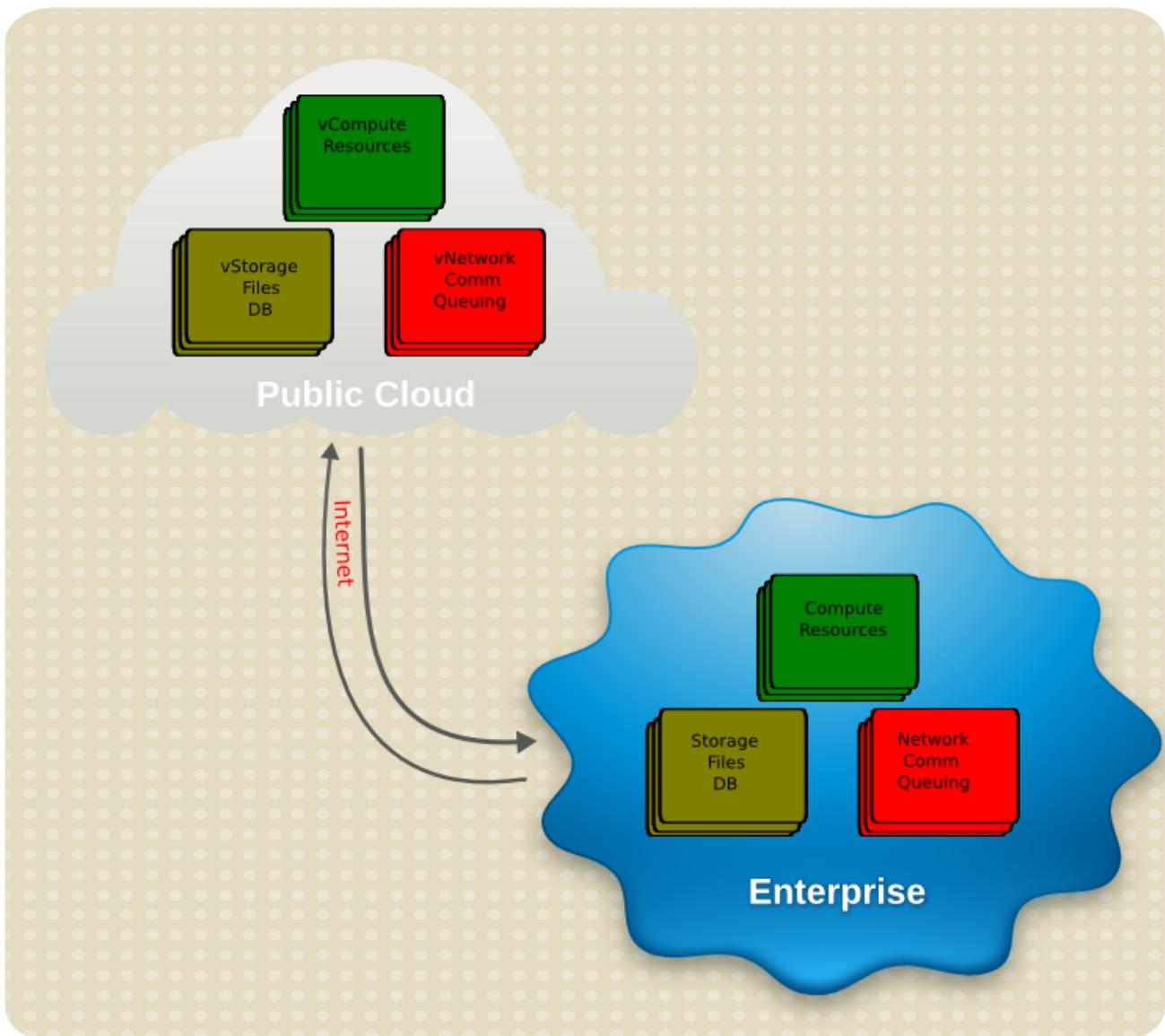


Figure 2.1: Public Cloud



A **private cloud** provides the hardware and software infrastructure that is operated solely for an organization. It may be managed by the organization or a third-party and may exist on premise or off-premises. By utilizing a private cloud service model an organization is able to take advantage of the elasticity and cost effectiveness of cloud technologies as well as maintain complete control of the infrastructure. See **Figure 2.2: Private Cloud** for an example of a private cloud model.

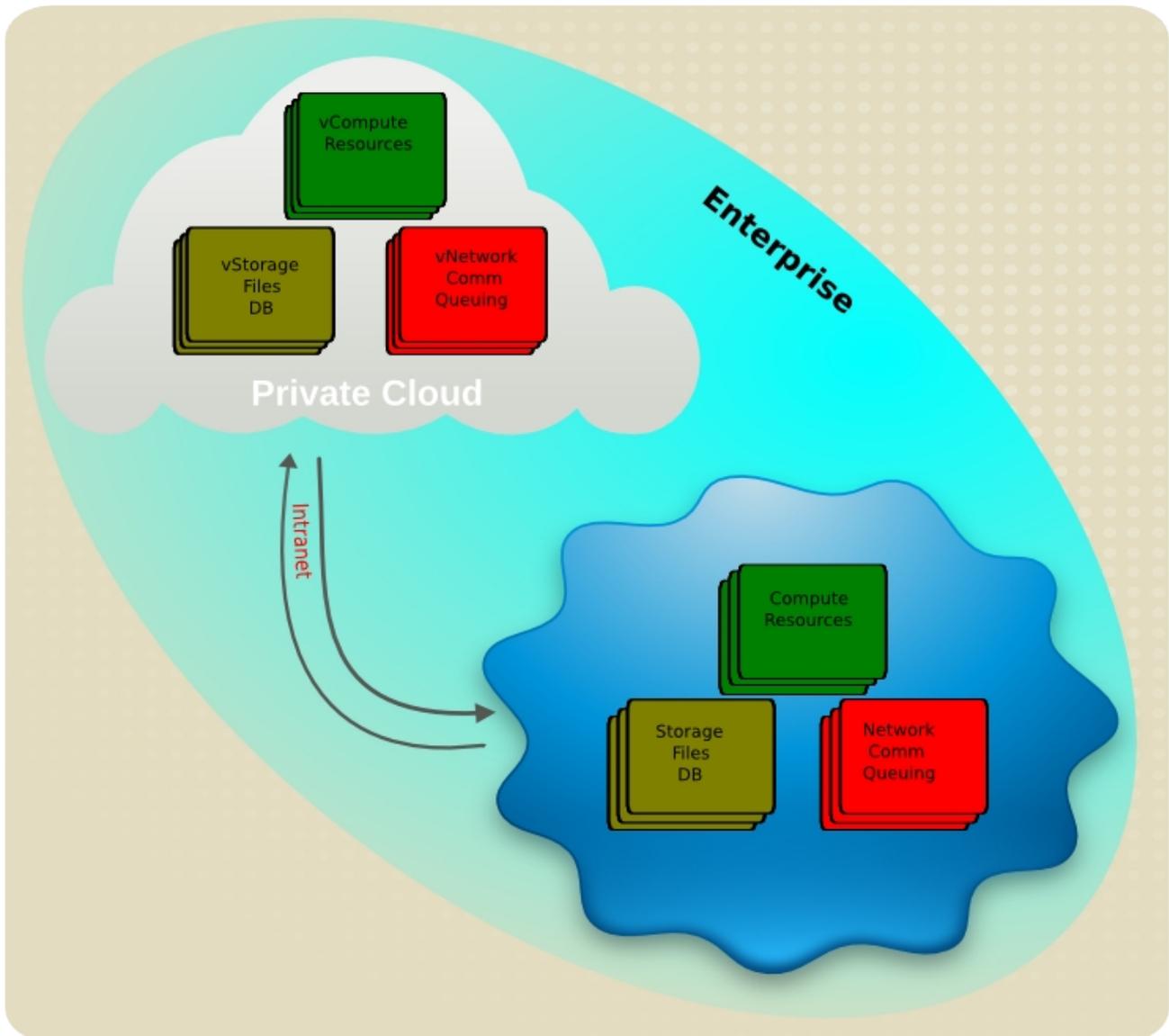


Figure 2.2: Private Cloud

A **hybrid cloud** provides infrastructure that is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting or load balancing between clouds). One possible usage model for a hybrid cloud would be to host all services on the private cloud until those resources became exhausted. At that point, the resources on the private cloud could scale to the third-party public cloud provider until the high utilization reduces to a point where the resources could be brought back in-house. See **Figure 2.3: Hybrid Cloud** for a hybrid cloud example.

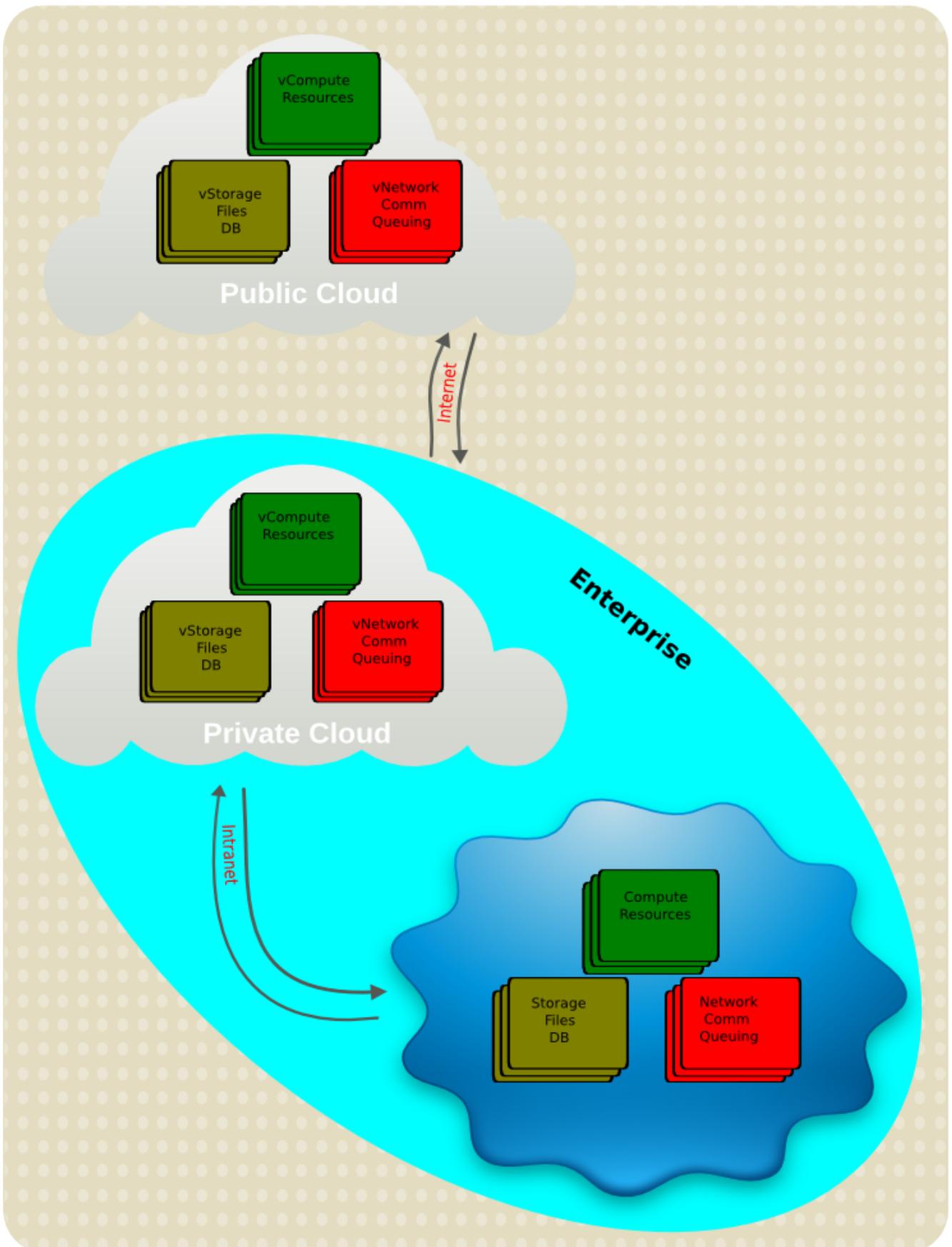


Figure 2.3: Hybrid Cloud



A **community cloud** is an infrastructure that is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organization or a third-party and may exist on or off premise.

3 Red Hat and Cloud Computing

3.1 A Phased Approach to Cloud Computing

While virtualization is an important underlying technology with regards to cloud computing, it is inaccurate to equate cloud computing with virtualization. See **Figure 3.1: Cloud Layers** for the different levels of abstraction addressed by virtualization and cloud computing respectively.

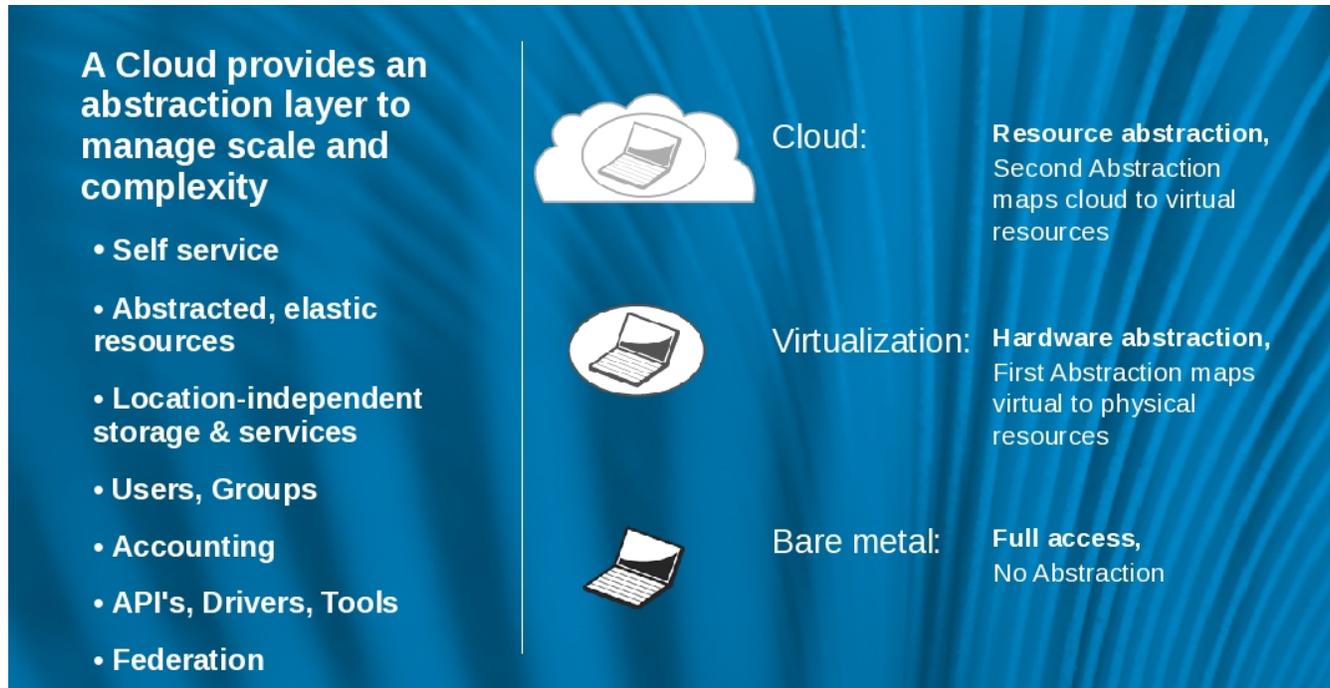


Figure 3.1: Cloud Layers



Figure 3.2: Phased Deployment illustrates a phased approach to technology adoption starting with server consolidation using virtualization, then automating large deployments of virtualization within an enterprise using private clouds, and finally extending private clouds to hybrid environments leveraging public clouds as a utility.

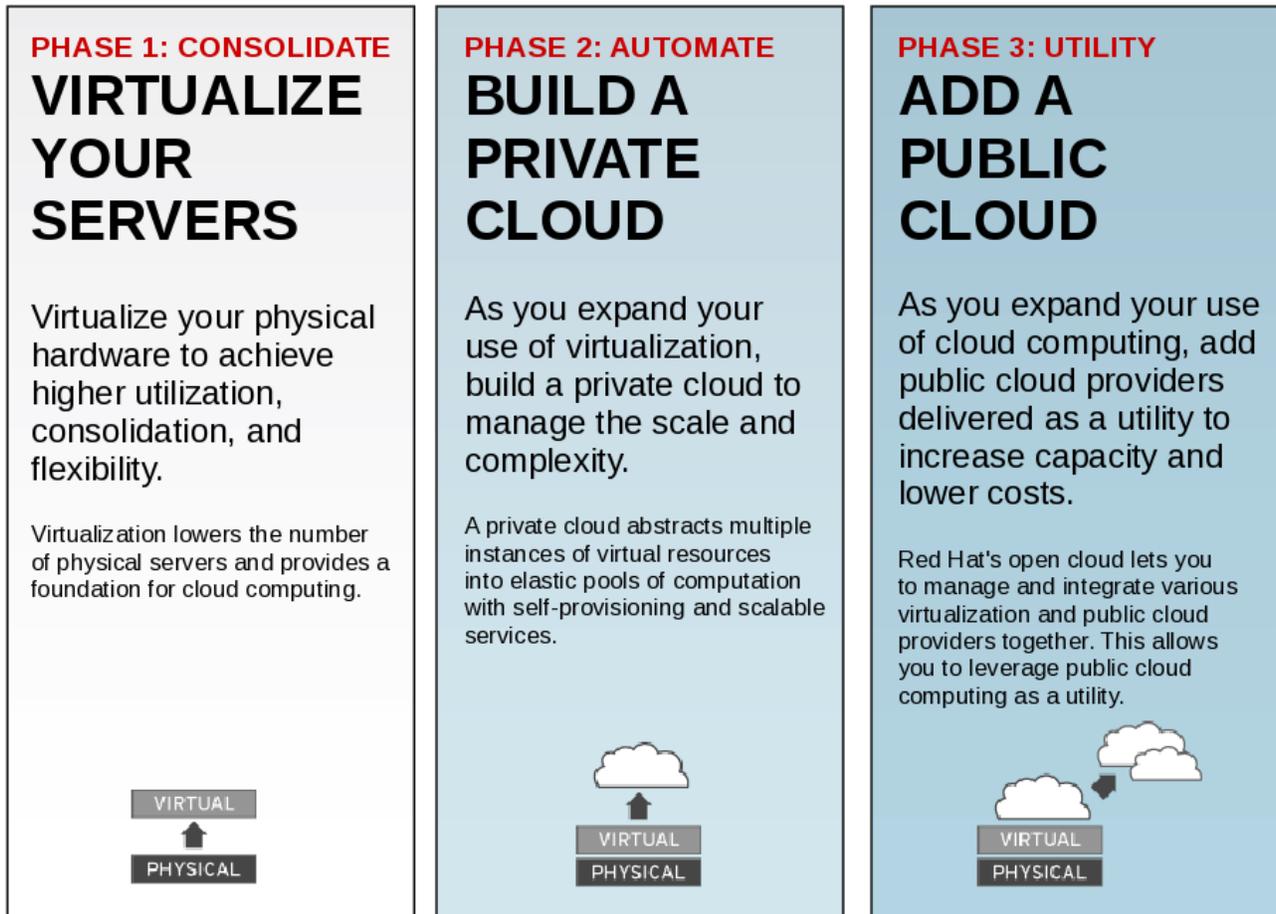


Figure 3.2: Phased Deployment

3.2 Unlocking the Value of the Cloud

Red Hat's approach does not lock an enterprise into one vendor's cloud stack, but instead offers a rich set of solutions for building a cloud. These can be used alone or in conjunction with components from third-party vendors to create the optimal cloud to meet unique needs.

Cloud computing is one of the most important shifts in information technology to occur in decades. It has the potential to improve the agility of organizations by allowing them to:

1. Enhance their ability to respond to opportunities
2. Bond more tightly with customers and partner
3. Reduce the cost to acquire and use IT in ways never before possible

Red Hat is proud to be a leader in delivering the infrastructure necessary for reliable, agile, and cost-effective cloud computing. Red Hat's cloud vision is unlike that of any other IT vendor. Red Hat recognizes that IT infrastructure is composed of pieces from many different hardware and software vendors. Red Hat enables the use and management of these diverse assets as one cloud. Enabling cloud to be an evolution, not a revolution. Red Hat's vision spans the entire range of cloud models:

- Building an internal Infrastructure as a Service (IaaS) cloud, or seamlessly using a third-party's cloud
- Creating new Linux, LAMP, or Java applications online, as a Platform as a Service (PaaS)
- Providing the easiest path to migrating applications to attractive Software as a Service (SaaS) models

Red Hat's open source approach to cloud computing protects and manages existing and diverse investments as one cloud -- whether Red Hat Enterprise Linux or Microsoft Windows, Red Hat Enterprise Virtualization, VMware or Microsoft Hyper-V, Amazon EC2 or another vendor's IaaS, .Net or Java, JBoss or WebSphere, x86 or mainframe.

Red Hat's approach to open source cloud computing also prevents vendor specific lock-in. For example, when the third-party cloud providers are abstracted, the consumer can choose where to run the virtual machines or where to scale the applications based on cost and reliability.

3.3 Redefining the Cloud

Cloud computing is the first major market wave where open source technologies are built-in from the beginning, powering the vast majority of early clouds. Products that make up Red Hat's cloud infrastructure include:

- Red Hat Enterprise Virtualization
- Red Hat Enterprise Linux
- Red Hat Network Satellite
- Red Hat Enterprise MRG Grid
- JBoss Enterprise Middleware

In addition, Red Hat is leveraging open source work and investing in several open source projects related to cloud computing. As these projects mature, after they undergo rigorous testing, tuning, and hardening, the ideas from many of these projects may be incorporated into future versions of the Red Hat cloud infrastructure. These projects include:

- Cobbler - Installation server for rapid set up of network installation environment
- Condor - Batch system managing millions of machines worldwide
- Hail - Umbrella cloud computing project for cloud services
- Libvirt - Common, generic, and scalable layer to securely manage domains on a node
- Spice - Open remote computing solutions for interaction with virtualized desktop devices



4 Red Hat Cloud Foundations Components

4.1 Red Hat Enterprise Linux 5

Red Hat Enterprise Linux is the world's leading open source application platform. On one certified platform, RHEL offers a choice of:

- Applications - Thousands of certified ISV applications
- Deployment - Including standalone or virtual servers, cloud computing, and software appliances
- Hardware - Wide range of platforms from the world's leading hardware vendors

Red Hat released the sixth update to RHEL 5: Red Hat Enterprise Linux 5.6 in November 2010. RHEL 5.6 is designed to support newer processors and chipsets, I/O (iSCSI & iSNS) and multimedia, combined with numerous driver updates. The new platform leverages Red Hat's history in scalable performance with new levels of core counts, memory and I/O, offering users a very dense and scalable platform balanced for performance across many workload types.

Red Hat also continues to make enhancements to our virtualization platform. New to RHEL 5.6 is support for sVirt (SELinux virtualization), which enables Mandatory Access Control (MAC) profiles to be applied to guests, enhancing overall system security. The new hardware and protocols included in the latest release significantly improve network scaling by providing direct access from a guest to the network.

4.2 Red Hat Enterprise Linux 6

Red Hat Enterprise Linux 6, the latest release of Red Hat's trusted datacenter platform, delivers advances in application performance, scalability, and security. With Red Hat Enterprise Linux 6, physical, virtual and cloud computing resources can be deployed within the data center.

Reliability, availability, and security (RAS):

- More sockets, more cores, more threads and more memory
- RAS hardware-based hot add of CPUs and memory is enabled
- Memory pages with errors can be declared as "poisoned" and will be avoided

Filesystems:

- ext4 is the default filesystem and scales to 16TB
- XFS is available as an add-on and can scale to 100TB
- Fuse allows filesystems to run in user space allowing testing and development on newer fuse-based filesystems (such as cloud filesystems)

High Availability:

- The web interface based on Conga has been redesigned for added functionality and ease of use
- Unified logging and debugging simplifies administrative work
- Virtualized KVM guests can be run as managed services, which enables fail-over, including between physical and virtual hosts

Resource Management:

- Cgroups organize system tasks so that they can be tracked and so that other system services can control the resources that cgroup tasks may consume
- Cpuset applies CPU resource limits to cgroups, allowing processing performance to be allocated to tasks

There are many other feature enhancements to Red Hat Enterprise Linux 6. Please see the Red Hatⁱⁱⁱ website for more information.

4.3 Red Hat Enterprise Virtualization (RHEV) for Servers

Red Hat Enterprise Virtualization (RHEV) for Servers is an end-to-end virtualization solution that is designed to enable pervasive data center virtualization, and unlock unprecedented capital and operational efficiency. RHEV is the ideal platform on which to build an internal or private cloud of Red Hat Enterprise Linux or Microsoft Windows virtual machines. RHEV consists of the following two components:

- **Red Hat Enterprise Virtualization Manager (RHEV-M):** A feature-rich server virtualization management system that provides advanced capabilities for hosts and guests, including high availability, live migration, storage management, system scheduler, and more.
- **Red Hat Enterprise Virtualization Hypervisor:** A modern hypervisor based on Kernel-Based Virtual Machine (KVM) which can be deployed as either RHEV-H, a standalone bare metal hypervisor (included with Red Hat Enterprise Virtualization for Servers), or as Red Hat Enterprise Linux 5.4 and later (purchased separately) installed as a hypervisor.

Some key characteristics of RHEV are listed below:

Scalability:

- Host: Up to 512 cores, 1 TB RAM
- Guest/VM: Up to 16 vCPUs, 256 GB RAM

Advanced features:

- Memory page sharing, advanced scheduling capabilities, and more, inherited from the Red Hat Enterprise Linux kernel

Guest operating system support:

- Paravirtualized network and block drivers for highest performance
- Red Hat Enterprise Linux Guests (32-bit & 64-bit): Red Hat Enterprise Linux 3, 4 and 5



- Microsoft® Windows® Guests (32-bit & 64-bit): Microsoft Windows 2003 server, Microsoft Windows 2008 server, Microsoft Windows XP, SVVP, and WHQL certified.

Hardware support:

- All 64-bit x86 servers that support Intel VT or AMD-V technology and are certified for Red Hat Enterprise Linux 5 are certified for Red Hat Enterprise Virtualization.
- Red Hat Enterprise Virtualization supports NAS/NFS, Fibre Channel, and iSCSI storage topologies.

4.4 Red Hat Network (RHN) Satellite

All RHN^{iv} functionality is on the network, allowing much greater functionality and customization. The Satellite server connects with Red Hat over the public Internet to download new content and updates. This model also allows customers to take their Red Hat Network solution completely off-line if desired. Features include:

- Support for provisioning Red Hat Enterprise Linux 6
- Compliance features SELinux context deployment and reporting
- API layer allows the creation of scripts to automate functions or integrate with existing management applications.
- External repository synchronization
- Create staged environments (development, test, production) to select, manage and test content in a structured manner.
- The ability to remove duplicate profiles
- Access to advanced features in the Provisioning Module, such as bare metal PXE boot provisioning and integrated network install trees.
- Access to Red Hat Network Monitoring Module for tracking system and application performance.

RHN Satellite is Red Hat's on-premise systems management solution that provides software updates, configuration management, provisioning and monitoring across both physical and virtual Red Hat Enterprise Linux servers. It offers customers opportunities to gain enhanced performance, centralized control and higher scalability for their systems, while deployed on a management server located inside the customer's data center and firewall.

Red Hat released RHN Satellite 5.4 in July 2010, the latest version of the product which is used in this paper. This version offers opportunities for increased flexibility and faster provisioning setups for customers with the incorporation of open source Cobbler technology in its provisioning architecture.

4.5 JBoss Enterprise Middleware

The following JBoss^v Enterprise Middleware Development Tools, Deployment Platforms and Management Environments are available via subscriptions that deliver not only industry leading SLA-based production and development support, but also includes patches, updates, multi-year maintenance policies, and software assurance from Red Hat.

