Red Hat Reference Architecture Series

Red Hat Cloud Foundations Reference Architecture

Hybrid IaaS Clouds

Version 1.0
August 2010

redhat.

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 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

Red Hat's suite of open source software provides a rich infrastructure for cloud providers to build public/private/hybrid cloud offerings.

The first Red Hat Cloud Foundations document provided the infrastructure for a Private IaaS Clouds.

This document extends that functionality by describing the foundation for building Hybrid IaaS Clouds:

1. Recap the procedures for deploying **Red Hat Cloud Foundations: Private IaaS Clouds**
   - Deployment of infrastructure management services, e.g., Red Hat Network (RHN) Satellite, Red Hat Enterprise Virtualization (RHEV) Manager (RHEV-M), DNS service, DHCP service, PXE server, NFS server for ISO images, JON, MRG Manager - most of them installed in virtual machines (VMs) in a Red Hat Cluster Suite (