Development Tools:

- JBoss Developer Studio - PE (Portfolio Edition): Everything needed to develop, test and deploy rich web applications, enterprise applications and SOA services

Enterprise Platforms:

- JBoss Enterprise Application Platform: Everything needed to deploy, and host enterprise Java applications and services
- JBoss Enterprise Web Platform: A standards-based solution for light and rich Java web applications
- JBoss Enterprise Web Server: a single enterprise open source solution for large-scale websites and lightweight web applications
- JBoss Enterprise Portal Platform: Platform for building and deploying portals for personalized user interaction with enterprise applications and automated business processes
- JBoss Enterprise SOA Platform: A flexible, standards-based platform to integrate applications, SOA services, and business events as well as to automate business processes
- JBoss Enterprise BRMS: An open source business rules management system that enables easy business policy and rules development, access, and change management
- JBoss Enterprise Data Services Platform: Bridge the gap between diverse existing enterprise data sources and the new forms of data required by new projects, applications, and architectures

Enterprise Frameworks:

- JBoss Hibernate Framework: Industry-leading object/relational mapping and persistence
- JBoss Seam Framework: Powerful application framework for building next generation Web 2.0 applications
- JBoss Web Framework Kit: A combination of popular open source web frameworks for building light and rich Java applications
- JBoss jBPM Framework: Business process automation and workflow engine

Management:

- JBoss Operations Network (JON): An advanced management platform for inventorying, administering, monitoring, and updating JBoss Enterprise Platform deployments

4.5.1 JBoss Enterprise Application Platform (EAP)

JBoss Enterprise Application Platform is the market leading platform for innovative and scalable Java applications. Integrated, simplified, and delivered by the leader in enterprise open source software, it includes leading open source technologies for building, deploying, and hosting enterprise Java applications and services.

JBoss Enterprise Application Platform balances innovation with enterprise class stability by integrating the most popular clustered Java EE application server with next generation



application frameworks. Built on open standards, JBoss Enterprise Application Platform integrates JBoss Application Server, with JBoss Hibernate, JBoss Seam, and other leading open source Java technologies from JBoss.org into a complete, simple enterprise solution for Java applications.

Features and Benefits:

- Complete Eclipse-based Integrated Development Environment (JBoss Developer Studio)
- Built for Standards and Interoperability: JBoss EAP supports a wide range of Java EE and Web Services standards
- Enterprise Java Beans and Java Persistence
- JBoss EAP bundles and integrates Hibernate, the de facto leader in Object/Relational mapping and persistence
- Built-in Java naming and directory interface (JNDI) support
- Built-in JTA for two-phase commit transaction support
- JBoss Seam Framework and Web Application Services
- Caching, Clustering, and High Availability
- Security Services
- Web Services and Interoperability
- Integration and Messaging Services
- Embeddable, Service-Oriented Architecture microkernel
- Consistent Manageability

4.5.2 JBoss Operations Network (JON)

JON is an integrated management platform that simplifies the development, testing, deployment and monitoring of JBoss Enterprise Middleware. From the JON console one can:

- Inventory resources from the operating system to applications
- Control and audit application configurations to standardize deployments. manage, monitor and tune applications for improved visibility, performance and availability

One central console provides an integrated view and control of JBoss middleware infrastructure.

The JON management platform (server-agent) delivers centralized systems management for the JBoss middleware product suite. With it one can coordinate the many stages of application life cycle and expose a cohesive view of middleware components through complex environments, improve operational efficiency and reliability through thorough visibility into production availability and performance, and effectively manage configuration and rollout of new applications across complex environments with a single, integrated tool.

- Auto-discover application resources: operating systems, applications and services
- From one console, store, edit and set application configurations
- Start, stop, or schedule an action on an application resource
- Remotely deploy applications
- Monitor and collect metric data for a particular platform, server or service
- Alert support personnel based upon application alert conditions

- Assign roles for users to enable fine-grained access control to JON services

4.5.3 Red Hat Enterprise MRG Grid

MRG^{vi} Grid provides high throughput and high performance computing. Additionally, it enables enterprises to move to a utility model of computing to help them achieve both higher peak computing capacity and higher IT usage by leveraging their existing infrastructure to build high performance grids.

Based on the Condor project, MRG Grid provides the most advanced and scalable platform for high throughput and high performance computing with capabilities such as:

- Scalability to run the largest grids in the world
- Advanced features for handling priorities, workflows, concurrency limits, usage, low latency scheduling, and more
- Support for a wide variety of tasks, ranging from sub-second calculations to long running, highly parallel (MPI) jobs
- The ability to schedule to all available computing resources, including local grids, remote grids, virtual machines, idle desktop workstations, and dynamically provisioned cloud infrastructure

MRG Grid also enables enterprises to move to a utility model of computing, where they can:

- Schedule a variety of applications across a heterogeneous pool of available resources
- Automatically handle seasonal workloads with high efficiency, usage, and flexibility
- Dynamically allocate, provision, or acquire additional computing resources for additional applications and loads
- Execute across a diverse set of environments, ranging from virtual machines to bare metal hardware to cloud-based infrastructure



5 RHCF Proof of Concept Configuration

Customers often have different requirements and standards when it comes to what can be deployed. This proof of concept provides one combination of the hardware and software versions that were tested in the Red Hat Reference Architecture labs. This section provides an overview of the software and hardware required to build a Red Hat Cloud Foundations solution.

As shown in **Figure 5.1: Infrastructure Overview**, there are two clusters configured. The first cluster is a cloud services infrastructure management cluster set up with Red Hat Cluster High Availability running on Red Hat Enterprise Linux version 6.0 hosted on two IBM BladeCenter^{vii} servers. Attached is an HP MSA fibre channel storage array that presents a single 500GB LUN to both servers. This shared storage enables the virtual machines to be stored on the SAN and failed over to the other node in the event of a host failure or scheduled maintenance. This cluster hosts the virtual machines that provide management and infrastructure services like Satellite server, DNS, DHCP, PXE, JBoss Operations Network and MRG Manager.

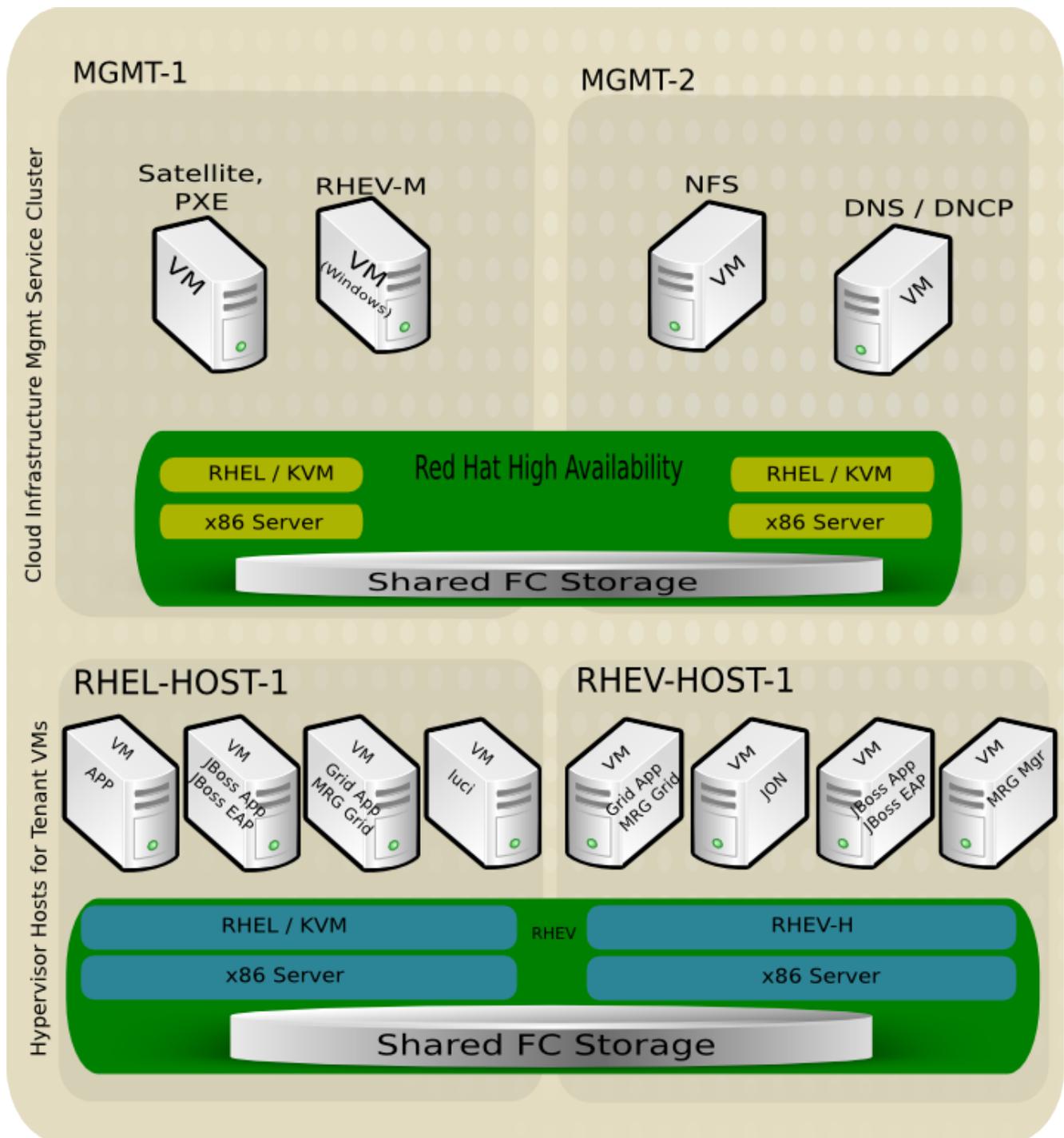


Figure 5.1: Infrastructure Overview

The second cluster is running Red Hat Enterprise Virtualization with one host using a Red Hat Enterprise Virtualization Hypervisor and the other host running Red Hat Enterprise Linux with KVM – both are on IBM BladeCenter servers. The HP MSA fibre channel storage array is also presenting a single 500GB LUN to these two hosts which enables fail-over of the virtual machines. This cluster is responsible for hosting the virtual machines that run the applications which scale out. These applications are a Seam based hotel reservation system running on JBoss Enterprise Application Platform, a custom Java application called javaApp and finally a perfect number application running on MRG grid execution nodes.



By utilizing industry standard hardware for storage, servers and networking along with open source software that is fully supported and tested for compatibility one can deploy a Red Hat Cloud Foundations solution with confidence. The remainder of this chapter explores the details of how an environment needs to be prepared to deploy a Red Hat Cloud Foundations solution.

5.1 Operating Systems

The operating systems that drive the Red Hat Cloud Foundations stack are detailed in **Table 1: Operating Systems**. The software stack provides a reliable and stable environment for customers to shift into the cloud.

| Server | Operating System | Physical or Virtual |
|---------------|------------------------------|---------------------|
| MGMT1 | Red Hat Enterprise Linux 6 | Physical |
| MGMT2 | Red Hat Enterprise Linux 6 | Physical |
| RHEV-H | Red Hat Enterprise Linux 5.6 | Physical |
| RHEL-H | Red Hat Enterprise Linux 5.6 | Physical |
| SAT-VM | Red Hat Enterprise Linux 5.6 | Virtual |
| LUCI-VM | Red Hat Enterprise Linux 6 | Virtual |
| RHEVM-VM | Microsoft Windows 2008 R2 | Virtual |
| NFS-VM | Red Hat Enterprise Linux 6 | Virtual |
| DNSDHCP-VM | Red Hat Enterprise Linux 6 | Virtual |
| JON-SERVER | Red Hat Enterprise Linux 6 | Virtual |
| MRG-EXEC-NODE | Red Hat Enterprise Linux 5.6 | Virtual |
| JAVA-APP | Red Hat Enterprise Linux 6 | Virtual |
| MRG-MANAGER | Red Hat Enterprise Linux 5.6 | Virtual |
| JBOSS-EAP | Red Hat Enterprise Linux 5.6 | Virtual |
| WINDOWS-VM | Microsoft Windows 2008 R2 | Virtual |

Table 1: Operating Systems

5.1.1 Applications and Tools

The applications and tools that drive the Red Hat Cloud Foundations stack are shown in **Table 2: Applications**. The host that the software is running on is shown in the “Host” column.

| Software | Version | Host |
|----------------------------------------------|---------------|------------|
| jon-plugin-pack-eap | 2.2.3-45.el15 | JON-SERVER |
| jon-plugin-pack-ews | 3.5.4-68.el6 | JON-SERVER |
| jon-plugin-pack-soa | 2.4.1 | JON-SERVER |
| jon-server | 2.4.1 | JON-SERVER |
| jboss-seam-booking-ds.xml | 2.4.1 | JBOSS-EAP |
| jboss-seam-booking.ear | 2.4.1 | JBOSS-EAP |
| virtio-drivers | 1.0.0-45801 | WINDOWS-VM |
| virtio-win | 1.1.16-0 | WINDOWS-VM |
| RHEV-toolsSetup | 2.2_52832 | WINDOWS-VM |
| Satellite Server | 5.4 | SAT-VM |
| Red Hat Cluster High Availability - cman | 3.0.12-23 | MGMT{1,2} |
| Red Hat Cluster High Availability - luci | 0.22.2-14 | MGMT{1,2} |
| Red Hat Enterprise Virtualization Manager | 2.2.4.51796 | RHEV-M |
| RHEV Hypervisor VDSM | 2.2.62.30 | RHEV-H |
| RHEV Hypervisor VDSM | 2.2.63.21 | RHEL-H |
| RHEL Server Resilient Storage (LVM2-Cluster) | 2.02.72-8 | MGMT{1,2} |

Table 2: Applications



5.2 Hardware

This case study used IBM servers and HP Fibre channel attached storage. Please review the following tables for details.

5.2.1 Servers

The servers used in this case study are IBM BladeCenter servers with Quad Socket, Quad Core Intel Xeon processors with 64GB of RAM installed. See **Table 3: Hardware Stack** for more details.

| Hardware Systems | Specifications |
|----------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| MGMT{1,2} – RHCS Cluster Nodes [2 x IBM BladeCenter HS22] | Quad Socket, Quad Core (24 cores) Intel® Xeon® CPU X5680 @ 3.33GHz, 64GB RAM |
| | 2 x 146 GB SATA SSD internal disk drive (mirrored) |
| | 2 x QLogic ISP2532-based 8Gb FC HBA |
| | 2 x Emulex OneConnect 10Gb Ethernet Controller |
| RHEL-H, RHEV-H Cluster Nodes [2 x IBM BladeCenter HS22] | Quad Socket, Quad Core (24 cores) Intel® Xeon® CPU X5680 @ 3.33GHz, 64GB RAM |
| | 2 x 146 GB SATA SSD internal disk drive (mirrored) |
| | 2 x QLogic ISP2532-based 8Gb FC HBA |
| | 2 x Emulex OneConnect 10Gb Ethernet Controller |

Table 3: Hardware Stack

5.2.2 Storage

Table 4: Storage Hardware Displays the storage hardware used in this environment. Please refer to the storage vendors documentation for proper configuration guidelines.

| Hardware | Specifications |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| 1 x HP StorageWorks MSA2324fc Fibre Channel Storage Array + HP StorageWorks 70 Modular Smart Array with Dual Domain IO Module [24+25 x 146GB 10K RPM SAS disks] | Storage Controller: Code Version: M100R18 Loader Code Version: 19.006 |
| | Memory Controller: Code Version: F300R22 |
| | Management Controller Code Version: W440R20 Loader Code Version: 12.015 |
| | Expander Controller: Code Version: 1036 |
| | CPLD Code Version: 8 |
| | Hardware Version: 56 |
| 1 x HP StorageWorks 4/16 SAN Switch | Firmware: v5.3.0 |
| 1 x HP StorageWorks 8/40 SAN Switch | Firmware: v6.1.0a |

Table 4: Storage Hardware

Storage Volumes

Table 5: Storage LUNS shows the LUNs that are presented to the clusters. For one cluster, one LUN is presented and shared between both hosts so that fail-over of the virtual machines can be accomplished. The same goes for the second cluster. The RHHA Volume will utilize the Clustered Logical Volume software from the Red Hat Enterprise Linux Resilient Storage channel.

| Volume | Size | Presentation | Purpose |
|------------|--------|--------------|----------------------------------------|
| RHHA VOL 1 | 500 GB | MGMT{1,2} | Storage used for host Virtual Machines |
| RHEV VOL 1 | 500 GB | RHEVH{1,2} | RHEV-M Storage Pool |

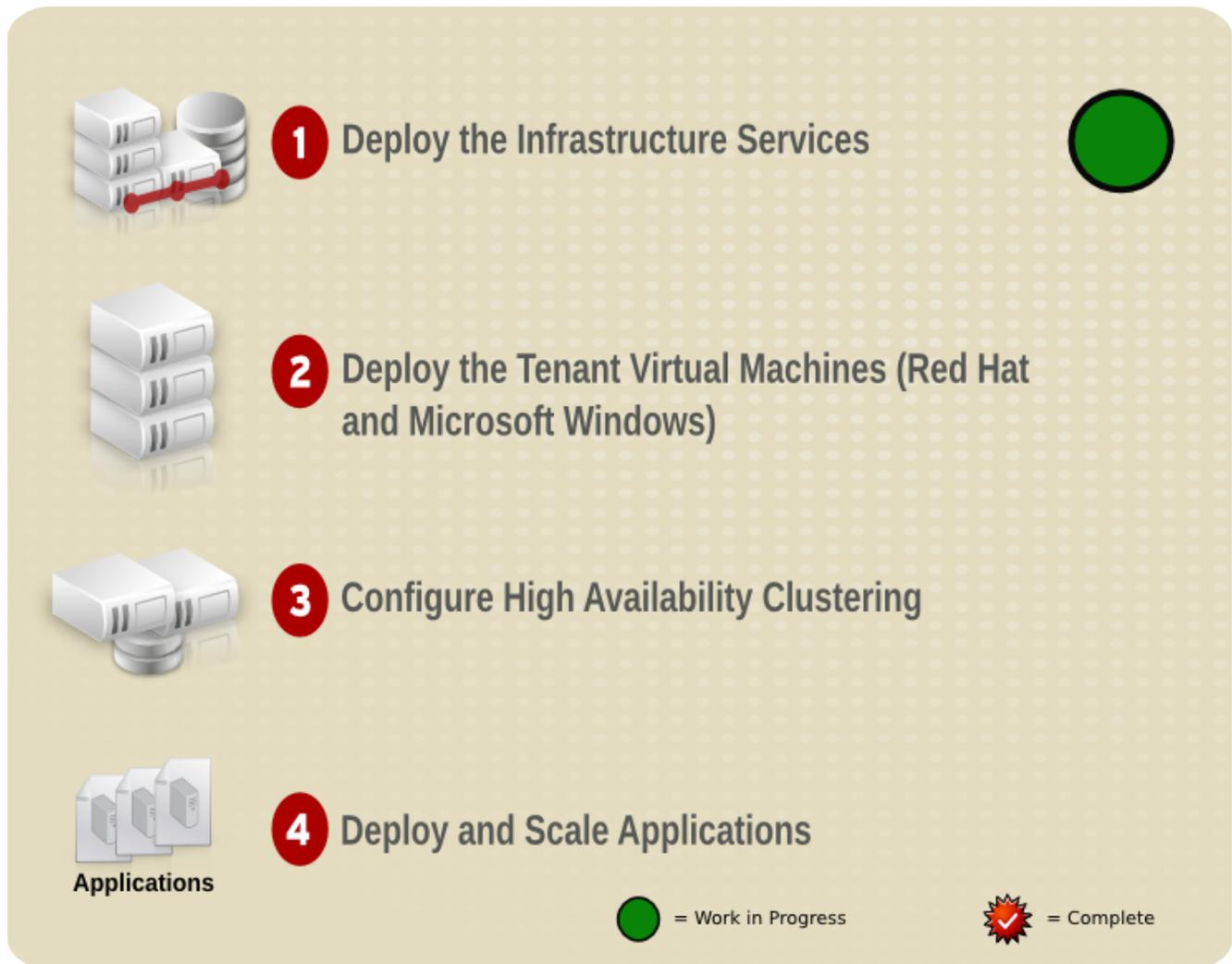
Table 5: Storage LUNS



6 Deploying Red Hat Cloud Foundations – Infrastructure Services

6.1 Overview

Figure 6.1: Overview shows the steps required to deploy the Red Hat Cloud Foundations infrastructure. The first step is to deploy the infrastructure indicated by the check box. The infrastructure consists of Red Hat High Availability cluster nodes, a Red Hat Network Satellite virtual machine, a DNS / DHCP server, an NFS server and finally the Red Hat Enterprise Virtualization hosts. The second step is to deploy Red Hat Enterprise Linux and Microsoft Windows tenant virtual machines. The reason for this is to explore deploying different types of virtual machines as well as to build out the foundation for the applications that are deployed later. Third, a high availability cluster is set up to provide the management services virtual machines. Finally, deploying and scaling applications demonstrates how Red Hat Cloud Foundations technologies can be utilized to create a dynamic infrastructure for the environment.



This section moves into the details of what needs to happen to deploy this infrastructure. At a high level, the steps that need to be accomplished.

1. Download the software
2. Deploy the first management server
3. Deploy the satellite virtual machine
4. Create the kickstart profiles and activation keys
5. Deploy the DNS / DHCP / Cobbler virtual machine
6. Deploy the RHEV-M virtual machine
7. Deploy the RHEV-H hypervisor
8. Deploy the RHEV-KVM hypervisor managed by RHEV
9. Configure RHEV: datacenter, cluster, storage, ISO domain

6.2 Download Software

Start by downloading the appropriate software from <http://rhn.redhat.com> to prepare the environment for a Red Hat Cloud Foundations build-out. The list of needed channels and software:

1. When logged into RHN, click on “Download Software”
2. In the “Red Hat Enterprise Virtualization” channel
 - a) Click on the “Red Hat Enterprise Virt Manager for Servers (v.2 x86)” channel
 - b) Download the following software
3. RHEV-M Microsoft Windows Installer
4. VirtIO Drivers VFD
5. Guest Tools ISO for 2.2
 - a) Click on the “Red Hat Enterprise Virtualization Hypervisor (v 5 x86-64)”
 - Click on the “Packages” tab in the upper navigation menu
 - Download the following software
6. `rhev-hypervisor-5.5-2.2.8.7.el5_5rhev2_2.noarch`
7. `rhev-hypervisor-pxe-5.5-2.2.8.7.el5_5rhev2_2.noarch`
 - Click on “Download Packages”
 - Click on “Download Selected Packages Now”
 - Save the files
8. In the “Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)” channel
 - a) Click on the “Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)” channel
 - b) Download RHEL5
9. In the “Red Hat Enterprise Linux (v. 6 for 64-bit x86_64)” channel
 - a) Click on the “Red Hat Enterprise Linux (v. 6 for 64-bit x86_64)” channel



- b) Download RHEL5
10. Log into <https://access.redhat.com> and download the JBoss Software
 - a) Click on the “Downloads” link in the upper navigation menu
 - b) Click on the “Download your software” under the “If you have active subscriptions” text
 - c) On the “Software Downloads” screen, select “JBoss ON for EAP”
 - Download the “EAP Lougin Pack for JBoss ON 2.4.1”
 - d) On the “Software Downloads” screen, select “JBoss ON for EWS”
 - Download the “EWS Plugin Pack for JBoss ON 2.4.1”
 - e) On the “Software Downloads” screen, select “JBoss Operations Network”
 - Download the “JBoss Operations Network 2.4.1 Base Distribution”
 - f) On the “Software Downloads” screen, select “JBoss ON for SOA-P”
 - Download the “SOA Plugin Pack for JBoss ON 2.4.1”

6.2.1 Download Scripts

Find a temporary place to put the software, once mgmt1 is installed along with the sat-vm as discussed in **Deploy the Satellite Virtual Machine**, the scripts and applications are placed there. Place the scripts and applications on the Red Hat Network Satellite server and place the installation ISOs on mgmt1 server. On the Red Hat Network Satellite server create two directories:

```
# mkdir -p /var/www/html/pub/{scripts,apps}
```

The following scripts are provided in **Appendix B Scripts** for reference. Place these scripts in `/var/www/html/pub/scripts`

- jboss-eap-install.sh
- jon-agent-install.sh
- rhq-agent-env.sh
- rhq-install.sh
- add-vms.ps1
- rhq-server.sh
- firewall-config.sh
- mrgMgr-config.sh
- mrgExec-config.sh
- condor_config.local.mrgexec
- condor_config.local.mrgManager

Place these applications in in `/var/www/html/pub/apps`

- jboss_plugins
- jboss-seam-booking-ds.xml
- jboss-seam-booking.ear
- jon-agent-2.4.1.GA.zip
- jon-license.xml
- jon-plugin-pack-dataservices-2.4.1.GA.zip
- jon-plugin-pack-eap-2.4.1.GA.zip
- jon-plugin-pack-ews-2.4.1.GA.zip
- jon-plugin-pack-soa-2.4.1.GA.zip
- jon-server-2.4.1.GA.zip
- org.jboss.on-jboss-on-parent-2.4.1.GA-sources.zip
- rhq-enterprise-agent-default.GA.jar
- perfect_number.sub
- perfect.tgz

Once the environment is set up, proceed with the deployment.

6.3 Deploy mgmt1 and Configure

The mgmt1 server is the catalyst for the rest of the environment. This system will host virtual machines and become one node of a two node cluster which is covered in **Configure High Availability Environment**. When installing this host, make sure to disconnect the SAN storage or use the `ignoredisk` as a kernel option in the Kickstart profile for RHEL 6 hosts or `nostorage` for RHEL5 so the operating system does not get installed on the external LUN.

Target System: mgmt1

1. Install Red Hat Enterprise Linux 6 (Reference a Red Hat Enterprise Linux 6 Admin Guide Here – Follow best practices for Security)
 - Include Clustering and Virtualization software groups when selecting software components
 - Set SELinux to permissive mode
 - Enable the firewall leaving ports open for ssh, http, and https
2. Register the system with RHN if that has not been completed
 1. Update the system via RHN

```
# yum update
```

3. Reboot into the new kernel if available
4. Install the bridge-utils package

```
# yum -y install bridge-utils
```

5. Make the following modifications to `ifcfg-eth0` file (Configuring the Bridge)
6. This will create a bridge to allow the virtual machines to communicate with each other directly.

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="none"
HWADDR="00:1A:64:76:00:00"
```



```
NM_CONTROLLED="yes"  
ONBOOT="yes"  
# IPADDR=10.16.139.20  
# NETMASK=255.255.248.0  
BRIDGE=rhcf
```

7. Copy ifcfg-eth0 to ifcfg-rhcf and make the following changes to the ifcfg-rhcf file

```
# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/network-  
scripts/ifcfg-rhcf  
# cat /etc/sysconfig/network-scripts/ifcfg-rhcf  
DEVICE="rhcf"  
BOOTPROTO="none"  
NM_CONTROLLED="yes"  
ONBOOT="yes"  
IPADDR=10.16.139.20  
NETMASK=255.255.248.0  
TYPE=Bridge
```

8. Disable netfilter on bridges by adding the following to /etc/sysctl.conf

```
# Disable netfilter on bridges.  
net.bridge.bridge-nf-call-ip6tables = 0  
net.bridge.bridge-nf-call-iptables = 0  
net.bridge.bridge-nf-call-arptables = 0
```

9. Make the netfilter changes active

```
# sysctl -p
```

10. Restart network services

```
# service network restart
```

11. Check to make sure the bridge interface is up

```
# brctl show
```

12. Set up multipath

- This ensures that there are multiple routes to the storage from either host depending on the hardware configuration. This is important for path failover. Please refer to the storage vendor documentation on how to properly configure the storage hardware.

13. Copy the example multipath.conf file to /etc/.

```
# cat /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf >  
/etc/multipath.conf
```

14. Set up the aliases

1. See **Appendix A** for example multipath.conf file
2. Change user friendly names to “no”
3. Pull the WWID and Set up the alias
4. Pull the WWID with multipath

```
# multipath -ll
35000cca00ac76d38 dm-1 IBM-ESXS,CBRCA146C3ETS0 N
size=137G features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  `- 0:0:0:0 sda 8:0 active ready running
rhcflun (3600c0ff000d84a99659e414d01000000) dm-2 HP,MSA2324fc
size=373G features='0' hwhandler='0' wp=rw
|+- policy='round-robin 0' prio=50 status=active
| |- 1:0:2:3 sdg 8:96 active ready running
| |- 2:0:2:3 sdh 8:112 active ready running
| |- 1:0:3:3 sdi 8:128 active ready running
| `-- 2:0:3:3 sdj 8:144 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 1:0:0:3 sdc 8:32 active ready running
  |- 2:0:0:3 sdd 8:48 active ready running
  |- 1:0:1:3 sde 8:64 active ready running
  `-- 2:0:1:3 sdf 8:80 active ready running
```

15. Restart multipathd

```
# service multipathd restart
# chkconfig multipathd on
```

16. Confirm the device is accessible via /dev/mapper or multipath -ll
17. User parted to apply a GPT label to the device

```
# parted /dev/mapper/rhcflun
(parted) mklabel gpt
(parted) quit
```

18. Install virtualization software

```
# yum install qemu-kvm libvirt virt-manager
```

19. Start libvirtd

```
# service libvirtd restart
```

20. Configure iptables to open ports for the cluster services. See **Table 6: Iptables for Cluster Services** for the ports that need to be opened. For this task, either create a new chain and place all custom rules in this chain or run the `firewall-config.sh` script that was downloaded earlier and is shown in **Appendix B Scripts**. The `firewall-config.sh` script takes port numbers and protocols as arguments. Please refer to the Red Hat Security Guide^{viii} for more information on using



iptables.

| IP Port Number | Protocol | Component |
|---------------------|-----------|---------------------------------------------------|
| 5404, 5405 | UDP | corosync / cman (Cluster Manager) |
| 11111 | TCP | ricci (part of Conga remote agent) |
| 21064 | TCP | dlm (distributed lock manager) |
| 50006, 50008, 50009 | TCP | ccsd (Cluster Configuration System Daemon) |
| 50007 | UDP | ccsd Cluster Configuration System Daemon) |
| 22 | TCP / UDP | Secure Shell Access |
| 80 | TCP / UDP | HyperText Transport Protocol |
| 443 | TCP / UDP | HyperText Transport Protocol over TLS / SSL |

Table 6: Iptables for Cluster Services

```
# ./firewall-config.sh
Please put the ports you would like to firewall here, separated by a space:
21064 22 443 80 50006 50008 50009 50007
Please put the protocols you would like to firewall here, separated by a
space:
tcp udp
iptables: Chain already exists.
iptables --append RHCf --protocol tcp --destination-port 21064 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 22 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 443 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 80 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 50006 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 50008 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 50009 --jump ACCEPT
iptables --append RHCf --protocol tcp --destination-port 50007 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 21064 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 22 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 443 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 80 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 50006 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 50008 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 50009 --jump ACCEPT
iptables --append RHCf --protocol udp --destination-port 50007 --jump ACCEPT
```

Remember to "service iptables save"

6.3.1 Set up the Logical Volumes for the Virtual Machines

For this section, a SAN LUN with at least 500GB of free space will need to be presented to mgmt1. 250GB will be more than enough space for this exercise. Using a SAN will allow for clustering this system later in the deployment.

Target System: mgmt1

1. Create the physical volume

```
# pvcreate /dev/mapper/rhcfdata
```

2. Create the volume group

```
# vgcreate vg_rhcf /dev/mapper/rhcfdata
```

3. Create the logical volumes

```
# lvcreate -L 200G -n lv_satellite vg_rhcf
# lvcreate -L 100G -n lv_rhevm vg_rhcf
# lvcreate -L 30G -n lv_nfs vg_rhcf
# lvcreate -L 20G -n lv_dnshcpcoobbler vg_rhcf
```

6.4 Deploy Satellite Virtual Machine and Install Satellite

6.4.1 Deploy the Satellite Virtual Machine^{ix}

Target System: mgmt1

1. Launch `virt-manager` and manually deploy a Red Hat Enterprise Linux 5.6 virtual machine and call it `rhcf-sat-vm`
 1. Launch `virt-manager`

```
# virt-manager
```

2. Click the "New" button on the top navigation pane.
 1. On the "Step 1 of 5 page"
 1. Provide the virtual machine name, e.g. `sat-vm`
 2. Choose "Local install media (ISO image or CDROM)"
 3. Click "Forward"
 2. On "Step 2 of 5"
 1. Browse to the ISO image
 2. Change "OS Type" to "Linux"
 3. Change "Version" to "Red hat Enterprise Linux 5.4 or Later"
 4. Click "Forward"



3. On “Step 3 of 5”
 1. Change “Memory (RAM)” to “8192” MB
 2. Change “CPUs” to “8”
 3. Click “Forward”
 4. On “Step 4 of 5”
 1. Select “Select managed or other existing storage” and browse out to the logical volume you created `lv_satellite`
 2. Click “Forward”
 5. On “Step 5 of 5”
 1. Click “Finish”
3. Click the “Open” button on the top navigation pane to open a console to the virtual machine and perform the install using the following guidelines:
- SELinux should be disabled

Deploy Satellite Software

Target System: `sat-vm`

1. After the virtual machine is installed, get the satellite server software from the software repository that was set up earlier

1. Update the system

```
# yum update
```

2. Install `satellite-server`^x

2. Synchronize the channels (This may take a while) See **Table 7: Red Hat Network Channels** for details on what each channel provides.

```
# satellite-sync \  
--channel rhel-x86_64-server-6 \  
--channel rhn-tools-rhel-x86_64-server-6 \  
--channel rhel-x86_64-server-ha-6 \  
--channel rhel-x86_64-server-5 \  
--channel rhn-tools-rhel-x86_64-server-5 \  
--channel rhel-x86_64-server-vt-5 \  
--channel rhel-x86_64-rhev-mgmt-agent-5 \  
--channel rhel-x86_64-server-rs-6 \  
--channel jbappplatform-5-x86_64-server-5-rpm \  
--channel rhel-x86_64-server-supplementary-5 \  
--channel rhel-x86_64-server-5-mrg-grid-1 \  
--channel rhel-x86_64-server-5-mrg-grid-execute-1 \  
--channel rhel-x86_64-server-5-mrg-management-1
```

| Channel | Name | Purpose |
|-----------------------------------------|-------------------------------------------------------------------|--------------------------------------------------------------|
| rhel-x86_64-server-6 | Red Hat Enterprise Linux (v. 6 for 64-bit AMD64 / Intel64 Server) | Operating System |
| rhn-tools-rhel-x86_64-server-6 | Red Hat Network Tools for Red Hat Enterprise Linux | Allows the systems to be managed by Satellite |
| rhel-x86_64-server-5 | Red Hat Enterprise Linux (v. 5 for 64-bit AMD64 / Intel64 Server) | Operating System |
| rhn-tools-rhel-x86_64-server-5 | Red Hat Network Tools for Red Hat Enterprise Linux | Allows the systems to be managed by Satellite |
| rhel-x86_64-server-vt-5 | Red Hat Enterprise Linux Virtualization Environment | |
| rhel-x86_64-rhev-mgmt-agent-5 | Red Hat Enterprise Virtualization Management Agent | |
| rhel-x86_64-server-rs-6 | Red Hat Enterprise Linux Server Resilient Storage | Provides required dependencies for Red Hat High Availability |
| jbappplatform-4-x86_64-server-5-rpm | JBoss Application Platform | |
| rhel-x86_64-server-supplementary-5 | Red Hat Enterprise Linux Supplementary Software | Provides access to openjdk |
| rhel-x86_64-server-5-mrg-grid-1 | Red Hat Enterprise MRG Grid | |
| rhel-x86_64-server-5-mrg-grid-execute-1 | Red Hat Enterprise MRG Grid Execute Node | |
| rhel-x86_64-server-5-mrg-management-1 | Red Hat Enterprise MRG Management | |

Table 7: Red Hat Network Channels

3. Visit the fully qualified domain name for the satellite server via a browser
 1. Create a satellite administrator

6.4.2 Configure Iptables

Target System: sat-vm

1. Configure iptables to open ports for the Red Hat Satellite. See **Table 8: Iptables for Satellite Server** for the ports that need to be opened. For this task, create a new chain



and place all custom rules in this chain. Please refer to the Red Hat Security Guide¹ for more information on using iptables

| IP Port Number | Protocol | Component |
|----------------|-----------|---------------------------------------------|
| 4545 | TCP / UDP | RHN Monitoring Funcionality |
| 5222 | TCP / UDP | RHN Push Funcionality |
| 22 | TCP / UDP | Secure Shell Access |
| 80 | TCP / UDP | HyperText Transport Protocol |
| 443 | TCP / UDP | HyperText Transport Protocol over TLS / SSL |

Table 8: Iptables for Satellite Server

```
# ./firewall-config.sh
Please put the ports you would like to firewall here, separated by a space:
80 443 50007 50006 50008 50009 21064 11111 5404 5405
Please put the protocols you would like to firewall here, separated by a
space:
tcp udp
iptables --append RHCF --protocol tcp --destination-port 80 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 443 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 50007 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 50006 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 50008 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 50009 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 21064 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 11111 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 5404 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 5405 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 80 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 443 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 50007 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 50006 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 50008 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 50009 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 21064 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 11111 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 5404 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 5405 --jump ACCEPT
```

Remember to "service iptables save"

¹ http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/96/html/Security_Guide/index.html

6.5 Create Kickstart Profiles and Activation Keys

6.5.1 Create Activation Keys

Target System: Browser with access to sat-vm

Create a Red Hat Enterprise Linux 5 KVM Hypervisor Activation Key

1. Click the “Systems” tab on the upper navigation menu
2. Click “Activation Keys” on left navigation menu
3. Click “Create New Key” on upper navigation menu
4. Provide the following information
 1. Description: “RHEV-H KVM Hypervisor”
 2. Key: “RHEVKVMHYPER”
 3. Add-On Entitlements: “Provisioning” and “Virtualization Platform”
 4. Click “Create Activation Key”
5. On the RHEV-H KVM Hypervisor Activation Key screen, click the “Child Channels” tab in the upper navigation menu
 1. Keep the defaults and select the following channels
 - “Red Hat Enterprise Virt Management Agent”
 - “RHEL Virtualization”
 - “RHN Tools”
 2. Click “Update Key”

Create a Red Hat Enterprise Linux 6 Cluster Activation Key

1. Click the “Systems” tab on the upper navigation menu
2. Click “Activation Keys” on left navigation menu
3. Click “Create New Key” on upper navigation menu
4. Provide the following information
 1. Description: “Red Hat Enterprise Linux Cluster”
 2. Key: “cluster”
 3. Add-On Entitlements: “Provisioning”
 4. Click “Create Activation Key”
5. On the Red Hat Enterprise Linux Cluster Activation Key screen, click the “Child Channels” tab in the upper navigation menu
 1. Keep the defaults and select the following channels
 - “RHEL Server High Availability (v. 6 for 64-bit x86_64)”



- “RHN Tools for RHEL (v. 6 for 64-bit x86_64)”
 - “RHEL Server Resilient Storage (v. 6 for 64-bit x86_64)”
2. Click “Update Key”

6.5.2 Create Kickstart Profiles

Target System: Browser with access to sat-vm

Create a Red Hat Enterprise Linux 5 Base Kickstart Profile

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
 1. Provide a “Label” for the kickstart “rhel5_base_bare_metal”
 2. On “Base Channel” select “Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)”
 3. On “Kickstartable Tree” select “ks-rhel-x86_64-server-5-u6”
 4. Click “Next”
5. On “Step 2”
 1. Click “Next”
6. On “Step 3”
 1. Provide a root password
7. Click “Finish”

Create a Red Hat Enterprise Linux 6 Base Kickstart Profile

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
 1. Provide a “Label” for the kickstart “rhel6_base_bare_metal”
 2. On “Base Channel” select “Red Hat Enterprise Linux (v. 6 for 64-bit x86_64)”
 3. On “Kickstartable Tree” select “ks-rhel-x86_64-server-6-6.0”
 4. Click “Next”
5. On “Step 2”
 1. Click “Next”
6. On “Step 3”
 1. Provide a root password

7. Click “Finish”

Create a Red Hat Enterprise Linux 5 Base Virtual Machine Kickstart Profile

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
 1. Provide a “Label” for the kickstart “rhel5_base_virtual_machine”
 2. On “Base Channel” select “Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)”
 3. On “Kickstartable Tree” select “ks-rhel-x86_64-server-5-u6”
 4. On “Virtualization Type” select “KVM Virtualized Guest”
 5. Click “Next”
5. On “Step 2”
 1. Click “Next”
6. On “Step 3”
 1. Provide a root password
7. Click “Finish”

Create a Red Hat Enterprise Linux 6 Base Virtual Machine Kickstart Profile

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
 1. Provide a “Label” for the kickstart “rhel6_base_virtual_machine”
 2. On “Base Channel” select “Red Hat Enterprise Linux (v. 6 for 64-bit x86_64)”
 3. On “Kickstartable Tree” select “ks-rhel-x86_64-server-6-6.0”
 4. On “Virtualization Type” select “KVM Virtualized Guest”
 5. Click “Next”
5. On “Step 2”
 1. Click “Next”
6. On “Step 3”
 1. Provide a root password
7. Click “Finish”



Create a Red Hat Enterprise Linux 6 Clustering Kickstart Profile and Associate Activation Key

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
 1. Provide a “Label” for the kickstart “rhel6_clustering”
 2. On “Base Channel” select “Red Hat Enterprise Linux (v. 6 for 64-bit x86_64)”
 3. On “Kickstartable Tree” select “ks-rhel-x86_64-server-6-6.0”
 4. Click “Next”
5. On “Step 2”
 1. Click “Next”
6. On “Step 3”
 1. Provide a root password
7. Click “Finish”
8. Click “Activation Keys” in the upper navigation menu
 1. In the “Kickstart Details” screen, select the “RHELCluster” key
 2. Click “Update Activation Keys”

Create a Red Hat Enterprise Linux 5 KVM Hypervisor Kickstart Profile and Associate Activation Key

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
 1. Provide a “Label” for the kickstart “rhel5_kvm_bare_metal”
 2. On “Base Channel” select “Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)”
 3. On “Kickstartable Tree” select “ks-rhel-x86_64-server-5-u6”
 4. Click “Next”
5. On “Step 2”
 1. Click “Next”
6. On “Step 3”
 1. Provide a root password
7. Click “Finish”

8. Click “Activation Keys” in the upper navigation menu
 1. In the “Kickstart Details” screen, select the “RHEVKVMHYPER” key
 2. Click “Update Activation Keys”

6.6 Deploy DHCP / DNS Virtual Machine

Target System: mgmt1

1. Launch `virt-manager` and manually deploy a Red Hat Enterprise Linux 6.0 virtual machine and call it `dnsdhcp-vm`
 1. Run `virt-manager` with the following command

```
# virt-manager
```

2. Click the “New” button on the top navigation pane.
 1. On the “Step 1 of 5 page”
 1. Provide the virtual machine name, e.g. `dnsdhcp-vm`
 2. Choose “Local install media (ISO image or CDROM)”
 3. Click “Forward”
 2. On “Step 2 of 5”
 1. Browse to the ISO image
 2. Change “OS Type” to “Linux”
 3. Change “Version” to “Red hat Enterprise Linux 6.0 or Later”
 4. Click “Forward”
 3. On “Step 3 of 5”
 1. Change “Memory (RAM)” to “8192” MB
 2. Change “CPUs” to “8”
 3. Click “Forward”
 4. On “Step 4 of 5”
 1. Select “Select managed or other existing storage” and browse out to the logical volume you created `lv_dnsdhcp`
 2. Click “Forward”
 5. On “Step 5 of 5”
 1. Click “Finish”
3. Click the “Open” button on the top navigation pane to open a console to the virtual machine and perform the install using the kickstart profile

6.6.1 Configure DHCP

Target System: `dhsdhcp-vm`



1. Use the example dhcp.conf file to get started with a template.

```
# cat /usr/share/doc/dhcp-4.1.1/dhcpd.conf.sample > /etc/dhcp/dhcpd.conf
```

See **Appendix A** for example dhcpd.conf

2. Make sure to backup any configuration files before making any changes.

6.6.2 Configure DNS

Target System: dnshcp-vm

1. Configure DNS, see **Appendix A** for example zone files, place in correct directory

```
# service named configtest
```

1. Test name resolution

```
# host sat-vm  
sat-vm.rhcf.lab has address x.x.x.x
```

2. Start DNS

```
# service named restart
```

3. Add the name server to /etc/resolv.conf
4. Make DNS a persistent service

```
# chkconfig named on
```

5. See **Appendix A** for example zone file

6.7 Deploy Red Hat Enterprise Virtualization Platform

In this section, one Red Hat Enterprise Virtualization Manager system and one RHEV system will be deployed, also, one RHEL system with KVM is going to be deployed and managed by RHEV. The cloud virtual machines are going to be deployed onto this infrastructure. This Red Hat Cloud Foundations Architecture is only using 2 hypervisors, but RHEV can support many more.

6.7.1 Deploy RHEV-M Virtual Machine

The Red Hat Enterprise Virtualization Manager virtual machine hosts the rhev-m software which controls the virtualization environment. This system runs Microsoft Windows Server 2008 Release 2.

Target System: mgmt1

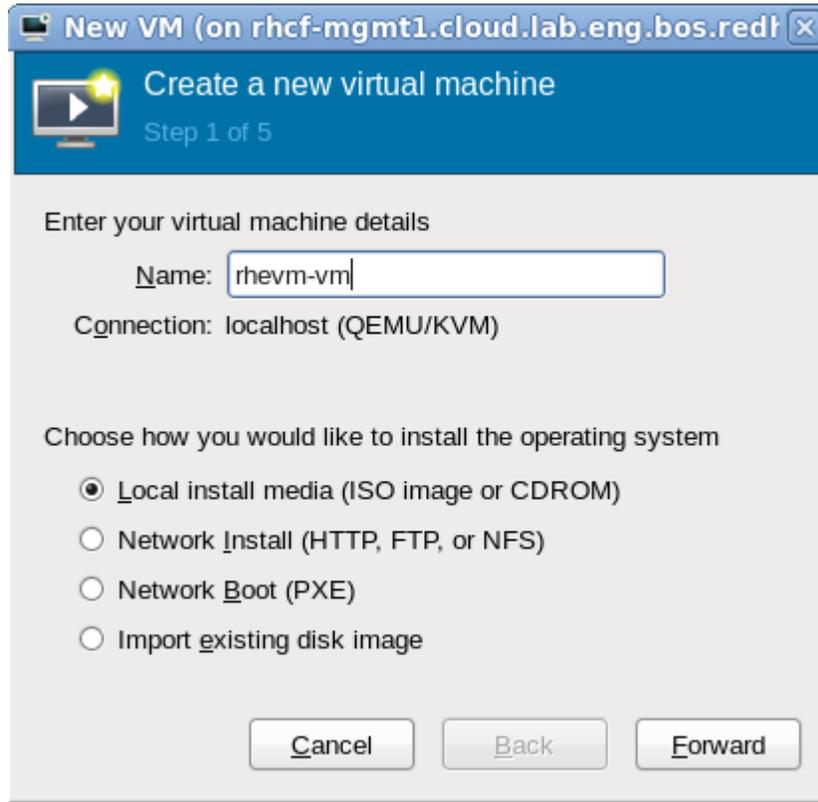
Launch `virt-manager` and manually deploy a Microsoft Windows 2008 R2 virtual machine and call it `rhev-m-vm`.

2. Install `virtio-win` drivers and Launch `virt-manager`

```
# yum install virtio-win  
# virt-manager
```



3. Click the “New” button on the top navigation pane.
 1. On “Step 1”
 1. Provide a virtual machine name
 2. Choose “Local install media (ISO image or CDROM)”
 3. Click “Forward” as shown in **Figure 6.2:virt-manager for RHEVM-M**



2. On “Step 2”
 1. Browse to the ISO image
 2. Change “OS Type” to “Windows”
 3. Change “Version” to “Microsoft Windows 2008”
 4. Click “Forward” as shown in **Figure 6.3: virt-manager for RHEVM-M**

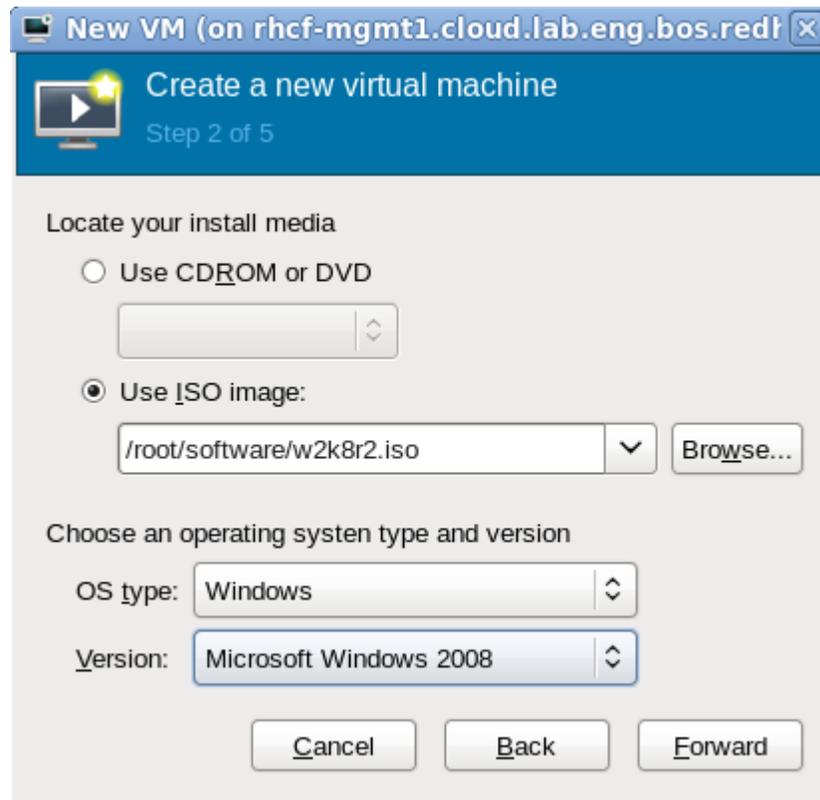


Figure 6.3: virt-manager for RHEVM-M

3. On “Step 3”



1. Change “Memory (RAM)” to “8092” MB
2. Change “CPUs” to “2”
3. Click “Forward” as shown in **Figure 6.4: virt-manager Memory for RHEVM**

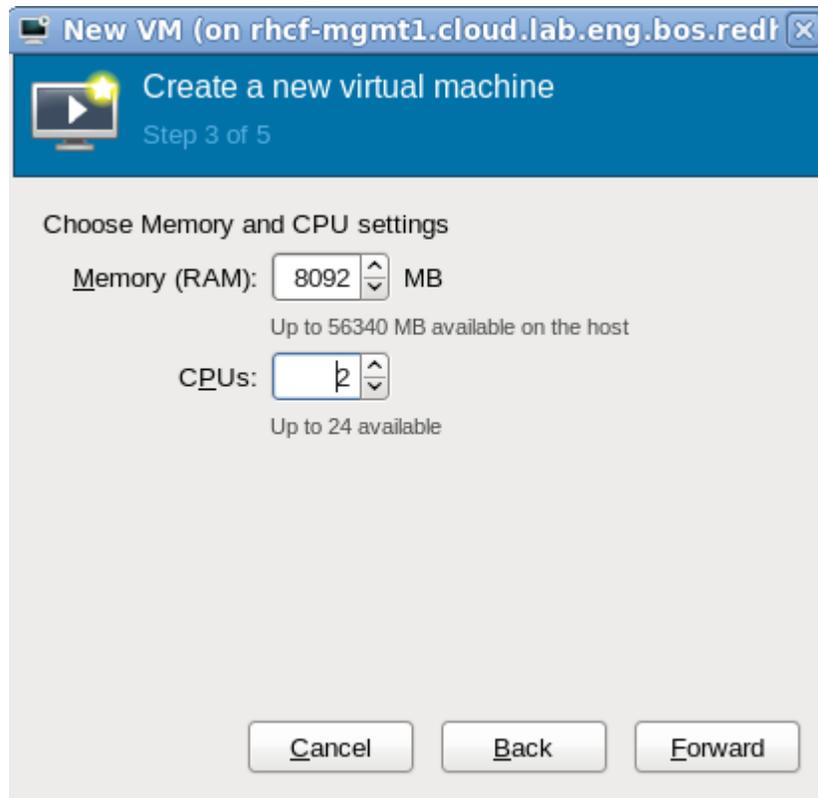


Figure 6.4: virt-manager Memory for RHEVM

4. On “Step 4”

1. Select “Select managed or other existing storage” and browse out to the logical volume that was created earlier: lv_rhev
2. Click “Forward” as shown in **Figure 6.5: RHEVM-M Volume**

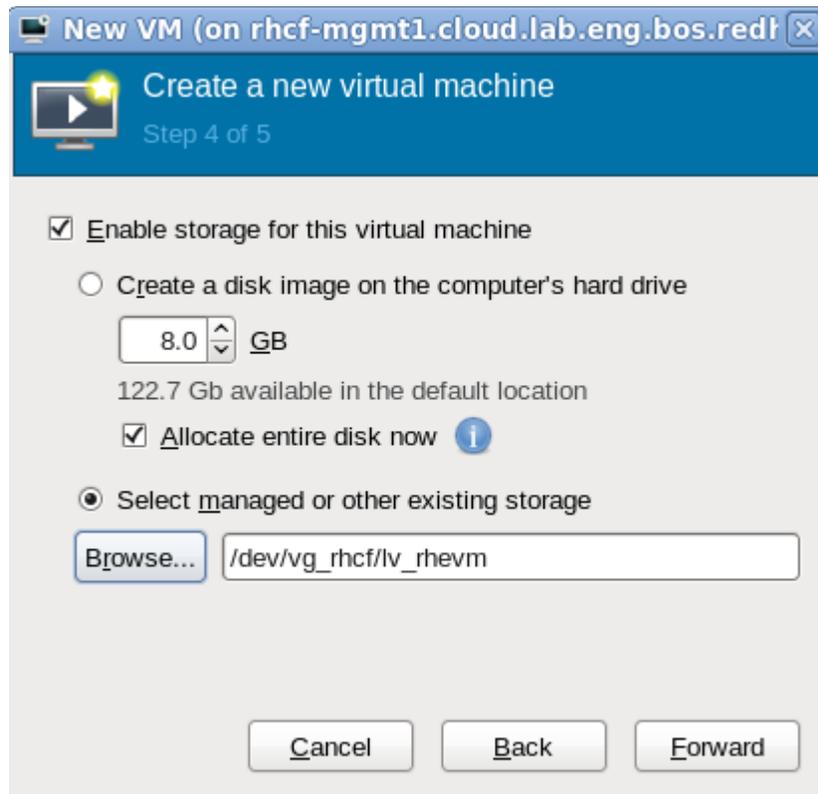


Figure 6.5: RHEVM-M Volume

5. On “Step 5”



1. Check the “Customize Configuration before install” check-box as shown in **Figure 6.6: RHEVM-M Customize**

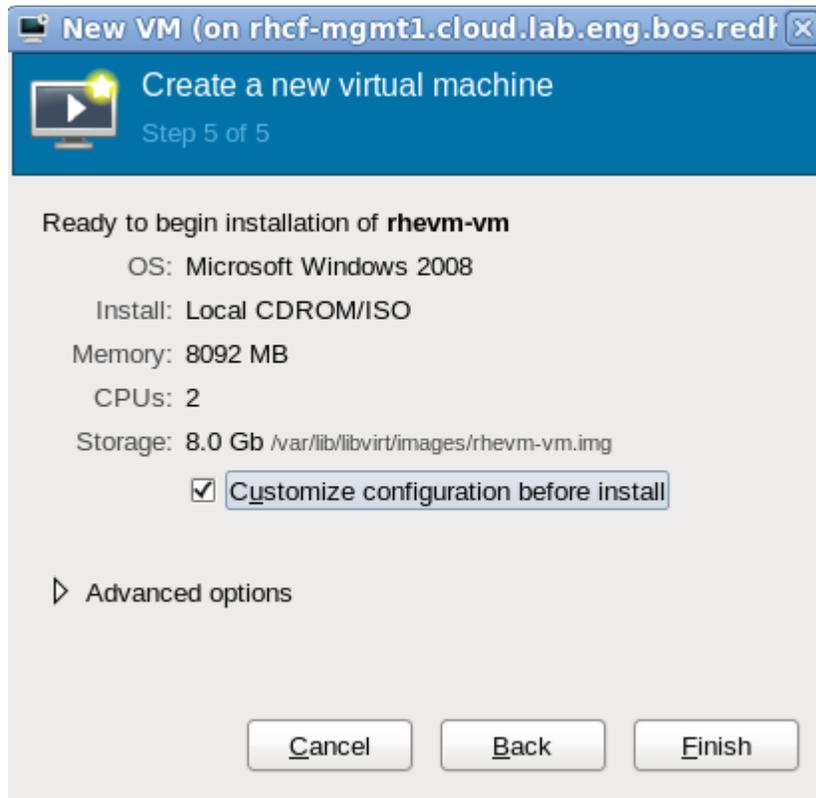


Figure 6.6: RHEVM-M Customize

2. Click “Finish”

1. On the “Basic Details” screen, click “Add Hardware” on the left navigation panel

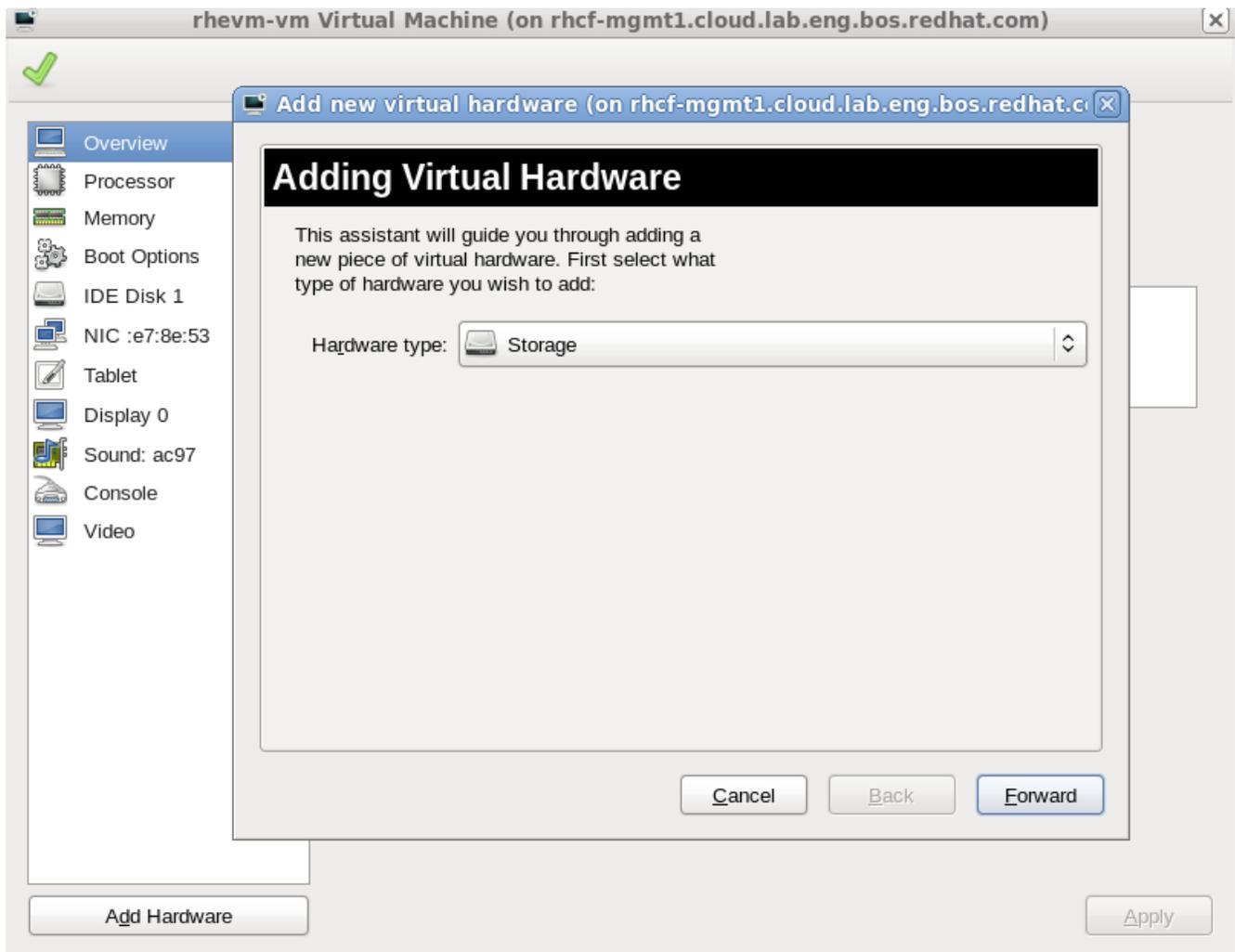


Figure 6.7: RHEVM Add Hardware

2. Click “Forward” on Adding Virtual Hardware screen as shown in **Figure**



1. Browse to the virtio-win.vfd drivers
 1. Click “Forward”
 2. Click “Finish” as shown in **Figure 6.8: RHEVM-M Add VirtIO Drivers**

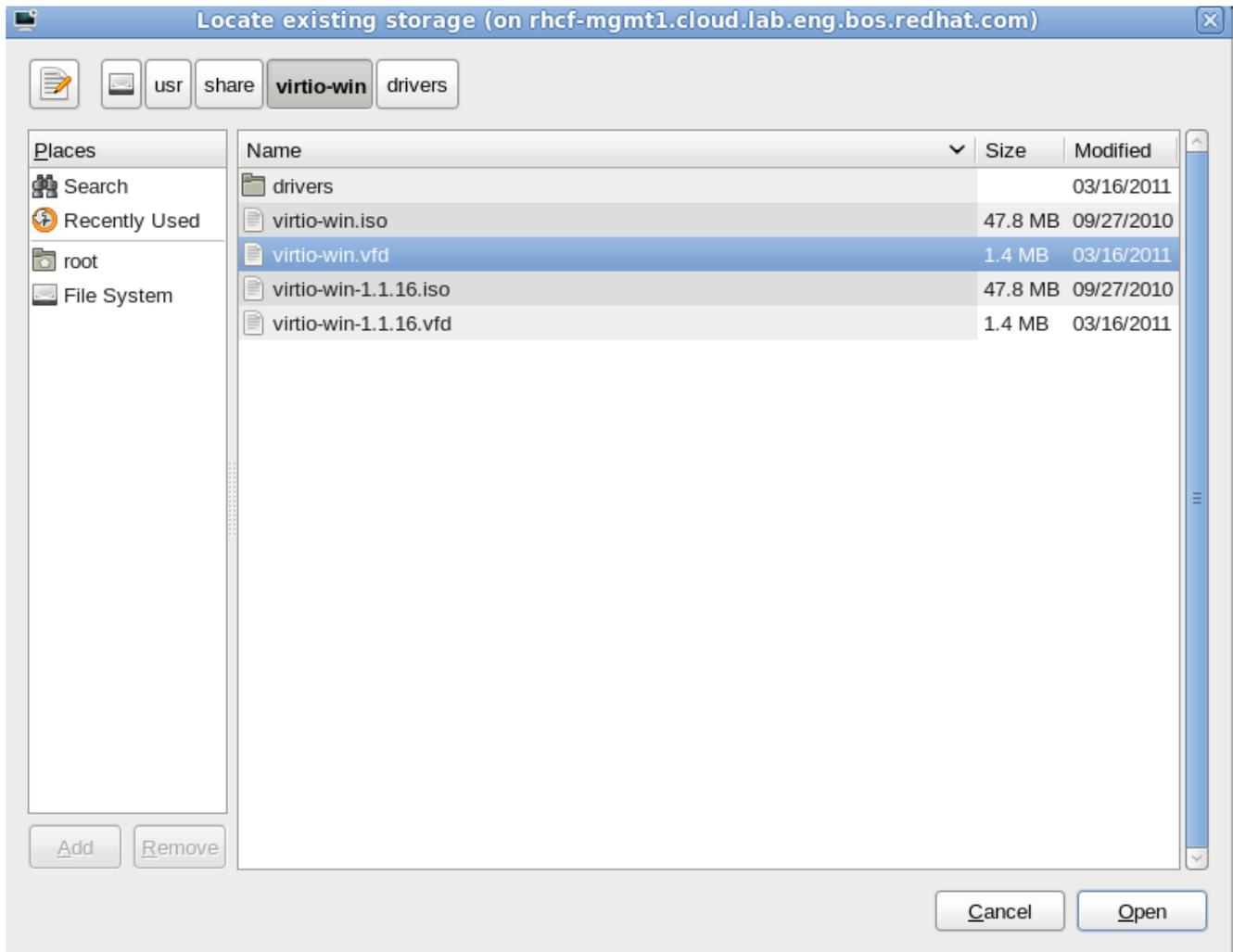


Figure 6.8: RHEVM-M Add VirtIO Drivers

3. Close the “Add Hardware” window
6. Click the “Open” button on the top navigation pane to open a console to the virtual machine and perform the install.
4. Install Microsoft Windows 2008 R2
 1. When the virtual machine is posting, hit “F12” and choose “CDROM”
 2. In the console of virt-manager for the VM
 1. Click “Next” on the opening screen of the install
 3. Click “Install Now”

4. On the “Select the operating system you want to install” screen, click “Next”
5. On the “Please read the license terms” screen, accept and click “Next”
6. On the “Which type of installation do you want” screen
 1. Click “Custom (advanced)”
7. On the “Where do you want to install Windows” screen
 1. Click “Load Driver”
8. On the “Load Driver” screen
 1. Click “Browse”
9. On the “Browse for Folder” screen, select “Floppy Disk Drive (A:)”
 1. Browse out to the drivers
 1. amd64 | Win2008
 2. Click “Ok”
10. On the “Select the driver to be installed” screen
 1. Click “Next”
11. On the “Where do you want to install Windows” screen, click “Next”
12. The system will install and reboot and complete the installation
5. When the system reboots, set the administrator password
6. Install the VirtIO network drivers
 1. On the `virt-manager` interface click the “View” menu item, then “Details”
 1. On the left navigation pane select the “NIC: ” and click “Remove”
 2. Click “Add Hardware”
 3. Change “Hardware Type” to “Network”
 4. Click “Forward”
 2. On the “Network” screen
 1. Change the “Device Model” to “VirtIO”
 2. Click “Forward”
 3. Click “Finish”
 4. Attach the VirtIO network ISO and add drivers
 1. On the `virt-manager` interface click the “View” menu item, then “Details”
 1. On the left navigation panel
 1. Select “IDE CDROM 1”
 2. Click “Connect”
 2. Select “ISO Image Location” and browse to the VirtIO network driver ISO file `/usr/share/virtio-win/virtio-win.iso`



5. Boot the virtual machine and install the drivers
 1. Open Microsoft Windows explorer and browse to the CDROM and install the drivers.
 1. The executable will be called “RHEV-ToolsSetup_52832”
 7. Enable Remote Desktop on the Microsoft Windows virtual machine
 8. Enable the appropriate ports via the Microsoft Windows Firewall configuration utility
 9. Add rhevm-vm entries to the DNS / DHCP server
 10. Connect to the rhevm-vm via rdesktop from a Linux host
- ```
$ rdesktop -0 -u Administrator -p $password rhevm-vm -f &
```
11. Update the system and reboot

## 6.7.2 Install Red Hat Enterprise Virtualization Manager Software

### 6.7.2.1 Prepare the Microsoft Windows 2008 R2 server for RHEV

Target System: rhevm-vm

1. Open the “Add Roles” wizard on the Microsoft Windows 2008 R2 virtual machine
2. On the “Before You Begin” screen,
  1. Click “Next”
3. On the “Select Server Roles” screen
  1. Select the appropriate roles
    - Application Server
    - Web Server
  2. Click “Next”
4. On the “Introduction to Application Server” screen
  1. Click “Next” and take the defaults the rest of the way
5. On the “To install the following roles, role services, or features” screen
  1. Click “Install”
  2. Click “Close” on the final screen

### 6.7.2.2 Disable “Internet Explorer Enhanced Security Configuration”

Target System: rhevm-vm

1. Open the “Server Manager” wizard

1. On the “Security Information” pane
  1. Click “Configure IE ESC”
    1. Disable IE ESC for administrators
    2. Click “OK”

### 6.7.2.3 Install Red Hat Enterprise Virtualization Manager

Target System: rhevm-vm

1. Copy the RHEV Manager software to the rhevm-vm server
  1. winscp<sup>xi</sup> will work for this or a direct download from RHN
2. Browse to where the RHEV-M software was downloaded and launch the installer
3. Take the defaults until the “Select RHEV Manager domain and User” screen
  1. Change “Select Local or Domain” to “Local”
  2. Provide the username and password, click “Next”
4. Take the defaults for the remainder of the install unless the environment requires customizations. Please reference the RHEV install guide for more information.

## 6.8 Deploy the Red Hat Enterprise Virtualization Hypervisor

### 6.8.1 Prepare the Environment to Deploy a RHEV Hypervisor

Target System: mgmt1

1. Copy it over to the mgmt1 server and install
  - rhev-hypervisor-pxe
  - rhev-hypervisor
2. Install the packages on the mgmt1 server, change into the working directory with the .tar file
  1. Expand the tar file

```
tar xvf rhn-packages.tar
```

2. Install the packages

```
yum localinstall rhn-packages/rhev-hypervisor-*
```

3. Using cobbler, set up a distribution RHEV-H server

```
cobbler distro add \
--name="rhev" \
--kernel=/usr/share/rhev-hypervisor/tftpboot/vmlinuz0 -
initrd=/usr/share/rhev-hypervisor/tftpboot/initrd0.img \
--kopts="rootflags=loop root=/rhev-hypervisor.iso rootfstype=auto
```



```
liveimg_rhn_url=https://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com
rhn_activationkey=1-RHEVKVMHYPER"
```

- Using cobbler, set up the profile for the RHEV-H server

```
cobbler profile add \
--name="rhev-profile" \
--distro="rhev" \
--kopts="storage_init=/dev/sda storage_vol=::::: management_server=rhcf-
rhev-vm.cloud.lab.eng.bos.redhat.com netconsole=rhcf-rhev-
vm.cloud.lab.eng.bos.redhat.com rootpw=<password> ssh_pwauth=1 firstboot
local_boot"
```

- Synchronize the new configuration

```
cobbler sync
```

- Open the tftp ports in iptables as shown in **Table 9: DNS / DHCP iptables**

| IP Port Number | Protocol  | Component |
|----------------|-----------|-----------|
| 69             | TCP / UDP | TFTP      |
| 53             | TCP / UDP | DNS       |
| 123            | TCP / UDP | NTP       |

**Table 9: DNS / DHCP iptables**

```
./firewall-config.sh
Please put the ports you would like to firewall here, separated by a
space:
53 123 69
Please put the protocols you would like to firewall here, separated by a
space:
tcp udp
iptables: Chain already exists.
iptables --append RHCF --protocol tcp --destination-port 53 --jump
ACCEPT
iptables --append RHCF --protocol tcp --destination-port 123 --jump
ACCEPT
iptables --append RHCF --protocol tcp --destination-port 69 --jump
ACCEPT
iptables --append RHCF --protocol udp --destination-port 53 --jump
ACCEPT
iptables --append RHCF --protocol udp --destination-port 123 --jump
ACCEPT
iptables --append RHCF --protocol udp --destination-port 69 --jump
ACCEPT
```

Remember to "service iptables save"

## 6.8.2 Deploy the RHEV Hypervisor

Target System: Bare Metal RHEV-H Server - rhev1

1. PXE boot the RHEV-H server and install.
2. Add to DNS / DHCP

## 6.8.3 Approve RHEV Hypervisor

Target System: RHEV-M Virtual Machine

1. On the "Hosts" tab
  1. Right click on the host that was just PXE booted and click "Approve"
  2. On the "Edit and Approve Host" screen
  3. Click "Enable Power Management"
  4. Provide the Appropriate information for the system

## 6.9 Deploy the RHEL KVM Hypervisor

### 6.9.1 Deploy Red Hat Enterprise Linux 5.6 Server with KVM

The rhev2 server will serve as the second RHEL Hypervisor. This server runs Red Hat Enterprise Linux 5.6 with KVM.

Target System: Bare Metal RHEV Server – rhev2

1. PXE install rhev2
2. Use the kickstart profile and activation key that was created earlier called rhl5\_base\_bare\_metal

## 6.10 Add the RHEV KVM Hypervisor in RHEV-M

Target System: rhevm-vm

1. Click on the "Hosts" tab in the upper navigation menu
2. Click the "New" button on the upper navigation menu
3. On the "New Host" screen provide the following information
  1. Name: rhev2
  2. Address: IP Address
  3. Host Cluster: Cluster Name
  4. Root Password: Password
  5. Enable Power Management and provide appropriate entries



- Power management tests will only work when there are at least two nodes in the cluster. The reason for this is because the test is based on fencing. There has to be at least two nodes in order for that to work.
6. Click “OK” as shown in **Figure 6.9: RHEV New Host**

**New Host**

Name:

Address:

Port:

Host Cluster:

Root Password:

Enable Power Management

Address:

User Name:

Password:

Type:

Port:

Slot:

Options:

Secure:

**Figure 6.9: RHEV New Host**

The software will be installed and the system will be rebooted.

## 6.11 Configure RHEV Datacenter, Cluster, and Storage Domain

Target System: rhevm-vm

1. Click on the “Data Center” tab on the upper navigation menu
2. Click on the “New” button on the upper navigation menu

1. Provide a “Name”
  2. Provide a “Description”
  3. Change the type to “FCP”
  4. Click “OK”
3. On the “New Data Center – Guide Me” Screen
1. Click “Configure Clusters”
  2. Provide a “Name”
  3. Provide a “Description”
  4. Pick the correct “CPU Name”
  5. Click “OK” as shown in **Figure 6.10: New Cluster Configuration**



**Figure 6.10: New Cluster Configuration**

4. On the “New Data Center – Guide Me” Screen
1. Click “Configure Storage”
  2. Provide “Name”
  3. Select the “Storage Type” of “FCP”
  4. In the “Discovered LUNS” panel, select the shared LUN
  5. Click “Add”



6. Click “OK” as shown in **Figure 6.11: RHEV Storage**

**New Domain**

Name:

Domain function:  Data  ISO  Export

Storage type:  NFS  iSCSI  FCP

Build New Domain  Use Preconfigured Volume Group

Use host:

**Discovered LUNs**

| LUN ID | UUID | Multipathing | Dev. Size |
|--------|------|--------------|-----------|
|--------|------|--------------|-----------|

**Selected LUNs**

| LUN ID | UUID | Multipathing | Dev. Size |
|--------|------|--------------|-----------|
|--------|------|--------------|-----------|

**Figure 6.11: RHEV Storage**

5. On the “New Data Center – Guide Me” Screen
  1. Click “Attach Storage”
  2. Select the “Storage Domain That Was Just Created”
  3. Click “OK”

At this point, the Data center should have a green arrow pointing upwards. This means that the RHEV environment is fully functional. Refer to the RHEV install guide<sup>xii</sup> for more information on configuration and troubleshooting.

## 6.12 Configure ISO Domain

For this section, attach an existing Network File System (NFS) share to RHEV or deploy a virtual machine. Use `virt-manager` to deploy Red Hat Enterprise Linux 6 NFS virtual machine via PXE using the `rhel6_base_virtual_machine` kickstart profile created above. The NFS export directory must be configured for read write access and must be owned by `vdsm:kvm`. If these users do not exist on the external NFS server, owner/group should be set to `36:36` for RHEV functionality:

```
chown 36:36 /mnt/nfs
```

1. Click on the “Storage” tab on the upper navigation menu
2. Click on the “New Domain” button on upper navigation menu
  1. Provide “Name”
  2. Select “NFS” and “ISO”
  3. Provide the “Export Path”
    1. At this point, the storage will show up “Unattached”
3. Click on the “Data Centers” tab in the upper navigation tab.
  1. Click on “Attach ISO” on the lower navigation panel
  2. Select the ISO domain to connect to
  3. Click “OK” as shown in **Figure 6.12: RHEV ISO Domain**



### New Domain

Name:

Domain function:  Data  ISO  Export

Storage type:  NFS

---

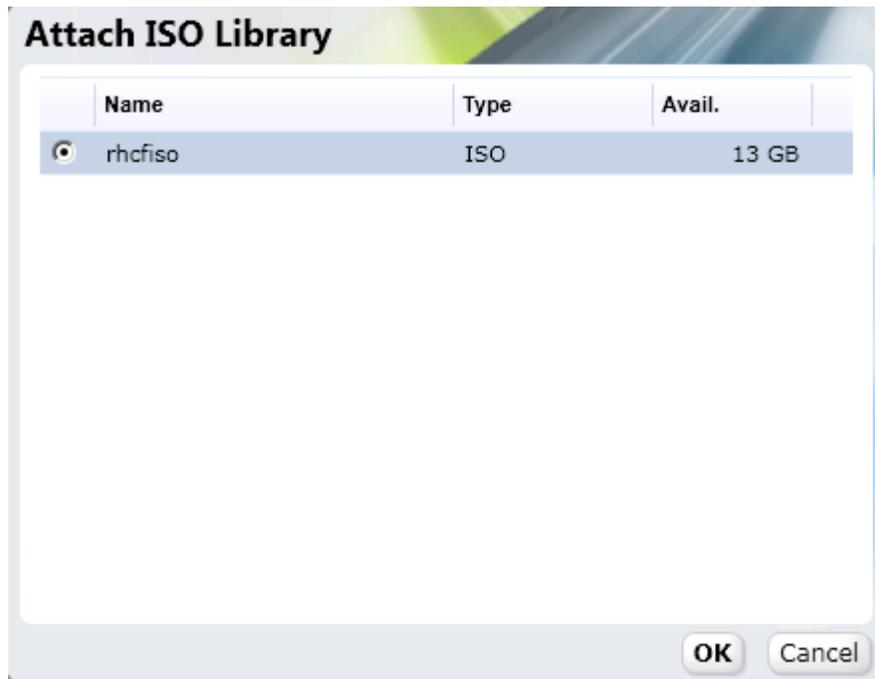
Use host:

Export path:

**Figure 6.12: RHEV ISO Domain**

4. Attach the ISO domain to the data center
  1. Click on the “Data Centers” tab in the upper navigation menu
    1. Select the data center
    2. Click the “Storage” tab on the lower navigation menu
    3. Click the “Attach ISO” button

4. Select the ISO domain that was just created, click OK as shown in **Figure 6.13: RHEV Attach ISO**

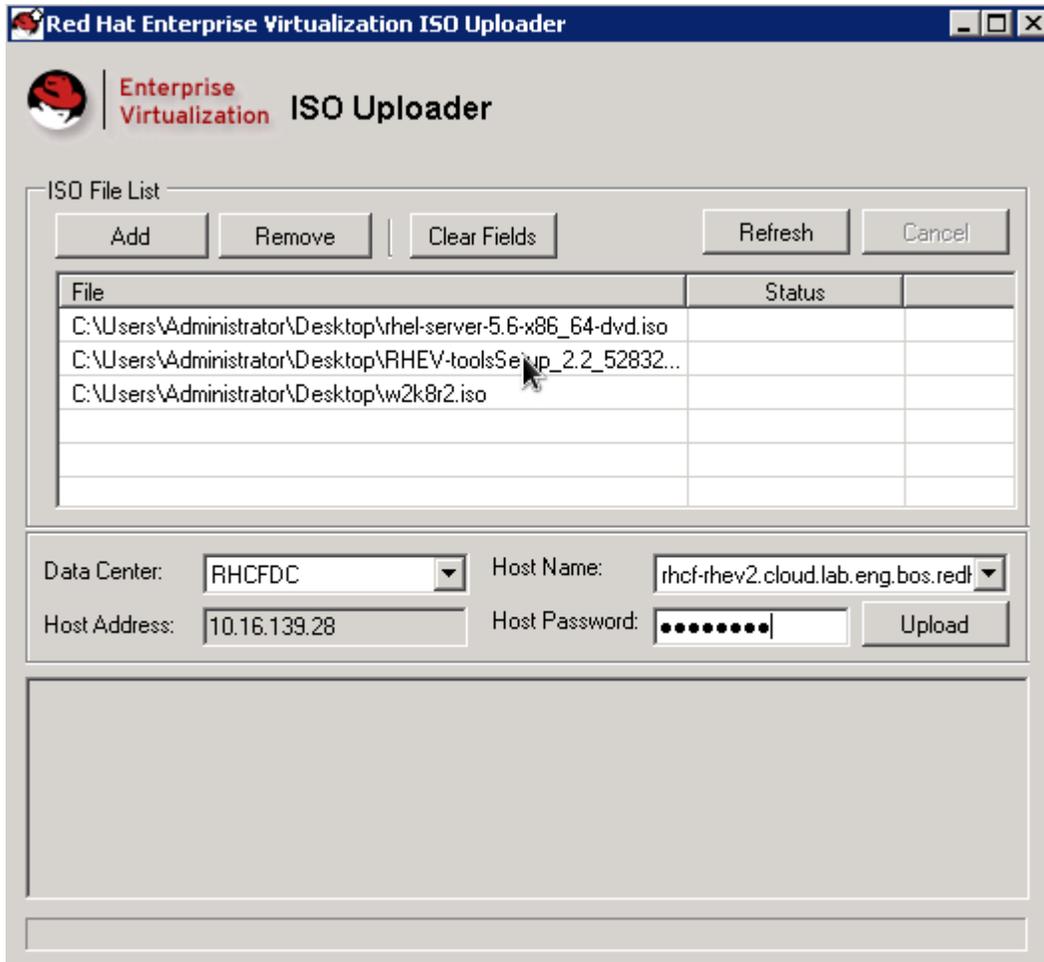


**Figure 6.13: RHEV Attach ISO**

5. Click the “Activate” button on the lower navigation menu

### Upload the RHEV Guest Tools ISO

1. Click on “Start” | “All Programs” | “Red Hat” | “RHEV Manager” | “ISO Uploader”
2. On the ISO Uploader tool
  1. Click “Add”
  2. Browse to the ISO files for
    - RHEV Guest Tools
    - Microsoft Windows Server 2008R2
    - RHEL 5.6
  3. Enter the password
4. Click “Upload” as shown in **Figure 6.14: RHEV ISO Uploader**



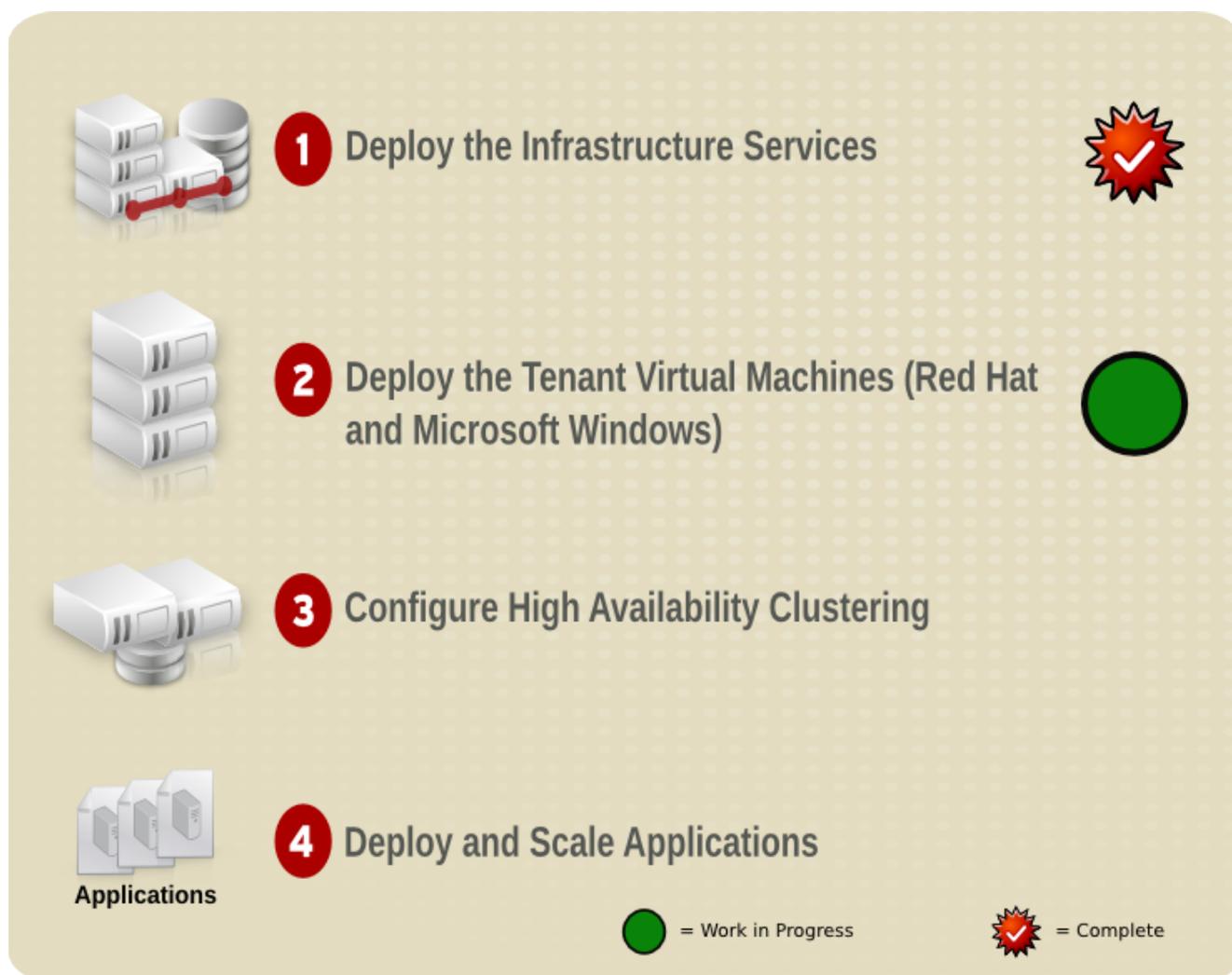
**Figure 6.14: RHEV ISO Uploader**

5. Close the ISO Uploader

# 7 Deploy Tenant Virtual Machines

## 7.1 Overview

This section provides the details on how to deploy Red Hat Enterprise Linux virtual machines via PXE, ISO and template. Also in this section details on how to deploy Microsoft Windows virtual machines via ISO and template are discussed. **Figure 7.1: Installation Overview** shows the current step in the overall process.



## 7.2 Deploy Red Hat Enterprise Linux 5.6 on RHEV VM Via PXE

Target System: rhevm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu



1. Provide a “Name”
2. Provide a “Description”
3. Change “Memory Size” to “1024 MB”
4. Change the “Operating System” to “Red Hat Enterprise Linux 5.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
  2. Click “OK”
5. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
6. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
7. Right click the virtual machine and select “Run Once”
  1. In the “Boot Sequence” panel, select “Network (PXE)” and move it up
  2. Click “OK”
8. Click the “Console” button in the top navigation menu and perform the PXE installation

## 7.3 Deploy Red Hat Enterprise Linux 5.6 on RHEV VM Via ISO

Target System: rhevm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name” e.g., test-vm
  2. Provide a “Description”
  3. Change “Memory Size” to “4GB”
  4. Change the “Operating System” to “Red Hat Enterprise Linux 5.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
  2. Click “OK”
5. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”

6. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
7. Right click the virtual machine and select “Run Once”
  1. Click the box “Attach CD” and choose the Red Hat Enterprise Linux media
  2. In the “Boot Sequence” panel, select “CD-ROM” and move it up
  3. Click “OK”
8. Click the “Console” button in the top navigation menu and perform the ISO installation

## 7.4 Deploy Red Hat Enterprise Linux 5.6 on RHEV via Template

Target System: rhevm-m

1. Log into the VM that was installed via PXE and run `sys-unconfig`

```
sys-unconfig
```

2. Right click on the powered off virtual machine and select “Make Template”
  1. Provide a “Name”
  2. Click “OK” after the template is finished
3. Click “New Server” in the top navigation menu
  1. Provide “Name”
  2. Provide “Description”
  3. On “Based on Template” choose the template that was just created
4. On the “High Availability” tab on the left navigation menu
  1. Check the “Highly Available” box
  2. Click “OK”
5. Launch the VM after the installation is finished
  1. Select the VM that was deployed from the template and right click and select “Run”
  2. Right click on the VM and choose “Console”
  3. Once logged into the VM, register it with the satellite server

## 7.5 Deploy Microsoft Windows 2008 R2 VM Via ISO

Target System: rhevm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu



3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name”
  2. Provide a “Description”
  3. Change “Memory Size” to “1024 MB”
  4. Change the “Operating System” to “Windows 2008 R2”
    1. This adds another tab on the left navigation pane, “Windows Sys. Prep.”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
  2. Click “OK”
5. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
6. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
7. Upload the VirtIO Drivers that were downloaded earlier with the ISO uploader
  1. virtio-driver-1.0.0-45801.vfd
8. Right click the virtual machine and select “Run Once”
  1. Click the box “Attach CD” and choose the Microsoft Windows media
  2. Click the box “Attach Floppy” and choose the VirtIO drivers that were uploaded
  3. In the “Boot Sequence” panel, select “CD-ROM” and move it up
  4. Click “OK”
9. Click the “Console” button in the top navigation menu and perform the ISO installation
10. Take the defaults or pick the required options until the “Where do you want to install Windows?” screen
  1. Click “Load Driver” and browse to the drivers and add SCSI drivers. The Ethernet drivers will be loaded later demonstrating RHEV tools
11. Complete the install
12. Install the Ethernet drivers and the agent
  1. Right click on the Microsoft Windows virtual machine and use “Change CD” to select the “RHEV-toolsSetup” ISO
  2. Open the console to the Microsoft Windows virtual machine

3. Browse to the CD-ROM using Microsoft Windows Explorer and launch the RHEV-ToolsSetup executable
  1. Take the defaults and complete the install.
    - Instead, it will run when derived VMs first boot, injecting the desktop drivers and tools they required (SPICE, etc...)

## 7.5.1 Sysprep Microsoft Windows VM and Create a Template

Target System: windows-vm

1. Install RHEV APT (Application Provisioning Tools) on the Virtual Machine, but do not start the service. RHEV APT comes from the RHEV-toolsSetup ISO
  - The RHEV APT tool will inject the appropriate drivers automatically when the VM reboots
2. Launch the System Preparation wizard
  1. Click the “Start” button and enter the following in the “Search Programs and Files” box

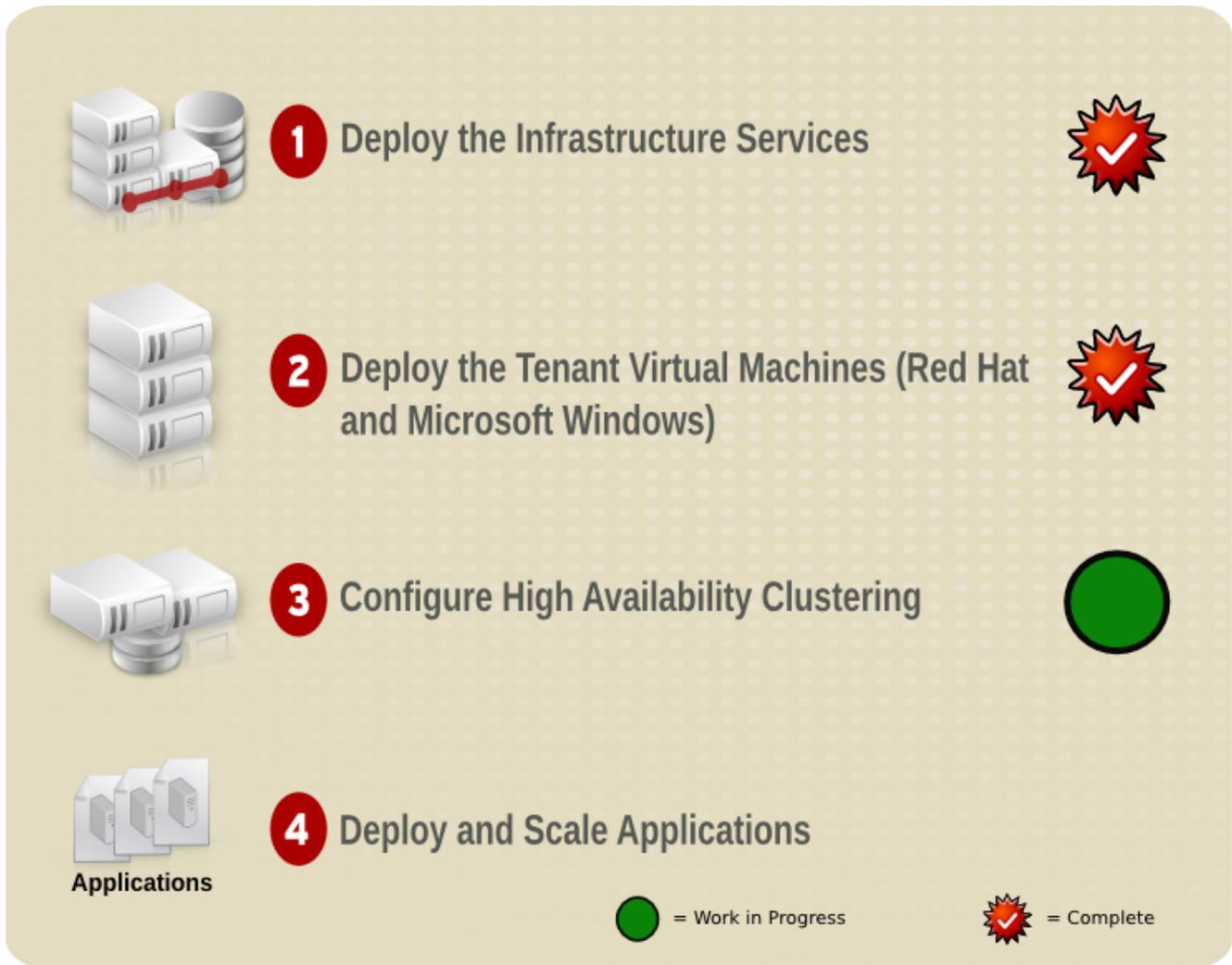
```
c:\windows\system32\sysprep\sysprep.exe
```

3. On the Sysprep wizard select “Enter System Out-of-Box Experience (OOBE)” and check the box “Generalize”
  1. Change the shutdown option to “Shutdown”
    1. Click “OK”
4. Right click on the powered off virtual machine and select “Make Template”
  1. Provide a “Name”
  2. Click “OK”
5. Click “New Server” in the top navigation menu
  1. Provide “Name”
  2. Provide “Description”
  3. On “Based on Template” choose the template that was just created
6. On the “High Availability” tab on the left navigation menu
  1. Check the “Highly Available” box
  2. This can be avoided if the Template is edited and the HA box is checked. However, some VMs may not need to run HA, so it is disabled by default.
  3. Click “OK”
7. Launch the VM
  1. Select the VM that was deployed from the template and right click and select “Run”
  2. Right click on the VM and choose “Console”



# 8 Configure High Availability Environment

This section covers setting up the Red Hat Enterprise Linux High Availability environment. The overview is to install luci on a dedicated RHEV virtual machine, then install the second bare metal server and deploy the cluster software onto it. Once that is finished, cluster the virtual machines, make sure the XML cluster configuration files are on both nodes and test failover. **Figure 8.1: High Level Overview** shows the current step in the overall process.



## 8.1 Install Luci Virtual Machine

For the networking configuration, there are a few options. The first option is to have all the cluster communications run over a single network. That is the way this reference architecture is configured. The second way is to separate the networks and create a “private” and “public” network. The only requirement is to make sure that the cluster communication network has access to the fence devices. Please refer to the installation guide<sup>xiii</sup> for more information.

Target System: rhelv-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name” e.g., luci-vm
  2. Provide a “Description”
  3. Change “Memory Size” to “1024 MB”
  4. Change the “Operating System” to “Red Hat Enterprise Linux 6.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
  2. Click “OK”
5. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
6. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
7. Right click the virtual machine and select “Run Once”
  1. In the “Boot Sequence” panel, select “Network (PXE)” and move it up
  2. Click “OK”
8. Click the “Console” button in the top navigation menu and perform the PXE installation using the kickstart profile that has the Red Hat Enterprise Linux HA Activation channel associated with it
9. Ensure there is a DNS entry for the luci-vm

## Install Luci

Target System: luci-vm

1. Install luci

```
yum -y install luci
```

2. Start the luci service

```
service luci restart
chkconfig luci on
```

3. Configure iptables with the ports shown in **Table 10: Luci Firewall**



| IP Port Number | Protocol  | Component |
|----------------|-----------|-----------|
| 8084           | TCP / UDP | Luci      |
| 11111          | TCP / UDP | ricci     |

**Table 10: Luci Firewall**

```
./firewall-config.sh
Please put the ports you would like to firewall here, separated by a space:
8084 11111
Please put the protocols you would like to firewall here, separated by a
space:
tcp udp
iptables: Chain already exists.
iptables --append RHCF --protocol tcp --destination-port 8084 --jump ACCEPT
iptables --append RHCF --protocol tcp --destination-port 11111 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 8084 --jump ACCEPT
iptables --append RHCF --protocol udp --destination-port 11111 --jump ACCEPT

Remember to "service iptables save"
```

4. Log into the Luci web interface using the link provided and log in to test

## 8.2 Install Cluster nodes

Flow of operations for this section:

1. Install second bare metal server
  1. Configure the bridge networking
2. Register mgmt1 with internal satellite server
3. Follow SAN providers instructions and ensure the LUN is presented to both cluster nodes
  - mgmt1
  - mgmt2
4. Set up the cluster

### 8.2.1 Install mgmt2, Configure Networking on mgmt2, Register mgmt1 with Satellite

#### Install second bare metal server

The second bare metal server will serve as the second cluster node. This node provides a second host for the virtual machines to fail over to.

Target System: mgmt2

1. PXE boot the second bare metal server and install with the cluster profile and use the rhel6\_clustering kickstart profile created earlier.
2. Configure bridge networking for mgmt2
3. Copy the ifcfg-rhcf interface file from mgmt1 to mgmt2

```
ssh mgmt1 cat /etc/sysconfig/network-scripts/ifcfg-rhcf > /etc/sysconfig/network-scripts/ifcfg-rhcf
```

4. Edit the local ifcfg-eth0 file to reflect the new bridge

```
DEVICE="eth0"
BOOTPROTO="static"
HWADDR="00:1A:64:76:00:08"
ONBOOT="yes"
IPADDR=10.16.139.21
NETMASK=255.255.248.0
BRIDGE=rhcf
```

5. Restart networking

```
service network restart
```

6. Check the bridge

```
brctl show
```

7. Install virtualization software

```
yum install qemu-kvm libvirt virt-manager
```

8. Start libvirtd

```
service libvirtd restart
chkconfig libvirtd on
```

## Register mgmt1 with internal satellite server

Target System: sat-vm

1. Bootstrap mgmt1 to the internal satellite server

```
rhn-bootstrap --activation-keys 1-cluster --allow-config-actions --allow-remote-commands --script bootstrap_cluster_node.sh
```

1. This will produce some verbose output, with a path to bootstrap\_cluster\_node.sh, copy that path for the next command



```
cat /var/www/html/pub/bootstrap/bootstrap_cluster_node.sh | ssh rhcf-
mgmt1 /bin/bash
```

## Follow SAN providers instructions and ensure the LUN is presented to both cluster nodes

Here the SAN LUN that mgmt1 has used to install the virtual machines to will need to be presented to mgmt2, verify both servers can see the LUN. Follow your storage manufacturer's guidelines to ensure the storage environment is set up properly.

### 8.2.2 Set up the Cluster

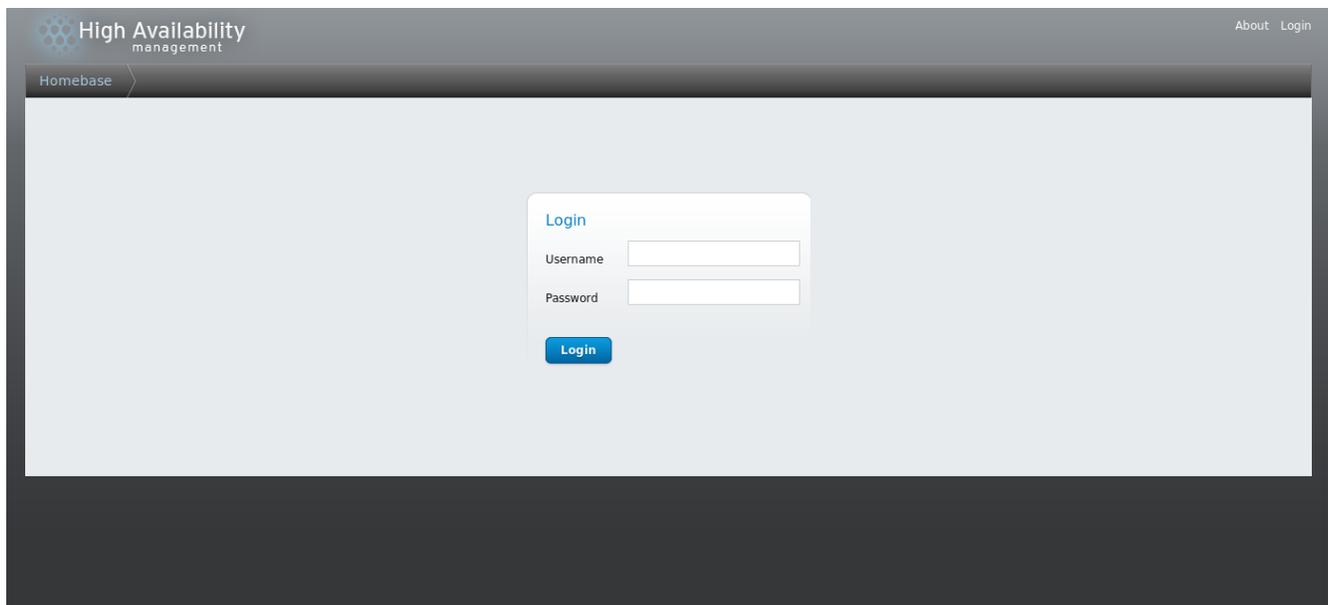
1. Install the cluster management software

```
yum install ricci
```

2. Start the service and ensure it turns on when the system is booted

```
service ricci start && chkconfig ricci on
```

3. Log into the luci web based management interface as shown in **Figure 8.2: Luci Login**



**Figure 8.2: Luci Login**

4. Click on “Manage Clusters” and click “Create” and add the nodes and click “Create Cluster” as shown in **Figure 8.3: Luci Create Cluster**

## Create a Cluster ✕

Cluster Name

Use same password for all nodes

| Node Hostname                               | Root Password                         | Ricci Port                                                |
|---------------------------------------------|---------------------------------------|-----------------------------------------------------------|
| <input type="text" value="mgmt1.rhcf.lab"/> | <input type="password" value="●●●●"/> | <input style="width: 40px;" type="text" value="11111"/> ✕ |
| <input type="text" value="mgmt2.rhcf.lab"/> | <input type="password" value="●●●●"/> | <input style="width: 40px;" type="text" value="11111"/> ✕ |

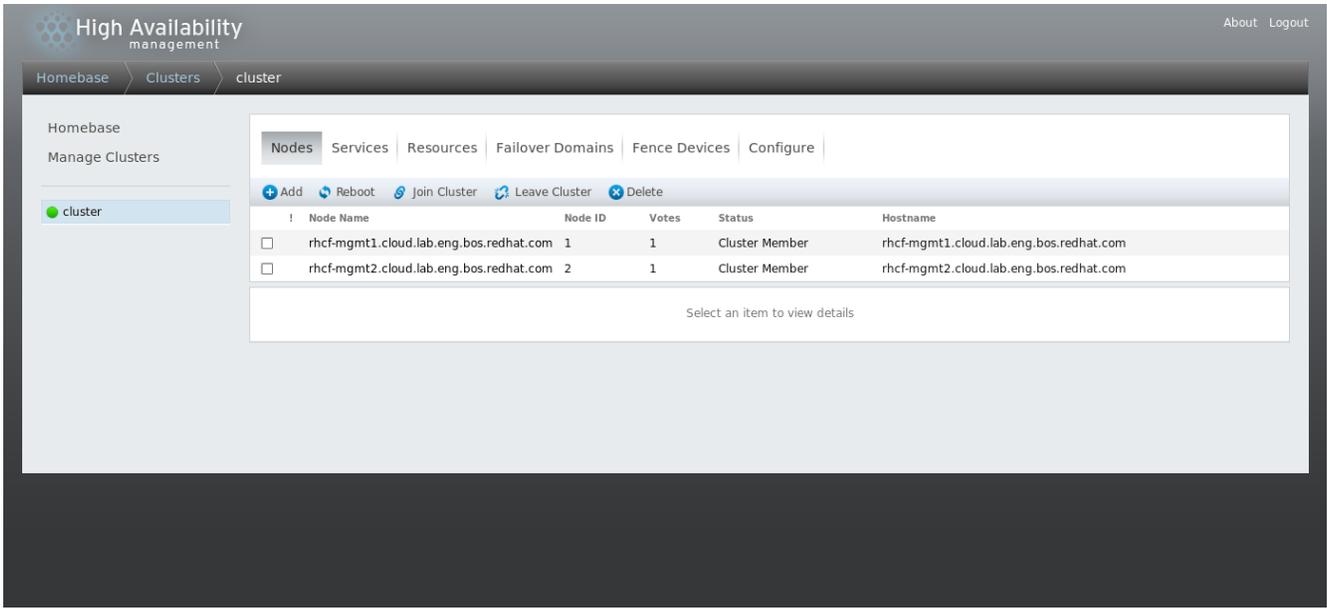
Download Packages  
 Use locally installed packages

Reboot nodes before joining cluster

Enable shared storage support

**Figure 8.3: Luci Create Cluster**

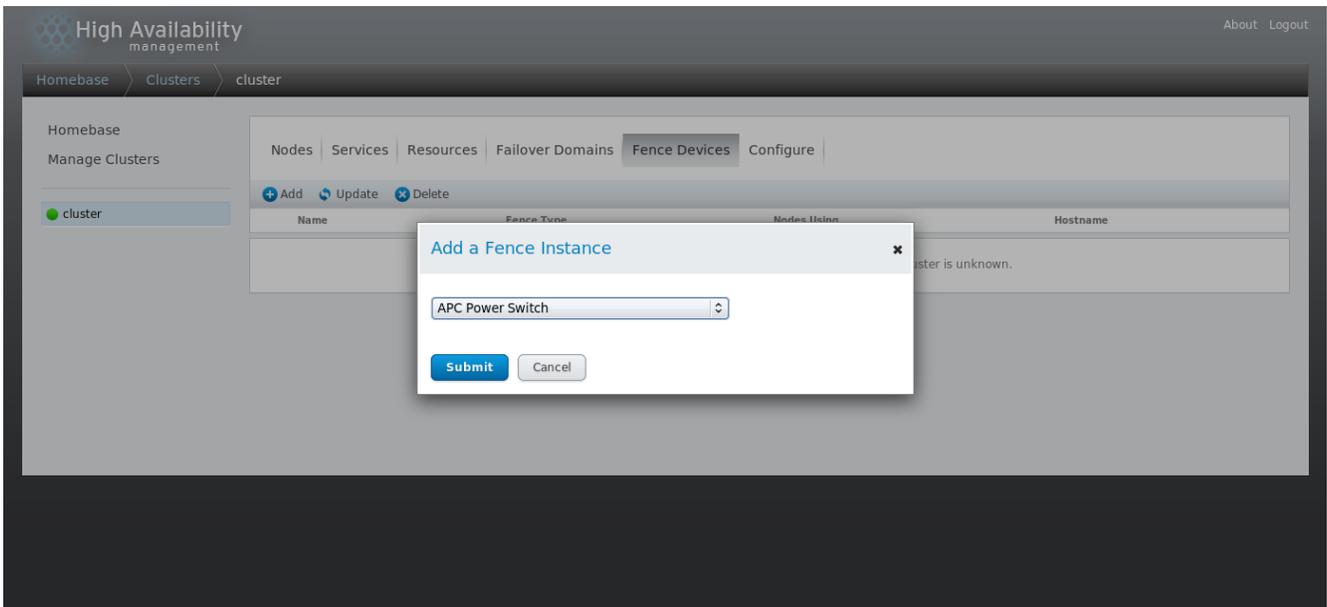
- Once the cluster is added, click on the cluster name the nodes will be listed as shown in **Figure 8.6: Luci Failover Domain**



**Figure 8.4: Luci Node List**

5. Configure fencing for the cluster

1. Click on “Fence Devices” on the upper navigation menu, click “Add” as shown in **Figure 8.5: Luci Fence Device**



**Figure 8.5: Luci Fence Device**

6. Create a failover domain, click on “Failover Domain” in the upper navigation menu, click “Add” as shown in **Figure 8.6: Luci Failover Domain**

### Create a Failover Domain ✕

Name

Prioritized      Order the nodes to which services failover.

Restricted      Service can run only on nodes specified.

No Failback      Do not send service back to 1st priority node when it becomes available again.

|                                         | Member                              | Priority                                  |
|-----------------------------------------|-------------------------------------|-------------------------------------------|
| rhcf-mgmt1.cloud.lab.eng.bos.redhat.com | <input checked="" type="checkbox"/> | <input style="width: 40px;" type="text"/> |
| rhcf-mgmt2.cloud.lab.eng.bos.redhat.com | <input checked="" type="checkbox"/> | <input style="width: 40px;" type="text"/> |

**Figure 8.6: Luci Failover Domain**

7. Share SSH keys between the two hosts (perform this on both nodes)

1. Target System: mgmt1

```
ssh-keygen -t rsa
ssh-copy-id -i .ssh/id_rsa.pub mgmt2
```

2. Target System: mgmt2

```
ssh-keygen -t rsa
ssh-copy-id -i .ssh/id_rsa.pub mgmt1
```

## 8.3 Cluster Virtual Machines

Target System: Luci web interface and mgmt1

### Create HA Service for sat-vm

1. Shut down the virtual machines and copy virtual machine XML files from mgmt1 to mgmt2

```
scp /etc/libvirt/qemu/nfs-vm.xml rhcf-mgmt2:/etc/libvirt/qemu/
scp /etc/libvirt/qemu/sat-vm.xml rhcf-mgmt2:/etc/libvirt/qemu/
scp /etc/libvirt/qemu/rhev-vm.xml rhcf-mgmt2:/etc/libvirt/qemu/
scp /etc/libvirt/qemu/dnsdhcp-vm.xml rhcf-mgmt2:/etc/libvirt/qemu/
```

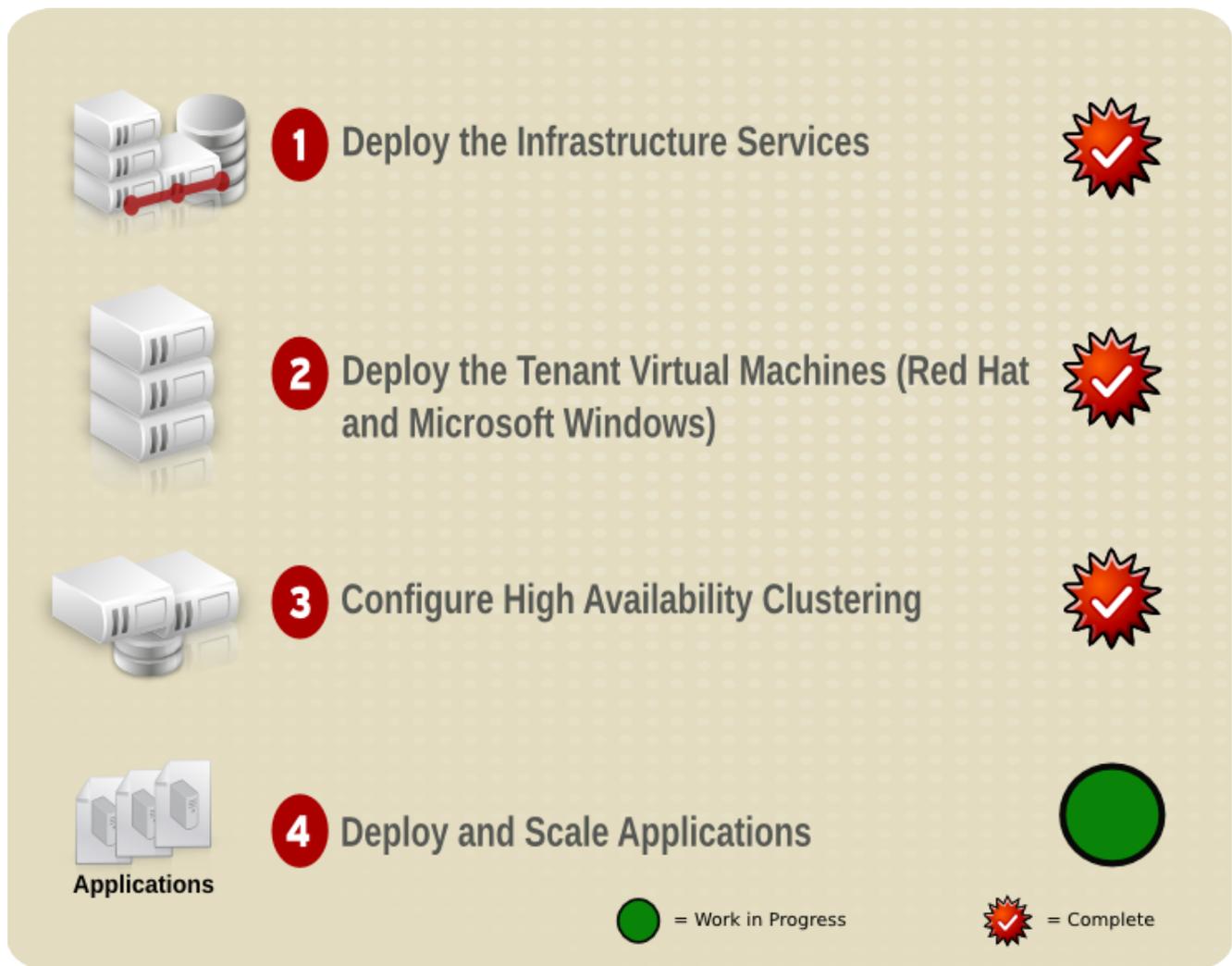
2. From luci home page, Click on “Cluster”
3. Click “Services”
4. Click “Add Virtual Machine Service”



5. Provide "Service Name"
6. Check the box "Automatically Start this Service"
7. Select the "Failover Domain"
8. Click "Add a Resource"
  1. Select "Virtual Machine"
    1. Provide "VM Configuration file path"
    2. Click "Submit"
9. Repeat this process for each virtual machine
10. Check the virtual machines status on the command line

# 9 Deploy and Scale Applications

This section discusses deploying and scaling three types of applications. The first application is a generic Java based application that is used to demonstrate ease of deployment and scaling of a simple application. The second software stack is the Jboss EAP framework. This application is a bit more complicated but using technologies like Red Hat Network satellite server and scripts the process can still be easy. Finally, the last application is a MRG perfect number based application. This application demonstrates ease of configuration and deployment of MRG software. **Figure 9.1: High Level Overview** shows current status of the overall process.



## 9.1 Deploy Java Application

This section requires access to the `javaApp` package that was downloaded earlier.



## 9.1.1 Configure GPG and Sign the javaApp package

Target System: sat-vm

1. Create GPG key for RPM

```
gpg --gen-key
```

1. Take the defaults
  2. DSA
  3. 2048 Bits
  4. Doesn't expire
  5. Provide "Real Name"
  6. Provide "Email Address"
  7. Provide "Comment"
2. Get the GPG key

```
gpg --list-keys --fingerprint
/root/.gnupg/pubring.gpg
```

```

pub 1024D/F24F1B08 2002-04-23 [expired: 2004-04-22]
Key fingerprint = D8CC 06C2 77EC 9C53 372F C199 B1EE 1799 F24F
1B08
uid Red Hat, Inc (Red Hat Network) <rhn-
feedback@redhat.com>

pub 1024D/25AA0649 2011-03-19
Key fingerprint = C663 9B55 4CC5 985B F5CA BA15 1080 39D1 25AA
0649
uid Scott Collier (application key)
<scollier@redhat.com>
sub 2048g/D4C4DF51 2011-03-19
```

3. Place the GPG key in ~/.rpmmacros with the following content

```
cat ~/.rpmmacros
%_signature gpg
%_gpg_name C15579CF
```

4. Resign the rpm

```
rpm --resign javaApp-2-0.noarch.rpm
```

5. Check the signature on the rpm

```
rpm --checksig javaApp-2-0.noarch.rpm
```

6. Export the key so clients can import it

```
gpg --export --armor C15579CF > APP-RPM-GPG-KEY
```

7. Import the key

```
rpm --import APP-RPM-GPG-KEY
```

8. Check the signature on the rpm again

```
rpm --checksig javaApp-2-0.noarch.rpm
```

9. Make the key available

```
cp APP-RPM-GPG-KEY /var/www/html/pub/.
```

## 9.1.2 Set up Software Channel on Satellite Server

Target System: sat-vm RHN web interface

1. Click on the “Channels” tab in the upper navigation menu
2. On the left navigation bar, click “Manage Software Channels”
3. On the right navigation bar, click “Create New Channel”
4. Provide information for:
  - “Channel Name”
  - “Channel Label”
  - “Parent Channel”
  - “Parent Channel Architecture”
  - “Channel Summary”
  - “Organization Sharing” to “Public”
  - “GPG Key URL”
  - “GPG Key ID”
5. Click “Update Channel”

## 9.1.3 Upload Application

On the satellite server

1. Use `rhnpush` to populate the `rhcf-apps` software channel and enter a password when prompted

```
rhnpush --channel=rhcf-apps \
--server=http://sat-vm.rhcf.local --username=admin \
--password=
```



## 9.1.4 Create RHN Activation Key for Custom Channel

On the RHN Satellite Server

Target System: RHN web UI

1. On the upper navigation bar, click on “Systems” tab
  1. On the left navigation bar, click on “Activation Keys”
  2. On the right navigation bar, click on “Create New Key”
2. Provide information for
  - “Description”
  - Key
  - Select “Provisioning” and “Virtualization” from “Add-on Entitlements”
3. Click “Create Activation Key”
  1. Click “Child Channels” tab on the top of the page
  2. Select the “RHCF Apps” Channel
4. Click “Update Key”

## 9.1.5 Create a New Kickstart Profile

On the RHN Satellite Server

1. Click on “Systems” tab on the top navigation bar
2. Click “Kickstart” on the left navigation bar
3. On right navigation bar, click “Create New Kickstart Profile”
4. Provide Information for
  - “Label”
  - Change “Base Channel” to “Red Hat Enterprise Linux (v. 6 for 64-bit x86\_64)”
  - Change “Virtualization Type” to “KVM Virtualized Guest”
5. Click “Next”
6. Take the defaults for “Step 2”
7. Provide a password for “Step 3”
8. Click “Finish”
9. Click “Activation Keys” on the “Kickstart: rhel-java-apps” screen
10. Check “RHCF Custom Applications”
11. Click “Update Activation Keys”

12. Click “Update Activation Keys” button
13. Click “Scripts” in the upper navigation menu
  1. Click “Add New Kickstart Script” on the upper navigation menu
    1. Provide “Scripting Language: /bin/bash”
    2. Provide the following for “Script Contents”

```
wget http://x.x.x.x/pub/APP-RPM-GPG-KEY -O /etc/pki/rpm-gpg/APP-RPM-GPG-KEY
rpm --import /etc/pki/rpm-gpg/APP-RPM-GPG-KEY

yum -y install javaApp
```

3. Change “Script Execution Time” to “Post Script”
4. Click “Update Kickstart”

## 9.1.6 Deploy Virtual Machine with javaApp via PXE

Target System: rhevm-vm

1. Click “New Server”
2. On the “General” tab, provide Information for
  - “Name”
  - “Description”
3. On the “Boot Sequence” tab, provide information for
  - Change “Second Device” to “Network (PXE)”
4. Click “OK”
5. On the “New Virtual Machine – Guide Me” screen,
  1. Click “Configure Network Interfaces”
    - Take the defaults and click “OK”
  2. Click “Configure Virtual Disks”
  3. On the “New Virtual Disk” screen, provide information for
    - Set “Size(GB)” to 10
    - Check “Wipe After Delete”
  4. Click “OK”
  5. On the “New Virtual Machine – Guide Me” screen, click “Configure Later”
6. Boot the “rhcf-java-app-vm” virtual machine
7. On the top navigation bar, click the “Run” button
8. On the top navigation bar, Click on the “Console” button
9. When the virtual machine boots, select the appropriate entry from the PXE menu.



10. After the virtual machine is installed, log in and verify javaApp is running

```
pgrep javaApp
```

### 9.1.7 Create a Template from the javaApp Virtual Machine

Target System: rhevm-vm

1. Make sure the virtual machine is stopped, right click on the virtual machine
  1. Choose “Make Template”
  2. Provide a “Name”
  3. Provide a “Description”
  4. Click “OK”

### 9.1.8 Scale the javaApp Virtual Machine

Target System: rhevm-m

1. Open the RHEV-M Scripting library
2. Log in and launch script add-vms.ps1 **Appendix A**
3. Launch the following command to instantiate 5 virtual machines with javaApp installed

```
.\add-vms.ps1 -baseTemplate t_javaApp -baseName t_javaApp -num 5 -run
```

4. Log on to the virtual machines and ensure javaApp is running

```
pgrep javaApp
```

## 9.2 Deploy and Scale JBoss EAP Application

In this section, JBoss Operations network will be deployed to demonstrate how easy it is to manage JBoss servers. A new method of provisioning the virtual machine involving the RHEVM scripting engine will be explored as well. Here are the overall steps:

### Copy scripts and JON to sat-vm

1. Copy JBoss ON 2.4.1 to `/var/www/html/pub/kits` from the software repository that was set up earlier.
2. Copy `rh-server.sh` to `/var/www/html/scripts`
3. Copy the JBoss ON license

```
cp licenseFile /var/www/html/scripts/jon-license.xml
```

### Create Activation key for JBoss ON Server

Target System: Browser connected to RHN server

1. Click the “Systems” tab on the upper navigation menu

2. Click “Activation Keys” on left navigation menu
3. Click “Create New Key” on upper navigation menu
4. Provide the following information
  1. Description: “jon-server”
  2. Key: “jon”
  3. Click “Create Activation Key”
5. On the jon-server Activation Key screen, click the “Packages” tab in the upper navigation menu
  1. Add the following packages
    - postgresq
    - postgresql-server
    - java-1.6.0-openjdk.x86\_64
  2. Click “Update Key”

## **Create a Red Hat Enterprise Linux 6 Base Virtual Machine Kickstart Profile**

Target System: Browser connected to RHN server

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
  1. Provide a “Label” for the kickstart “jon-server”
  2. On “Base Channel” select “Red Hat Enterprise Linux (v. 6 for 64-bit x86\_64)”
  3. On “Kickstartable Tree” select “ks-rhel-x86\_64-server-6-6.0”
  4. On “Virtualization Type” select “KVM Virtualized Guest”
  5. Click “Next”
5. On “Step 2”
  1. Click “Next”
6. On “Step 3”
  1. Provide a root password
7. Click “Finish”
8. On the upper navigation menu, click “Kickstart Details”, then “Details”
  1. Change “Virtual Bridge” to “rhcf”
  2. Check “Log custom post scripts”
9. Click “Update Kickstart”



10. Click “Activation Keys” in the upper navigation menu
  1. Select “jon-server”
  2. Click “Update Activation Keys”
11. Add a new Post script
  1. Click “Scripts” in the upper navigation menu
  2. Click “Add new kickstart script”
    1. Add the following to the “Script Contents” section (use the script that was downloaded in “Section 5.3 Downloading Software”

```
wget http://sat-vm.rhcf.lab/pub/scripts/rhq-install.sh -O /tmp/rhq-install.sh
chmod +x rhq-install.sh
./rhq-install.sh
```

2. Change “Script Execution Time” to “Post Script”
3. Click “Update Kickstart”

## 9.2.1 Provision JBoss ON Server on RHEV

Target System: rhevm-vm

1. Click “New Server”
2. On the “General” tab, provide Information for
  - “Name”
  - “Description”
  - 4GB of Memory
  - 2 CPU Sockets
  - 1 Core
3. On the “Boot Sequence” tab, provide information for
  - Change “Second Device” to “Network (PXE)”
4. Click “OK”
5. On the “New Virtual Machine – Guide Me” screen,
  1. Click “Configure Network Interfaces”
    - Take the defaults and click “OK”
  2. Click “Configure Virtual Disks”
  3. On the “New Virtual Disk” screen, provide information for
    - Set “Size(GB)” to 15
    - Check “Wipe After Delete”
  4. Click “OK”

5. On the “New Virtual Machine – Guide Me” screen, click “Configure Later”
6. Boot the “jon-server” virtual machine
7. On the top navigation bar, click the “Run” button
8. On the top navigation bar, Click on the “Console” button
9. When the virtual machine boots, select the appropriate entry from the PXE menu.
12. When the jon-server reboots, log in and check the rhq service

```
service rhq-server.sh status
```

13. Check the install log to ensure everything completed

```
cat ks-post.log
```

14. Configure iptables

| IP Port Number | Protocol  | Component                  |
|----------------|-----------|----------------------------|
| 16163          | TCP / UDP | Server to Agent            |
| 7080           | TCP / UDP | Agent to Server (Standard) |
| 7443           | TCP / UDP | Agent to Server (Secure)   |
| 5432           | TCP / UDP | PostgreSQL                 |

```
./firewall-config.sh
```

```
Please put the ports you would like to firewall here, separated by a space:
```

```
16163 7080 7443 5432
```

```
Please put the protocols you would like to firewall here, separated by a space:
```

```
tcp udp
```

```
iptables: Chain already exists.
```

```
iptables --append RHCF --protocol tcp --destination-port 16163 --jump ACCEPT
```

```
iptables --append RHCF --protocol tcp --destination-port 7080 --jump ACCEPT
```

```
iptables --append RHCF --protocol tcp --destination-port 7443 --jump ACCEPT
```

```
iptables --append RHCF --protocol tcp --destination-port 5432 --jump ACCEPT
```

```
iptables --append RHCF --protocol udp --destination-port 16163 --jump ACCEPT
```

```
iptables --append RHCF --protocol udp --destination-port 7080 --jump ACCEPT
```

```
iptables --append RHCF --protocol udp --destination-port 7443 --jump
```



```
ACCEPT
iptables --append RHCF --protocol udp --destination-port 5432 --jump
ACCEPT
```

Remember to "service iptables save"

## Verify JBoss ON Server Operations and Cluster the JBoss ON Server

Target System: Any machine with a browser

1. Open Browser, hit JON IP address "x.x.x.x:7080"
2. Log in with Username: rhqadmin Password: rhqadmin

## 9.3 Deploy JBoss Enterprise Application Platform

### Create a activation key for the JBoss Server

Target System: Browser connected to RHN server

1. Click the "Systems" tab on the upper navigation menu
2. Click "Activation Keys" on left navigation menu
3. Click "Create New Key" on upper navigation menu
4. Provide the following information
  1. Description: "jboss-eap"
  2. Key: "jboss-eap"
  3. Add-On Entitlements: "Provisioning"
  4. Click "Create Activation Key"
5. On the jboss-eap Activation Key screen, click the "Child Channels" tab in the upper navigation menu
  1. Keep the defaults and select the following channels
    - "JBoss Application Platform (v 5.1.0) for 5 Server x86\_64)
    - "RHEL Supplementary (v. 5 for 64-bit x86\_64)"
  2. Click "Update Key"

### Create Kickstart Profile for jboss-server

Target System: rhevm-vm

1. Click "Systems" in the top navigation pane
2. On the left navigation pane click "Kickstart"
3. In the "Kickstart Actions" pane click "Create a New Kickstart Profile"
4. On "Step 1"
  1. Provide a "Label" for the kickstart "jboss-eap"
  2. On "Base Channel" select "Red Hat Enterprise Linux (v. 5 for 64-bit x86\_64)

3. On “Kickstartable Tree” select “ks-rhel-x86\_64-server-5-u6”
4. On “Virtualization Type” select “KVM Virtualized Guest”
5. Click “Next”
5. On “Step 2”
  1. Click “Next”
6. On “Step 3”
  1. Provide a root password
7. Click “Finish”
8. Click “Activation Keys” in the upper navigation menu
  1. In the “Kickstart Details” screen, select the “jboss-eap” key
  2. Click “Update Activation Keys”

## Deploy JBoss Server With Application Automation

Target System: rhvm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name”
  2. Provide a “Description”
  3. Change “Memory Size” to “1024 MB”
  4. Change the “Operating System” to “Red Hat Enterprise Linux 5.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
5. Click on the “Boot Sequence” tab on the left navigation pane
  1. Change “Second Device” to “Network (PXE)”
  2. Click “OK”
6. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
7. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”



8. Click the “Run” button in the top navigation menu and perform the install
9. Click the “Console” button in the top navigation menu and monitor the PXE installation

### Automate the deployment of JBoss by Adding jbossas to Activation Key

1. Change the activation key
2. Click “Systems” on the upper navigation menu
3. Click “Activation Keys” on the left navigation menu
4. Select the “jboss-eap” activation key
  1. Click “Packages” on the upper navigation menu
  2. Enter “jbossas” in the “Enter Package Names Below” window
  3. Click “Update Key”

### Automate the Kickstart by Adding post scripts

1. Click “Systems” on the upper navigation menu
2. Click “Kickstart” on left navigation menu
3. Click “View a list of Kickstart Profiles” in the “Kickstart Actions” pane
4. Select the “jboss-eap” kickstart profile
5. Click on “Scripts” on the upper navigation menu
6. Click “Add new Kickstart Script” in the upper navigation menu
  1. Add the following content

```
wget http://sat-vm.rhcf.lab/pub/scripts/jboss-eap-install.sh -O
/tmp/jboss-eap-install.sh
chmod +x /tmp/jboss-eap-install.sh
/tmp/jboss-eap-install.sh
```

2. Change “Script Execution Time” to “Post Script”
3. Click “Update Kickstart”
7. Add one more script and include the following content

```
wget http://sat-vm.rhcf.lab/pub/scripts/rhq-agent-install.sh -O
/tmp/rhq-agent-install.sh
chmod +x /tmp/rhq-agent-install.sh
/tmp/rhq-agent-install.sh
```

1. Change “Script Execution Time” to “Post Script”
2. Click “Update Kickstart”

## Deploy JBoss Server Application Automatically

Target System: rhevm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name”
  2. Provide a “Description”
  3. Change “Memory Size” to “1024 MB”
  4. Change the “Operating System” to “Red Hat Enterprise Linux 5.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
5. Click on the “Boot Sequence” tab on the left navigation pane
  1. Change “Second Device” to “Network (PXE)”
  2. Click “OK”
6. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
7. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
8. Click the “Run” button in the top navigation menu and perform the install
9. Click the “Console” button in the top navigation menu and monitor the PXE installation
10. After the virtual machine is installed, open a browser and check the Seam Application

## Check the JBoss ON Console for Automatic Registration

Target System: Browser with access to jon-server

1. Open a browser and open page “<http://x.x.x.x/7080>”
2. In the “Auto-discovery” pane, approve the virtual machine that was just installed

### 9.3.1 Scale JBoss Deployment

Remove the automatic registration kickstart script from the jboss-eap kickstart profile



Target System: Browser with access to RHN

1. Click “System” on the upper navigation menu
2. Select “Kickstart” on the left navigation menu
3. Select the “jboss-eap” kickstart profile
  1. Click “Clone Kickstart” on the upper navigation menu
  2. Provide a new “Kickstart Label”
  3. Click “Clone Kickstart”
4. Select the new Kickstart profile
  1. Click “Scripts” in the upper navigation menu
  2. Click “Delete Kickstart” in upper navigation menu
    1. Click “Delete Kickstart Script”

### Create a Template from the jboss-eap Virtual Machine

Target System: rhevm-vm

1. Make sure the virtual machine is stopped, right-click on the virtual machine
  1. Choose “Make Template”
  2. Provide a “Name”
  3. Provide a “Description”
  4. Click “OK”
    1. After installing a VM from a template, remember to register it with RHN.

### Scale the jboss-eap Virtual Machine

Target System: rhevm-m

1. Open the RHEV-M Scripting library
2. Log in and launch script add-vms.ps1 **Appendix B Scripts**
3. Launch the following command to instantiate 5 virtual machines with j javaApp installed

```
.\add-vms.ps1 -baseTemplate t_jbosseap -baseName t_jbosseap -num 5 -run
```

4. Log on to the virtual machines and ensure JBoss is running

```
service jbossas status
```

5. Open a browser and go to the seam-booking site and ensure functionality

## 9.4 Deploy and Scale Applications – MRG Manager

The goal of this section is to install and configure MRG Grid and successfully run jobs on multiple nodes and then interpret the results. For this section, the files that are needed are the `perfect_number.sub` file which serves as the job submission file and the `perfect.tgz` file which has the code that will be run on the hosts.

### Create a activation key for the MRG Manager

Target System: Browser with access to RHN

1. Click the “Systems” tab on the upper navigation menu
2. Click “Activation Keys” on left navigation menu
3. Click “Create New Key” on upper navigation menu
4. Provide the following information
  1. Description: “mrg-manager”
  2. Key: “mrg-manager”
  3. Add-On Entitlements: “Provisioning”
  4. Click “Create Activation Key”
5. On the mrg-manager Activation Key screen, click the “Child Channels” tab in the upper navigation menu
  1. Keep the defaults and select the following channels
    - “MRG Grid v. 1 (for RHEL 5 Server 64-bit x86\_64)”
    - “MRG Management v. 1 (for RHEL 5 Server 64-bit x86\_64)”
    - “MRG Messaging Base v. 1 (for RHEL 5 Server 64-bit x86\_64)”
  2. Click “Update Key”
6. On the mrg-manager Activation Key screen, click the “Packages” tab in the upper navigation menu and add the following packages
  - cumin
  - sesame
  - qpidd
  - condor-qmf-plugins
  - condor
  - qmf
  - postgresql-server
7. Click “Update Key”

### Create Kickstart Profile for mrg-manager

Target System: Browser with access to RHN



1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
  1. Provide a “Label” for the kickstart “rhel5\_base\_bare\_metal”
  2. On “Base Channel” select “Red Hat Enterprise Linux (v. 5 for 64-bit x86\_64)”
  3. On “Kickstartable Tree” select “ks-thrl-x86\_64-server-5-u6”
  4. On “Virtualization Type” select “KVM Virtualized Guest”
  5. Click “Next”
5. On “Step 2”
  1. Click “Next”
6. On “Step 3”
  1. Provide a root password
7. Click “Finish”
8. Click “Activation Keys” in the upper navigation menu
  1. In the “Kickstart Details” screen, select the “mrg-manager” key
  2. Click “Update Activation Keys”

### Automate the Kickstart by Adding post scripts

Target System: Browser with access to RHN

1. Click “Systems” on the upper navigation menu
2. Click “Kickstart” on left navigation menu
3. Click “View a list of Kickstart Profiles” in the “Kickstart Actions” pane
4. Select the “mrg-manager” kickstart profile
5. Click on “Scripts” on the upper navigation menu
6. Click “Add new Kickstart Script” in the upper navigation menu
  1. Add the following content

```
wget http://sat-vm.rhcf.lab/pub/scripts/mrgMgr-config.sh -O
/tmp/mrg-install.sh
chmod +x /tmp/mrg-install.sh
/tmp/mrg-install.sh
```

2. Change “Script Execution Time” to “Post Script”
3. Click “Update Kickstart”

## Deploy MRG Server using RHEV

Target System: rhevm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name”
  2. Provide a “Description”
  3. Change “Memory Size” to “2048 MB”
  4. Change “Total Cores” to “2”
  5. Change the “Operating System” to “Red Hat Enterprise Linux 5.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
5. Click on the “Boot Sequence” tab on the left navigation pane
  1. Change “Second Device” to “Network (PXE)”
  2. Click “OK”
6. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
7. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
8. Click the “Run” button in the top navigation menu and perform the install
9. Click the “Console” button in the top navigation menu and monitor the PXE installation
10. After the virtual machine is installed, configure cumin

```
useradd mrgmgr
passwd mrgmgr
cumin-admin add-user mrgmgr
1. Provide a password
```

11. Create a “admin” user

```
useradd admin
password admin
```

Log into the MRG interface



1. Open a browser on the MRG localhost and go to <http://127.0.0.1:45672/login.html> or from a remote machine and log in.
2. Confirm functionality by logging in as mrgmgr with the password set above

### Check functionality via CLI

1. Log in as the “admin” user and issue the following command to look at the queue

```
$ condor_status -any
MyType TargetType Name
DaemonMaster None mrgexec.rhcf.lab
```

### Create a MRG Execution Node Activation Key

Target System: Browser with access to RHN

1. Click the “Systems” tab on the upper navigation menu
2. Click “Activation Keys” on left navigation menu
3. Click “Create New Key” on upper navigation menu
4. Provide the following information
  1. Description: “mrg-exec-node”
  2. Key: “mrg-exec-node”
  3. Add-On Entitlements: “Monitoring” and “Provisioning”
  4. Click “Create Activation Key”
5. On the RHEV-H KVM Hypervisor Activation Key screen, click the “Child Channels” tab in the upper navigation menu
  1. Keep the defaults and select the following channels
    - “MRG Grid Execute Node v. 1 (for RHEL 5 Server 64-bit x86\_64)”
    - “MRG Messaging Base v. 1 (for RHEL 5 Server 64-bit x86\_64)”
    - “MRG Messaging v. 1 (for RHEL 5 Server 64-bit x86\_64)”
    - “Red Hat Network Tools for RHEL Server (v.5 64-bit x86\_64)”
  2. Click “Update Key”
6. On the mrg-exec-node Activation Key screen, click the “Packages” tab in the upper navigation menu and add the following packages
  - condor
  - sesame
  - qpidd
7. Click “Update Key”

### Create Kickstart Profile for mrg-exec-node

Target System: Browser with access to RHN

1. Click “Systems” in the top navigation pane
2. On the left navigation pane click “Kickstart”
3. In the “Kickstart Actions” pane click “Create a New Kickstart Profile”
4. On “Step 1”
  1. Provide a “Label” for the kickstart “mrg-exec-node”
  2. On “Base Channel” select “Red Hat Enterprise Linux (v. 5 for 64-bit x86\_64)”
  3. On “Kickstartable Tree” select “ks-rhel-x86\_64-server-5-u6”
  4. On “Virtualization Type” select “KVM Virtualized Guest”
  5. Click “Next”
5. On “Step 2”
  1. Click “Next”
6. On “Step 3”
  1. Provide a root password
7. Click “Finish”
8. Click “Activation Keys” in the upper navigation menu
  1. In the “Kickstart Details” screen, select the “mrg-exec-node” key
  2. Click “Update Activation Keys”

### Automate the Kickstart by Adding post scripts

Target System: Browser with access to RHN

1. Click “Systems” on the upper navigation menu
2. Click “Kickstart” on left navigation menu
3. Click “View a list of Kickstart Profiles” in the “Kickstart Actions” pane
4. Select the “mrg-manager” kickstart profile
5. Click on “Scripts” on the upper navigation menu
6. Click “Add new Kickstart Script” in the upper navigation menu
  1. Add the following content

```
wget http://sat-vm.rhcf.lab/pub/scripts/mrgExec-config.sh -O
/tmp/mrg-exec-node-config.sh
chmod +x /tmp/mrg-exec-node-config.sh
/tmp/mrg-exec-node-config.sh
```

2. Change “Script Execution Time” to “Post Script”
3. Click “Update Kickstart”



## Deploy MRG Execution Nodes using RHEV

Target System: rhevm-vm

1. Click on the “Virtual Machines” tab on upper navigation menu
2. Click on the “New Server” button on upper navigation menu
3. On the “New Server Virtual Machine” screen, on the “General” tab on left navigation menu
  1. Provide a “Name”
  2. Provide a “Description”
  3. Change “Memory Size” to “2048 MB”
  4. Change “Total Cores” to “2”
  5. Change the “Operating System” to “Red Hat Enterprise Linux 5.x x64”
4. Click on the “High Availability” tab on the left navigation pane
  1. Check the “High Availability” box
5. Click on the “Boot Sequence” tab on the left navigation pane
  1. Change “Second Device” to “Network (PXE)”
  2. Click “OK”
6. Click “Configure Network Interfaces” button
  1. Take the defaults and click “OK”
7. Click “Configure Virtual Disks”
  1. Provide “Size”
  2. Check “Wipe after delete”
  3. Click “OK”
  4. Click “Configure Later”
8. Click the “Run” button in the top navigation menu and perform the install
9. Click the “Console” button in the top navigation menu and monitor the PXE installation
10. Confirm functionality by issuing the following command

```
$ condor_status -any
```

| MyType         | TargetType | Name                                  |
|----------------|------------|---------------------------------------|
| DaemonMaster   | None       | rhcf-mrgexec.cloud.lab.eng.bos        |
| <b>Machine</b> | <b>Job</b> | <b>slot1@rhcf-mrgexec.cloud.lab.e</b> |
| <b>Machine</b> | <b>Job</b> | <b>slot2@rhcf-mrgexec.cloud.lab.e</b> |
| Collector      | None       | RHCF_Grid@rhcf-mrgmgr.cloud.la        |
| Scheduler      | None       | rhcf-mrgmgr.cloud.lab.eng.bos.        |
| DaemonMaster   | None       | rhcf-mrgmgr.cloud.lab.eng.bos.        |
| Negotiator     | None       | rhcf-mrgmgr.cloud.lab.eng.bos.        |

11. Now deploy 3 more MRG Execution nodes using the same method to extend the grid

1. Once these nodes are deployed, name them mrgexec{1-3} and make sure the IP addresses are static and the hosts are set up in DNS and that name resolution is working properly.
12. After the nodes are installed, re-check that they are available with `condor_status -any` from the mrgmgr node

```
condor_status -any
```

| MyType       | TargetType | Name                           |
|--------------|------------|--------------------------------|
| Machine      | Job        | slot2@rhcf-mrgexec.cloud.lab.e |
| DaemonMaster | None       | rhcf-mrgexec1.cloud.lab.eng.bo |
| Machine      | Job        | slot1@rhcf-mrgexec1.cloud.lab. |
| Machine      | Job        | slot2@rhcf-mrgexec1.cloud.lab. |
| Machine      | Job        | slot3@rhcf-mrgexec1.cloud.lab. |
| Machine      | Job        | slot4@rhcf-mrgexec1.cloud.lab. |
| DaemonMaster | None       | rhcf-mrgexec2.cloud.lab.eng.bo |
| Machine      | Job        | slot1@rhcf-mrgexec2.cloud.lab. |
| Machine      | Job        | slot2@rhcf-mrgexec2.cloud.lab. |
| Machine      | Job        | slot3@rhcf-mrgexec2.cloud.lab. |
| Machine      | Job        | slot4@rhcf-mrgexec2.cloud.lab. |
| DaemonMaster | None       | rhcf-mrgexec3.cloud.lab.eng.bo |
| Machine      | Job        | slot1@rhcf-mrgexec3.cloud.lab. |
| Machine      | Job        | slot2@rhcf-mrgexec3.cloud.lab. |
| Machine      | Job        | slot3@rhcf-mrgexec3.cloud.lab. |
| Machine      | Job        | slot4@rhcf-mrgexec3.cloud.lab. |
| Collector    | None       | RHCF Grid@rhcf-mrgmgr.cloud.la |
| Scheduler    | None       | rhcf-mrgmgr.cloud.lab.eng.bos. |
| DaemonMaster | None       | rhcf-mrgmgr.cloud.lab.eng.bos. |
| Negotiator   | None       | rhcf-mrgmgr.cloud.lab.eng.bos. |



## Launch jobs on the MRG Execution nodes

Target System: mrgmgr

1. Create the “admin” user and log in with the “admin” user account, generate a ssh public / private key pair.
2. Create a “admin” user on each of the MRG execute virtual machines
3. Share the public key created in step 1 with the other MRG execute nodes using ssh-copy-id
4. Extract the perfect.tgz file in “admin” home directory

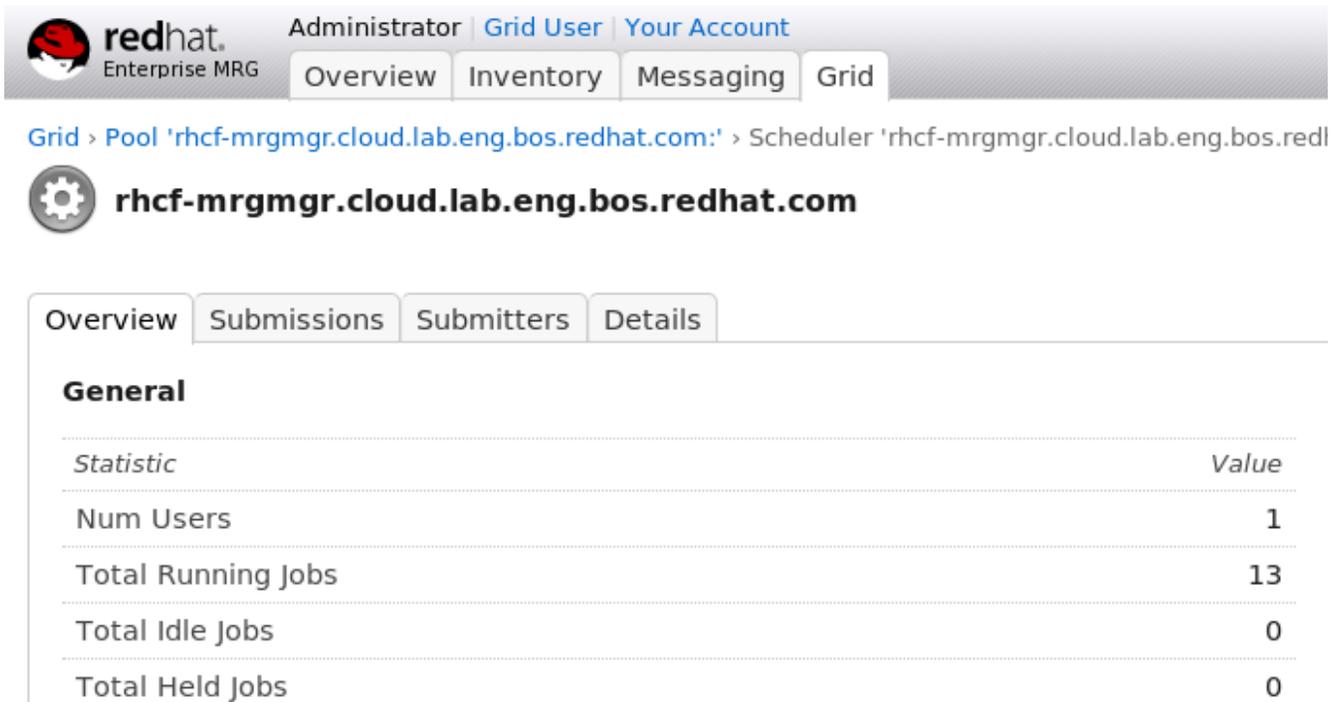
```
$ tar xzvf perfect.tgz
perfect/
perfect/perfect.py
perfect/mk_jobs.sh
perfect/output/
perfect/submit_jobs.sh
perfect/ec2_tunnel.sub
perfect/save/
perfect/save/mk_jobs.sh
perfect/ec2_mrgexec.sub
perfect/perfect
perfect/perfect_number.sub
```

5. Copy the “perfect” executable to the */home/admin/* directory on each MRG execution node and name it “perfect1”

6. Submit the perfect number job to multiple hosts

```
$ condor_submit perfect_number.sub
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 101.
```

7. Verify the job is running in the GUI as shown in **Figure 9.2: MRG Grid with Job Executing**



The screenshot shows the Red Hat Enterprise MRG GUI. At the top, there is a navigation bar with the Red Hat logo and 'Enterprise MRG' text. To the right of the logo, there are links for 'Administrator', 'Grid User', and 'Your Account'. Below these links are tabs for 'Overview', 'Inventory', 'Messaging', and 'Grid'. The main content area shows a breadcrumb trail: 'Grid > Pool 'rhcf-mrgmgr.cloud.lab.eng.bos.redhat.com:' > Scheduler 'rhcf-mrgmgr.cloud.lab.eng.bos.redhat.com:'. Below the breadcrumb is a gear icon and the text 'rhcf-mrgmgr.cloud.lab.eng.bos.redhat.com'. Underneath, there are tabs for 'Overview', 'Submissions', 'Submitters', and 'Details'. The 'Overview' tab is selected, and it displays a 'General' section with a table of statistics.

| Statistic          | Value |
|--------------------|-------|
| Num Users          | 1     |
| Total Running Jobs | 13    |
| Total Idle Jobs    | 0     |
| Total Held Jobs    | 0     |

**Figure 9.2: MRG Grid with Job Executing**



## 8. Check the job queue via CLI

```
$ condor_q
-- Submitter: mrgmgr.rhcf.lab : <10.16.139.32:9675> : mrgmgr.rhcf.lab
ID OWNER SUBMITTED RUN_TIME ST PRI SIZE CMD
91.0 admin 4/23 21:29 0+00:00:00 I 0 0.0 perfect1 1 1000000
92.0 admin 4/23 21:29 0+00:00:00 I 0 0.0 perfect1 1 1000000
93.0 admin 4/23 21:29 0+00:00:00 I 0 0.0 perfect1 1 1000000
94.0 admin 4/23 21:29 0+00:00:00 I 0 0.0 perfect1 1 1000000
95.0 admin 4/23 21:29 0+00:00:00 I 0 0.0 perfect1 1 1000000
```

## 9. While it is outside the scope of this paper to be a troubleshooting guide, if the job is submitted and it does not run, check what the problem is with `condor_q -better-analyze`

```
$ condor_q -better-analyze
-- Submitter: mrgmgr.rhcf.lab : <10.16.139.32:9675> : mrgmgr.rhcf.lab

096.000: Run analysis summary. Of 14 machines,
 0 are rejected by your job's requirements
 0 reject your job because of their own requirements
 0 match but are serving users with a better priority in the pool
 14 match but reject the job for unknown reasons
 0 match but will not currently preempt their existing job
 0 match but are currently offline
 0 are available to run your job
```

## 10. Verify the results of the perfect number job

```
$ cat perfect/output/perfect.out
6
28
496
8128
```

# 10 Summary

As customers move applications towards the cloud, it is going to become more important than ever to have an established, tested software stack that is easy to deploy, manage and scale. As described in this paper, Red Hat Cloud Foundations meets these needs . This paper describes the process to deploy a Red Hat Cloud Foundation solution.

The paper describes what software is needed, where to get the software, how to deploy the servers and applications, and finally, how to configure the applications. This Red Hat Cloud Foundations paper provides all the information required to set up an environment using best practices.



# 11 Appendix A

## 11.1 Configuration Files

### DNS Zone Files:

#### rhcf.local.db -

```
$TTL 1H
@ SOA dndsdhcp-vm root.rhcf-sat-vm.rhcf.local. (3
 3H
 1H
 1W
 1H)
NS dndsdhcp-vm
dndsdhcp-vm IN 1H A 192.168.0.100
mgmt1 IN 1H A 192.168.0.10
mgmt2 IN 1H A 192.168.0.11
rhev1 IN 1H A 192.168.0.12
rhev2 IN 1H A 192.168.0.13
sat-vm IN 1H A 192.168.0.14
luci-vm IN 1H A 192.168.0.15
rhev-vm IN 1H A 192.168.0.16
nfs-vm IN 1H A 192.168.0.17
cobbler-vm IN 1H A 192.168.0.18
sat IN 1H CNAME sat-vm
```

#### 192.168.0.db -

```
$TTL 1H
@ SOA dndsdhcp-vm.rhcf.local. root.dndsdhcp.rhcf.local. (3
 3H
 1H
 1W
 1H)
NS dndsdhcp-vm.rhcf.local.
100 PTR dndsdhcp-vm.rhcf.local.
10 PTR mgmt1.rhcf.local.
11 PTR mgmt2.rhcf.local.
12 PTR rhev1.rhcf.local.
13 PTR rhev2.rhcf.local.
14 PTR sat-vm.rhcf.local.
15 PTR luci-vm.rhcf.local.
16 PTR rhev-vm.rhcf.local.
17 PTR nfs-vm.rhcf.local.
18 PTR cobbler-vm.rhcf.local.
```

## dhcpcd.conf

```
dhcpcd.conf

option definitions common to all supported networks...
option domain-name "rhcf.lab";
option domain-name-servers dnsdhcp.rhcf.lab;

default-lease-time 600;
max-lease-time 7200;

Use this to enable / disable dynamic dns updates globally.
#ddns-update-style none;

If this DHCP server is the official DHCP server for the local
network, the authoritative directive should be uncommented.
#authoritative;

Use this to send dhcp log messages to a different log file (you also
have to hack syslog.conf to complete the redirection).
log-facility local7;

subnet 192.168.0.0 netmask 255.255.255.0 {
 range 192.168.0.10 192.168.0.30;
 option domain-name-servers 192.168.0.2; # DNS DHCP Virtual Machine
 option domain-name "rhcf.lab";
 option routers 192.168.0.1;
 option broadcast-address 192.168.0.255;
 default-lease-time 600;
 max-lease-time 7200;
 next-server 192.168.0.3; # Satellite Server
 filename = "pxelinux.0";
}

host X {
 hardware ethernet 0:0:c0:5d:bd:95;
 fixed-address 192.168.50.1
 server-name "rhev-vm";
}

Continue adding hosts
```

## Multipath.conf File (Changes are in bold):

```
This is a basic configuration file with some examples, for device mapper
multipath.
```



```
For a complete list of the default configuration values, see
/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf.defaults
For a list of configuration options with descriptions, see
/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf.annotated

By default, devices with vendor = "IBM" and product = "S/390.*" are
blacklisted. To enable mulitpathing on these devies, uncomment the
following lines.
#blacklist_exceptions {
device {
vendor "IBM"
product "S/390.*"
}
#}

Use user friendly names, instead of using WWIDs as names.
defaults {
 user_friendly_names no
}
##
Here is an example of how to configure some standard options.
##
#
#defaults {
udev_dir /dev
polling_interval 10
selector "round-robin 0"
path_grouping_policy multibus
getuid_callout "/lib/udev/scsi_id --whitelisted --device=/dev/%n"
prio alua
path_checker readsector0
rr_min_io 100
max_fds 8192
rr_weight priorities
failback immediate
no_path_retry fail
user_friendly_names yes
#}
##
The wwid line in the following blacklist section is shown as an example
of how to blacklist devices by wwid. The 2 devnode lines are the
compiled in default blacklist. If you want to blacklist entire types
of devices, such as all scsi devices, you should use a devnode line.
However, if you want to blacklist specific devices, you should use
a wwid line. Since there is no guarantee that a specific device will
not change names on reboot (from /dev/sda to /dev/sdb for example)
devnode lines are not recommended for blacklisting specific devices.
##
```

```

#blacklist {
wwid 26353900f02796769
devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
devnode "^hd[a-z]"
#}
multipaths {
multipath {
wwid 3600508b4000156d700012000000b0000
alias yellow
path_grouping_policy multibus
path_checker readsector0
path_selector "round-robin 0"
failback manual
rr_weight priorities
no_path_retry 5
}
multipath {
wwid 1DEC_____321816758474
alias red
}
multipath {
wwid 3600c0ff000d84a99659e414d01000000
alias rhcflun
}
#}
#devices {
device {
vendor "COMPAQ "
product "HSV110 (C)COMPAQ"
path_grouping_policy multibus
getuid_callout "/lib/udev/scsi_id --whitelisted --device=/dev/%n"
path_checker readsector0
path_selector "round-robin 0"
hardware_handler "0"
failback 15
rr_weight priorities
no_path_retry queue
}
device {
vendor "COMPAQ "
product "MSA1000 "
path_grouping_policy multibus
}
#}

```

**Cobbler** settings file (Changes are in bold and comments are removed):

```

allow_duplicate_hostnames: 0 allow_duplicate_ips: 0

```



```
allow_duplicate_macs: 0
anamon_enabled: 0
build_reporting_enabled: 0
build_reporting_sender: ""
build_reporting_email: ['root@localhost']
build_reporting_smtp_server: "localhost"
build_reporting_subject: ""
cheetah_import_whitelist:
 - "random"
 - "re"
 - "time"
createrepo_flags: "-c cache -s sha"
default_kickstart: /var/lib/cobbler/kickstarts/default.ks
default_name_servers: []
default_ownership:
 - "admin"
default_password_crypted: "1mF86/UHC$WvcIcX2t6crBz2onWxyac."
default_virt_bridge: xenbr0
default_virt_file_size: 5
default_virt_ram: 512
default_virt_type: xenpv
enable_menu: 1
func_auto_setup: 0
func_master: overlord.example.org
http_port: 80
kernel_options:
 ksdevice: bootif
 lang: ''
 text: ~
kernel_options_s390x:
 RUNKS: 1
 ramdisk_size: 40000
 root: /dev/ram0
 ro: ~
 ip: off
 vnc: ~
ldap_server: "ldap.example.com"
ldap_base_dn: "DC=example,DC=com"
ldap_port: 389
ldap_tls: 1
ldap_anonymous_bind: 1
ldap_search_bind_dn: ""
ldap_search_passwd: ""
ldap_search_prefix: 'uid='
mgmt_classes: []
mgmt_parameters:
 from_cobbler: 1
manage_dhcp: 1
```

```
manage_dns: 1
manage_forward_zones:
 - 'rhcf.lab'
manage_reverse_zones:
 - '192.168.0'
next_server: sat-vm.cloud.lab.eng.bos.redhat.com
power_management_default_type: 'ipmitool'
power_template_dir: "/etc/cobbler/power"
pxe_just_once: 1
pxe_template_dir: "/etc/cobbler/pxe"
redhat_management_type: "site"
redhat_management_server: "sat-vm.cloud.rhcf.lab"
redhat_management_key: ""
redhat_management_permissive: 0
register_new_installs: 0
reposync_flags: "-l -m -d"
restart_dns: 1
restart_dhcp: 1
run_install_triggers: 1
scm_track_enabled: 0
scm_track_mode: "git"
server: dnshcp-vm.rhcf.lab
snippetsdir: /var/lib/cobbler/snippets
template_remote_kickstarts: 0
virt_auto_boot: 1
webdir: /var/www/cobbler
xmlrpc_port: 25151
yum_post_install_mirror: 1
yum_distro_priority: 1
yumdownloader_flags: "--resolve"
safe_templating: true
```



# 12 Appendix B Scripts

## jboss-eap-install.sh

```
#!/bin/bash

JBOSS_DIR=/var/lib
#Config JBoss
echo configuring jboss
sed -i 's/#admin=admin/admin=100yard-/' /var/lib/jbossas/server/default/conf/props/jmx-console-
users.properties

deploy the test app
cd $JBOSS_DIR/jbossas/server/default/deploy
wget http://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com/pub/kits/jboss-seam-booking-ds.xml
wget http://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com/pub/kits/jboss-seam-booking.ear
chown jboss.jboss jboss-seam-*

configure JBoss auto start
chkconfig jbossas on
service jbossas start

Copy startup script
cd /etc/init.d
wget http://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com/pub/scripts/jboss-eap
chmod +x jboss-eap
/sbin/chkconfig --add jboss-eap
/sbin/chkconfig jboss-eap on
```

## jon-agent-install.sh

```
#!/bin/bash

download, install and configure the JON agent
cd /root
wget http://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com/pub/kits/rhq-enterprise-agent-default.GA.jar
java -jar /root/rhq-enterprise-agent-default.GA.jar --install
cd /root/rhq-agent/conf
line=$(grep -n "key=\"rhq.agent.configuration-setup-flag" agent-configuration.xml | cut -d: -f1)
before=`expr $line - 1`
after=`expr $line + 1`
sed -e "${after}d" -e "${before}d" agent-configuration.xml > agent-configuration.xml2
mv agent-configuration.xml2 agent-configuration.xml
sed -e '/rhq.agent.configuration-setup-flag/s/false/true/g' agent-configuration.xml > agent-configuration.xml2
mv agent-configuration.xml2 agent-configuration.xml
sed -e '/rhq.agent.server.bind-address/s/value=\.*/value="ra-jon-vm.cloud.lab.eng.bos.redhat.com"/g' agent-configuration.xml > agent-configuration.xml2
mv agent-configuration.xml2 agent-configuration.xml
cd /root/rhq-agent/bin
mv rhq-agent-env.sh rhq-agent-env.sh.orig
wget http://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com/pub/kits/rhq-agent-env.sh

configure JON agent to auto start
cd /etc/init.d
sed -e "s/readlink/readlink -e/g" /root/rhq-agent/bin/rhq-agent-wrapper.sh > /root/rhq-agent/bin/rhq-agent-wrapper.sh2
mv /root/rhq-agent/bin/rhq-agent-wrapper.sh2 /root/rhq-agent/bin/rhq-agent-wrapper.sh
sed -i 's/readlink/readlink -e/g' /root/rhq-agent/bin/rhq-agent-wrapper.sh
ln -s /root/rhq-agent/bin/rhq-agent-wrapper.sh .
chmod +x jboss-eap rhq-agent-wrapper.sh
/sbin/chkconfig --add rhq-agent-wrapper.sh
/sbin/chkconfig rhq-agent-wrapper.sh on
```



## rhq-agent-env.sh

```

====
RHQ Agent UNIX Startup Script Configuration File

====

RHQ_AGENT_DEBUG - If this is defined, the script will emit debug
messages. It will also enable debug
messages to be emitted from the agent itself.
If not set or set to "false", debug is turned off.
This does not implicitly enable Sigar native system
debug mode. You must explicitly enable
RHQ_AGENT_SIGAR_DEBUG in addition to enabling
RHQ_AGENT_DEBUG for Sigar logging to be enabled.

#RHQ_AGENT_DEBUG=true

RHQ_AGENT_SIGAR_DEBUG - Enables Sigar debug mode but only if agent debug
is also enabled. See RHQ_AGENT_DEBUG for more.
#RHQ_AGENT_SIGAR_DEBUG=false

RHQ_AGENT_HOME - Defines where the agent's home install directory is.
If not defined, it will be assumed to be the parent
directory of the directory where this script lives.

RHQ_AGENT_HOME="/root/rhq-agent"

RHQ_AGENT_JAVA_HOME - The location of the JRE that the agent will
use. This will be ignored if
RHQ_AGENT_JAVA_EXE_FILE_PATH is set.
If this and RHQ_AGENT_JAVA_EXE_FILE_PATH are
not set, the agent's embedded JRE will be used.

RHQ_AGENT_JAVA_HOME="/usr/lib/jvm/jre-1.6.0"

RHQ_AGENT_JAVA_EXE_FILE_PATH - Defines the full path to the Java
executable to use. If this is set,
RHQ_AGENT_JAVA_HOME is ignored.
If this is not set, then
$RHQ_AGENT_JAVA_HOME/bin/java
is used. If this and
RHQ_AGENT_JAVA_HOME are not set, the
agent's embedded JRE will be used.

#RHQ_AGENT_JAVA_EXE_FILE_PATH="/usr/local/bin/java"
```

```

RHQ_AGENT_JAVA_OPTS - Java VM command line options to be
passed into the agent's VM. If this is not defined
this script will pass in a default set of options.
If this is set, it completely overrides the
agent's defaults. If you only want to add options
to the agent's defaults, then you will want to
use RHQ_AGENT_ADDITIONAL_JAVA_OPTS instead.
#
#RHQ_AGENT_JAVA_OPTS="-Xms64m -Xmx128m -Djava.net.preferIPv4Stack=true"

RHQ_AGENT_JAVA_ENDORSED_DIRS - Java VM command line option to set the
endorsed dirs for the agent's VM. If this
is not defined this script will pass in a
default value. If this is set, it
completely overrides the agent's default.
However, if this is set to "none", the
agent will not be passed the VM argument
to set the endorsed dirs.
#
#RHQ_AGENT_JAVA_ENDORSED_DIRS="${RHQ_AGENT_HOME}/lib/endorsed"

RHQ_AGENT_JAVA_LIBRARY_PATH - The RHQ Agent has a JNI library that
it needs to find in order to do things
like execute PIQL queries and access
low-level operating system data. This
is the native system layer (SIGAR).
If you deploy a custom plugin that also
requires JNI libraries, you must add to
the library path here, but you must ensure
not to remove the RHQ Agent library path.
If this variable is set, it completely
overrides the agent's default.
However, if this is set to "none", the
agent will not be passed the VM argument
to set the library paths.
#
#RHQ_AGENT_JAVA_LIBRARY_PATH="${RHQ_AGENT_HOME}/lib"

RHQ_AGENT_ADDITIONAL_JAVA_OPTS - additional Java VM command line options
to be passed into the agent's VM. This
is added to RHQ_AGENT_JAVA_OPTS; it
is mainly used to augment the agent's
default set of options. This can be
left unset if it is not needed.
#
#RHQ_AGENT_ADDITIONAL_JAVA_OPTS="-
agentlib:jdwp=transport=dt_socket,address=9797,server=y,suspend=n"

```



```
RHQ_AGENT_CMDLINE_OPTS - If this is defined, these are the command line
arguments that will be passed to the RHQ Agent.
Any arguments specified on the command line
will be ignored. If this is not defined, the
command line arguments given to the script are
passed through to the RHQ Agent.
If you want to have command line arguments
added to the arguments specified here, append
'$*' to the end of this option. For example,
"--daemon $*". In this case, both the command
line options and the ones specified here will
be passed to the agent.
#
#RHQ_AGENT_CMDLINE_OPTS="--daemon --nonative --cleanconfig"
RHQ_AGENT_CMDLINE_OPTS="--daemon --cleanconfig"

RHQ_AGENT_IN_BACKGROUND - If this is defined, the RHQ Agent JVM will
be launched in the background (thus causing this
script to exit immediately). If the value is
something other than "nofile", it will be assumed
to be a full file path which this script will
create and will contain the agent VM's process
pid value. If this is not defined, the VM is
launched in foreground and this script blocks
until the VM exits, at which time this
script will also exit. NOTE: this should only
be used by the rhq-agent-wrapper.sh script.
If you want to launch the agent in the
background, use that script to do so instead of
setting this variable.
#
#RHQ_AGENT_IN_BACKGROUND=rhq-agent.pid

#=====
====
THE FOLLOWING ARE USED SOLELY FOR THE rhq-agent-wrapper.sh SCRIPT

RHQ_AGENT_PIDFILE_DIR - When rhq-agent-wrapper.sh is used to start
the agent, it runs the process in background
and writes its pid to a pidfile. The default
location of this pidfile is in the agent's
/bin directory. If you want to have the pidfile
written to another location, set this environment
variable. This value must be a full path to a
directory with write permissions.
#
RHQ_AGENT_PIDFILE_DIR="/var/run"
```

```

RHQ_AGENT_START_COMMAND - If defined, this is the command that will be
#
executed to start the agent.
#
Use this to customize how the agent process
#
is started (e.g. with "su" or "sudo").
#
This completely overrides the command used
#
to start the agent - you must ensure you
#
provide a valid command that starts the agent
#
script 'rhq-agent.sh'
#
Note that if this start command requires the
#
user to enter a password, you can show a
#
prompt to the user if you set the variable
#
RHQ_AGENT_PASSWORD_PROMPT.
#
Also note that if your agent install directory
#
has spaces in its name, you might have to do
#
some special string manipulation to get the
#
agent script to start. See below for an
#
example of how to do this.
#RHQ_AGENT_START_COMMAND="su -m -l user -c '${RHQ_AGENT_HOME}/bin/rhq-
agent.sh"
#RHQ_AGENT_START_COMMAND="su -m -l user -c '$(echo ${RHQ_AGENT_HOME}|sed 's/\\/
/)/bin/rhq-agent.sh"

RHQ_AGENT_PASSWORD_PROMPT - if "true", this indicates that the user
#
that is to run the agent must type the
#
password on the console in order to run.
#
Therefore, "true" forces a prompt message
#
to appear on the console. If this is
#
defined, but not equal to "true", the value
#
itself will be used as the prompt string.
#
This is not defined by default. Note that
#
this setting does nothing more than simply
#
printing a message to the console.
#
#RHQ_AGENT_PASSWORD_PROMPT=true

```

## rhq-install.sh

```

#!/bin/bash
quick 'n dirty JON/JOPR/RHQ installation/reinstallation script
for zipped distributions
#
Author: Steve Salevan <ssalevan@redhat.com>
Modified: BJ Lachance <blachanc@redhat.com>

source env vars

```



```
if [[-x varDefs.sh]]; then
source varDefs.sh
elif [[-x /root/varDefs.sh]]; then
source /root/varDefs.sh
elif [[-x /root/resources/varDefs.sh]]; then
source /root/resources/varDefs.sh
elif [[-x /root/distro/resources/varDefs.sh]]; then
source /root/distro/resources/varDefs.sh
else
echo "didn't find a varDefs.sh file"
fi

#script default values:
HOSTNAME=`hostname`
IP=`ifconfig eth0 | grep 'inet addr' | sed 's/.*inet addr://' | sed 's/ .*//`
CURR_USER=`whoami`
AUTOINSTALL_WAITTIME=300
UNINSTALL_ONLY=0
RECREATE_USER=0
JON installation defaults (what user gets created, where JON lands)
JON_ROOT=rhq/
JON_USER=rhq

Java defaults

JAVA_HOME=/usr/lib/jvm/jre-openjdk

JON-specific defaults
DB_CONNECTION_URL="jdbc:postgresql://127.0.0.1:5432\rhq"
DB_SERVER_NAME="127.0.0.1"
HA_NAME=$HOSTNAME
SAT_SERVER=http://rhcf-sat-vm.cloud.lab.eng.bos.redhat.com
JON_URL="$SAT_SERVER/pub/kits/jon-server-2.4.1.GA.zip"
SOA_PI_URL="$SAT_SERVER/pub/kits/jon-plugin-pack-soa-2.4.1.GA.zip"
EWS_PI_URL="$SAT_SERVER/pub/kits/jon-plugin-pack-ews-2.4.1.GA.zip"
EAP_PI_URL="$SAT_SERVER/pub/kits/jon-plugin-pack-eap-2.4.1.GA.zip"
JON_LICENSE_URL="$SAT_SERVER/pub/kits/jon-license.xml"

if [$CURR_USER != "root"]; then
 echo "You must be logged in as the root user to install JON."
 exit 1
fi

function jon_uninstall {
 # find service script
 echo " * Finding JON/JOPR/RHQ service script location..."
```

```

SVC_SCRIPT=`find /etc/init.d/ -iname "*rhq*"`
if [-z "$SVC_SCRIPT"]; then
 SVC_SCRIPT=`find /etc/init.d/ -iname "*jon*"`
fi
if [-z "$SVC_SCRIPT"]; then
 echo " - No previous installations found."
 return
fi
echo " - Found JON/JOPR/RHQ service script at: $SVC_SCRIPT"

find home directory
echo " * Finding first-defined JON/JOPR/RHQ home directory..."
for i in $SVC_SCRIPT; do
 for dir in `grep RHQ_SERVER_HOME= $i | sed 's/[-a-zA-Z0-9_]*=//`;
 do
 if [-a $dir]; then
 JON_HOME=$dir;
 fi
 done
 if [-z "$JON_HOME"]; then
 echo " - JON/JOPR/RHQ home directory was not defined in the service script,
uninstall failed."
 exit 1
 else
 break
 fi
done

if [-z "$JON_HOME"]; then
 echo " - JON/JOPR/RHQ home directory was not defined in the service script, uninstall
failed."
 exit 1
fi
echo " - Found JON/JOPR/RHQ home directory at: $JON_HOME"

echo " * Stopping all services, removing service script..."
$SVC_SCRIPT stop
rm -f $SVC_SCRIPT

echo " * Dropping Postgres tables..."
su - postgres -c "psql -c \"DROP DATABASE rhq;\""
su - postgres -c "psql -c \"DROP USER rhqadmin;\""

echo " * Deleting JON/JOPR/RHQ..."
rm -rf $JON_HOME
echo " - Uninstall complete!"
}

```



```
handle CLI overrides
for i in $*
do
case $i in
--jon-localuser=*)
 JON_USER=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--jon-rootdir=*)
 JON_ROOT=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--jon-url=*)
 JON_URL=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--licenseurl=*)
 JON_LICENSE_URL=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--db-connectionurl=*)
 DB_CONNECTION_URL=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--db-servername=*)
 DB_SERVER_NAME=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--ha-name=*)
 HA_NAME=""`echo $i | sed 's/[-a-zA-Z0-9]*=/'`"
 ;;
--uninstall*)
 UNINSTALL_ONLY=1
 ;;
--recreateuser*)
 RECREATE_USER=1
 ;;
*)
unknown option
echo "You entered an option I didn't recognize."
echo ""
echo "If an option is not specified, a default will be used."
echo "Available options:"
echo "--jon-url URL pointing to JON distribution zipfile"
echo "--jon-localuser Username for local user which JON will be installed"
echo "--jon-rootdir Directory beneath local user's home into which JON will be installed"
echo "--db-connectionurl DB connection URL (e.g. jdbc:postgresql://127.0.0.1:5432/rhq)"
echo "--db-servername DB server name (e.g. 127.0.0.1)"
echo "--ha-name Name for this server, if using High Availability"
echo "--licenseurl URL pointing to JON license XML file"
echo "--uninstall Only uninstall the current JON/JOPR/RHQ instance"
echo "--recreateuser Create or recreate the local user account as part of installation"
exit 1
;;
```

```

esac
done

cover uninstall only case
if [$UNINSTALL_ONLY -eq 1]; then
 jon_uninstall
 exit 0
fi

if specified JON user is not present, we must create it
/bin/egrep -i "^$JON_USER" /etc/passwd > /dev/null
if [$? != 0]; then
 echo " - Specified JON local user does not exist; hence, it will be created."
 RECREATE_USER=1
fi

get jon and pop it into a new jon user directory
echo " * Purging any old installs and downloading JON..."
jon_uninstall

if [$RECREATE_USER -eq 1]; then
 userdel -f $JON_USER
 rm -rf /home/$JON_USER
 useradd $JON_USER -p $1$1o751Xnc$kmQKHj6gtZ50IILNkHkkF0
mkdir /home/$JON_USER/$JON_ROOT
chown ${JON_USER}.${JON_USER} /home/$JON_USER/$JON_ROOT
fi

echo wget $SOA_PI_URL -O ./jon_soa_plugin.zip
wget $SOA_PI_URL -O ./jon_soa_plugin.zip
echo wget $EAP_PI_URL -O ./jon_eap_plugin.zip
wget $EAP_PI_URL -O ./jon_eap_plugin.zip
echo wget $EWS_PI_URL -O ./jon_ews_plugin.zip
wget $EWS_PI_URL -O ./jon_ews_plugin.zip
echo wget $JON_URL -O ./jon.zip
wget $JON_URL -O ./jon.zip
chown $JON_USER ./jon_soa_plugin.zip
chown $JON_USER ./jon_eap_plugin.zip
chown $JON_USER ./jon_ews_plugin.zip
chown $JON_USER ./jon.zip
mv ./jon.zip /home/$JON_USER
mv ./jon_soa_plugin.zip /home/$JON_USER
mv ./jon_eap_plugin.zip /home/$JON_USER
mv ./jon_ews_plugin.zip /home/$JON_USER

start postgres
echo " * Configuring Postgres..."
service postgresql initdb

```



```
service postgresql start
```

```
rig postgres
```

```
su - postgres -c "psql -c \"CREATE USER rhqadmin WITH PASSWORD 'rhqadmin';\""
```

```
su - postgres -c "psql -c \"CREATE DATABASE rhq;\""
```

```
su - postgres -c "psql -c \"GRANT ALL PRIVILEGES ON DATABASE rhq to rhqadmin;\""
```

```
echo "
```

```
TYPE DATABASE USER CIDR-ADDRESS METHOD
```

```
\"local\" is for Unix domain socket connections only
```

```
local all all trust
```

```
IPv4 local connections:
```

```
host all all 127.0.0.1/32 trust
```

```
host all all 10.0.0.1/8 md5
```

```
IPv6 local connections:
```

```
host all all ::1/128 trust
```

```
" > /var/lib/pgsql/data/pg_hba.conf
```

```
chkconfig postgresql on
```

```
service postgresql restart
```

```
echo " * Unzipping and configuring JON..."
```

```
unzip jon
```

```
su - $JON_USER -c 'unzip jon.zip'
```

```
su - $JON_USER -c 'rm jon.zip'
```

```
su - $JON_USER -c "mv jon-server-* $JON_ROOT"
```

```
su - $JON_USER -c "mv rhq* $JON_ROOT"
```

```
configure jon's autoinstall
```

```
sed -i "s/rhq.autoinstall.enabled=false/rhq.autoinstall.enabled=true/" /home/$JON_USER/
$JON_ROOT/bin/rhq-server.properties
```

```
sed -i "s/rhq.server.high-availability.name=/rhq.server.high-availability.name=$HA_NAME/" /home/
$JON_USER/$JON_ROOT/bin/rhq-server.properties
```

```
sed -i "s/rhq.server.database.connection-
```

```
url=jdbc:postgresql://127.0.0.1:5432\rhq/rhq.server.database.connection-
```

```
url=$DB_CONNECTION_URL/" /home/$JON_USER/$JON_ROOT/bin/rhq-server.properties
```

```
sed -i "s/rhq.server.database.server-name=127.0.0.1/rhq.server.database.server-
```

```
name=$DB_SERVER_NAME/" /home/$JON_USER/$JON_ROOT/bin/rhq-server.properties
```

```
copy rhq-server.sh to /etc/init.d
```

```
cp /home/$JON_USER/$JON_ROOT/bin/rhq-server.sh /etc/init.d
```

```
prepend chkconfig preamble
```

```
echo "#!/bin/sh
```

```
#chkconfig: 2345 95 20
```

```
#description: JON Server
```

```
#processname: run.sh
```

```
RHQ_SERVER_HOME=/home/$JON_USER/$JON_ROOT
```

```

RHQ_SERVER_JAVA_HOME=$JAVA_HOME" > /tmp/out
cat /etc/init.d/rhq-server.sh >> /tmp/out
mv /tmp/out /etc/init.d/rhq-server.sh
chmod 755 /etc/init.d/rhq-server.sh

rig JON as a service
echo " * Installing JON as a service..."
chkconfig --add rhq-server.sh
chkconfig rhq-server.sh --list
chkconfig --level 3 rhq-server.sh on
chkconfig --level 5 rhq-server.sh on

install JON license
echo " * Downloading JON license..."
wget $JON_LICENSE_URL -O /home/$JON_USER/
$JON_ROOT/jbossas/server/default/deploy/rhq.ear.rej/license/license.xml

echo " * Starting JON for the first time..."
service rhq-server.sh start

install JON plugins
echo " * Waiting until server installs..."
sleep $AUTOINSTALL_WAITTIME #wait for autoinstall to finish

echo " * Installing plugins..."
for i in /home/$JON_USER/*.zip ; do unzip -d /tmp/rhq-plugins $i *.jar ; done
find /tmp/rhq-plugins -name "*.jar" | xargs -i[] mv [] /home/$JON_USER/
$JON_ROOT/jbossas/server/default/deploy/rhq.ear/rhq-downloads/rhq-plugins
/bin/rm -rf /tmp/rhq-plugins

echo " * Restarting JON..."
service rhq-server.sh stop
service rhq-server.sh start

```

## satellite\_channels.sync

```

#!/bin/bash
#These next line attempts a direct import

Function to use for help throughout the script
help_func() {
 echo
 echo "usage: satellite-sync -e
 satellite-sync -r"
 echo
}

```



```
Check for existence of the satellite file
if [! -f satellite_channels.list]; then
 echo
 echo "The satellite_channels.list file
 does not exist. Please create the file"
 exit 1
 echo
fi

Make sure at least one argument was passed to the script
if ["$#" -eq "0"]; then
 help_func
fi

Loop through the satellite_channels.list file and construct list of channels for satellite-sync
SAT_CHANNELS=""
for channel in $(cat satellite_channels.list)
do
 SAT_CHANNELS="${SAT_CHANNELS} --channel=${channel}"
done
Handle options passed to script
while getopts ":er" opt; do
 case $opt in
 e) echo About to run: satellite-sync ${SAT_CHANNELS} ;;
 r) satellite-sync ${SAT_CHANNELS} ;;
 \?) help_func
 exit 1 ;;
 *) echo 'Use -e or -r'
 exit 1
 esac
done
shift $(($OPTIND - 1))
```

## add-vms.ps1

```
add-vms
tempName - source template (can not be Blank)
baseName - base name of created guest (default: guest)
num - number to create (default: 1)
run -start VMs (default: no)

Param($baseName = 'guest', $tempName, $num = 1, [switch]$run)

if ($tempName -eq $null) {
 write-host "Must specify a template!"
 exit
}
```

```

}

<#
write-host "baseName = $baseName"
write-host "tempName = $tempName"
write-host " num = $num"
write-host " run = $run"
#>

$my_clusId = -1;

$my_temp = select-template -SearchText $tempName
if ($my_temp -eq $null)
{
 Write-host "No matching templates found!"
 exit
} elseif ($my_temp.count -gt 1) {
 Write-host "Too many matching templates found!"
 exit
} elseif ($my_temp.name -eq "Blank") {
 Write-host "Can not use Blank template!"
 exit
}

#search for matching basenames
#search for matching basenames
$matches = select-vm -searchtext "$baseName" | where { $_.name -like "$baseName*" }
if ($matches -ne $null) {
 $measure = $matches | select-object name | foreach { $_.name.Replace("$baseName","") } |
measure-object -max
 $start = $measure.maximum + 1
 $x = $matches | select-object -first 1
 $my_clusId = $x.HostClusterId
} else {
 $start = 1
}

$Sid = $my_temp.HostClusterId

$clus = select-cluster | where { $_.ClusterID -eq $Sid }
if ($clus -ne $null) {
 if ($clus.IsInitialized -eq $true) {
 $my_clusId = $Sid
 } else {
 write-host "Cluster of Template is not initialized!"
 exit
 }
}

```



```
}
}

#loop over adds async
for ($i=$start; $i -lt $start + $num; $i++) {
write-host "-name $baseName$i -templateobject $my_temp -HostClusterId $my_clusId
-copytemplate -Vmtype server"
 if ($run -eq $true) {
 $my_vm = add-vm -name $baseName$i -templateobject $my_temp -HostClusterId $my_clusId
-copytemplate -Vmtype server

 #Until BZ 617730 and 617725 are addressed, use this work around
 $my_vm.DisplayType = 'VNC'
 $uv = update-vm -vmobject $my_vm

 $sv = start-vm -VmObject $my_vm

 } else {
 $my_vm = add-vm -name $baseName$i -templateobject $my_temp -HostClusterId $my_clusId
-copytemplate -Vmtype server -Async

 #Until BZ 617730 and 617725 are addressed, use this work around
 $my_vm.DisplayType = 'VNC'
 $my_vm.DisplayType = 'VNC'

 $uv = update-vm -vmobject $my_vm
 }
}
}
```

## firewall-config.sh

```
#!/bin/bash

PORTS="80 443 50007 50006 50008 50009 21064 11111 5404 5405"
PROTOS="udp tcp"

Backup existing iptables file
cp /etc/sysconfig/iptables{,.orig}

Grab the ports
echo "Please put the ports you would like to firewall here, separated by a space:"
read PORTS

Grab the protocols
echo "Please put the protocols you would like to firewall here, separated by a space:"
read PROTOS
```

```

Create the rhcf chain
iptables --new-chain RHCF
iptables --insert INPUT --jump RHCF

for PROTO in ${PROTOS}; do
 for PORT in ${PORTS}; do
 iptables --append RHCF --protocol ${PROTO} --destination-port ${PORT} --jump ACCEPT
 echo "iptables --append RHCF --protocol ${PROTO} --destination-port ${PORT} --jump
ACCEPT"
 done
done

echo
echo "Remember to \"service iptables save\""
echo

```

### **mrgMgr-config.sh**

```

register system with satellite
rpm -ivh http://RHCF-sat-vm.cloud.lab.eng.bos.redhat.com/pub/rhn-org-trusted-ssl-cert-1.0-1.noarch.rpm
rhnreg_ks --activationkey=1-mrg-manager --serverUrl=https://10.16.139.24/XMLRPC
--sslCACert=/usr/share/rhn/RHN-ORG-TRUSTED-SSL-CERT
rhn_check

#Configure iptables
/bin/cp /etc/sysconfig/iptables /tmp/iptables
cat <<EOF>>/etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -p tcp -m tcp --dport 875 -j ACCEPT
-A INPUT -p udp -m udp --dport 875 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 892 -j ACCEPT
-A INPUT -p udp -m udp --dport 892 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 32769 -j ACCEPT
-A INPUT -p udp -m udp --dport 32769 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 32803 -j ACCEPT
-A INPUT -p udp -m udp --dport 32803 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 662 -j ACCEPT
-A INPUT -p udp -m udp --dport 662 -j ACCEPT
-A INPUT -p udp -m udp --dport 111 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 111 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 2049 -j ACCEPT
-A INPUT -p udp -m udp --dport 2049 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 2020 -j ACCEPT
-A INPUT -p udp -m udp --dport 2020 -j ACCEPT

```



```
-A INPUT -p tcp --dport 4672 -m state --state NEW -j ACCEPT
-A INPUT -p udp --dport 4672 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 5672 -m state --state NEW -j ACCEPT
-A INPUT -p udp --dport 5672 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 45672 -m state --state NEW -j ACCEPT
-A INPUT -p udp --dport 45672 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 9618 -m state --state NEW -j ACCEPT
-A INPUT -p udp --dport 9618 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 9614 -m state --state NEW -j ACCEPT
-A INPUT -p udp --dport 9614 -m state --state NEW -j ACCEPT
-A INPUT -p tcp --dport 9600:9800 -m state --state NEW -j ACCEPT
-A INPUT -p udp --dport 9600:9800 -m state --state NEW -j ACCEPT
COMMIT
EOF
```

```
#tie down nfs ports
cat <<EOF>>/etc/sysconfig/nfs
RQUOTAD_PORT=875
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
STATD_PORT=662
STATD_OUTGOING_PORT=2020
EOF
```

```
setup export for rendering
mkdir /home/admin/render
chown admin:admin /home/admin/render
cat <<EOF>>/etc/exports
/home/admin/render *.cloud.lab.eng.bos.redhat.com(rw,sync,no_root_squash)
EOF
```

```
#Add mrgmgr user
#useradd mrgmgr
```

```
#Turn on Services
chkconfig sesame on
chkconfig postgresql on
chkconfig condor on
chkconfig qpidd on
chkconfig cumin on
chkconfig ntpd on
chkconfig nfs on
```

```
#Postgresql funkiness
rm -rf /var/lib/pgsql/data
su - postgres -c "initdb -D /var/lib/pgsql/data"
```

```

update software
yum -y update

#prepare actions to be performed on next boot
wget http://10.16.136.1/pub/resources/mrggrid.rc.local.add -O /etc/rc.d/mrggrid.rc.local.add
/bin/cp /etc/rc.d/rc.local /etc/rc.d/rc.local.shipped
cat /etc/rc.d/mrggrid.rc.local.add >> /etc/rc.d/rc.local
) >> /root/ks-post.log 2>&1

configure cumin.conf

cat <<EOF>>/etc/sysconfig/iptables
Fields in comments reflect the default values

[common]
database: dbname=cumin user=cumin host=localhost
brokers: localhost:5672
log-level: debug
[web]
log-file: $CUMIN_HOME/log/web.log
host: localhost ('0.0.0.0' binds to all local interfaces)
port: 45672
operator-email: [none]
update-interval: 10
host: 0.0.0.0
ssl: yes

[data]
log-file: $CUMIN_HOME/log/data.log
expire-interval: 3600
expire-threshold: 86400
vacuum-interval: 3600
packages: [all]
#packages: com.redhat.grid, com.redhat.sesame
EOF

```

## MRG Execution Node Configuration File

```

This config disables advertising to UW's world collector. Changing
this config option will have your pool show up in UW's world
collector and eventually on the world map of Condor pools.
CONDOR_DEVELOPERS = NONE

CONDOR_HOST = mrgmgr.rhcf.lab
COLLECTOR_HOST = $(CONDOR_HOST)
COLLECTOR_NAME = RHCF Grid

```



```
FILESYSTEM_DOMAIN = $(FULL_HOSTNAME)
UID_DOMAIN = $(FULL_HOSTNAME)

START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
ALLOW_WRITE = $(FULL_HOSTNAME), *.rhcf.lab
DAEMON_LIST = MASTER, STARTD
NEGOTIATOR_INTERVAL = 20
TRUST_UID_DOMAIN = TRUE
IN_HIGHPORT = 9700
IN_LOWPORT = 9600

Plugin configuration
MASTER.PLUGINS = $(LIB)/plugins/MgmtMasterPlugin-plugin.so
QMF_BROKER_HOST = mrgmgr.rhcf.lab
```

## MRG Manager Configuration File

```
This config disables advertising to UW's world collector. Changing
this config option will have your pool show up in UW's world
collector and eventually on the world map of Condor pools.
CONDOR_DEVELOPERS = NONE

COLLECTOR_NAME = REFARCH at $(FULL_HOSTNAME)
MASTER_DEBUG = D_FULLDEBUG D_COMMAND
COLLECTOR_DEBUG = D_FULLDEBUG D_COMMAND
NEGOTIATOR_DEBUG = D_FULLDEBUG D_COMMAND
SCHEDD_DEBUG = D_FULLDEBUG D_COMMAND

Accept TCP updates, necessary because default is UDP
and the relay tunnel is only TCP
COLLECTOR_SOCKET_CACHE_SIZE = 1024

Setting CCB_ADDRESS results in deadlock because
the Collector does not realize it is making a blocking
connection to itself via the CCB_ADDRESS. So, enable CCB
for only the Shadow, which needs to receive a connection
from the Starter for file transfer.
#SHADOW.CCB_ADDRESS = $(PUBLIC_HOST):$(PUBLIC_PORT)
#SCHEDD.CCB_ADDRESS = $(PUBLIC_HOST):$(PUBLIC_PORT)
CCB_ADDRESS = $(PUBLIC_HOST):$(PUBLIC_PORT)
COLLECTOR.CCB_ADDRESS =
```

```

Avoid needing CCB within the VPN
PRIVATE_NETWORK_NAME = mrgmgr

Set TCP_FORWARDING_HOST so CCB will advertise its public
address. Without this, it will advertise its private address
and the Starter will not be able to connect to reverse its
connection to the Shadow.
COLLECTOR.TCP_FORWARDING_HOST = $(PUBLIC_HOST)
As per TCP_FORWARDING_HOST semantics, the local port for
the Collector/CCB must match the relay port
COLLECTOR_HOST = $(FULL_HOSTNAME):$(PUBLIC_PORT)

Give access to relayed communication
ALLOW_WRITE = $(ALLOW_WRITE), $(PRIVATE_HOST)
CONDOR_HOST = $(FULL_HOSTNAME)
COLLECTOR_NAME = RHCF Grid
COLLECTOR_HOST = $(CONDOR_HOST)
NEGOTIATOR_HOST = $(CONDOR_HOST)
UID_DOMAIN = rhcf.lab
FILESYSTEM_DOMAIN = rhcf.lab
START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
ALLOW_WRITE = *.rhcf.lab, $(PRIVATE_HOST)
ALLOW_READ = *.rhcf.lab
DAEMON_LIST = COLLECTOR, MASTER, NEGOTIATOR, SCHEDD
NEGOTIATOR_INTERVAL = 20
TRUST_UID_DOMAIN = TRUE
IN_HIGHPORT = 9800
IN_LOWPORT = 9600
SCHEDD.PLUGINS = $(LIB)/plugins/MgmtScheddPlugin-plugin.so
COLLECTOR.PLUGINS = $(LIB)/plugins/MgmtCollectorPlugin-plugin.so
NEGOTIATOR.PLUGINS = $(LIB)/plugins/MgmtNegotiatorPlugin-plugin.so
Plugin configuration
MASTER.PLUGINS = $(LIB)/plugins/MgmtMasterPlugin-plugin.so
QMF_BROKER_HOST = rhcf-mrgmgr.rhcf.lab

```

### **perfect\_number.sub**

```

Universe = vanilla
Requirements = Arch != Undefined && FileSystemDomain != Undefined
Requirements = Arch != Undefined
Executable = /home/admin/perfect1
Arguments = 1 1000000
Log = /home/admin/perfect/output/perfect.log
Output = /home/admin/perfect/output/perfect.out
Error = /home/admin/perfect/output/perfect.err

```



```
should_transfer_files = YES
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_executable = true
QUEUE
```

- i <http://csrc.nist.gov/groups/SNS/cloud-computing/>
- ii <http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc>
- iii <http://www.redhat.com/rhel/server/>
- iv [http://www.redhat.com/red\\_hat\\_network/](http://www.redhat.com/red_hat_network/)
- v <http://www.redhat.com/jboss/>
- vi <http://www.redhat.com/mrg/>
- vii <http://www-03.ibm.com/systems/bladecenter/hardware/servers/hs22/index.html?>
- viii [http://docs.redhat.com/docs/en-US/Red\\_Hat\\_Enterprise\\_Linux/96/html/Security\\_Guide/index.html](http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/96/html/Security_Guide/index.html)
- ix [https://access.redhat.com/knowledge/docs/manuals/Red\\_Hat\\_Network\\_Satellite/](https://access.redhat.com/knowledge/docs/manuals/Red_Hat_Network_Satellite/)
- x [http://docs.redhat.com/docs/en-US/Red\\_Hat\\_Network\\_Satellite/5.4/pdf/Installation\\_Guide/Red\\_Hat\\_Network\\_Satellite-5.4-Installation\\_Guide-en-US.pdf](http://docs.redhat.com/docs/en-US/Red_Hat_Network_Satellite/5.4/pdf/Installation_Guide/Red_Hat_Network_Satellite-5.4-Installation_Guide-en-US.pdf)
- xi <http://wincp.net/eng/index.php>
- xii [https://access.redhat.com/knowledge/docs/Red\\_Hat\\_Enterprise\\_Virtualization\\_for\\_Servers/](https://access.redhat.com/knowledge/docs/Red_Hat_Enterprise_Virtualization_for_Servers/)
- xiii [https://access.redhat.com/knowledge/docs/Red\\_Hat\\_Enterprise\\_Linux/](https://access.redhat.com/knowledge/docs/Red_Hat_Enterprise_Linux/)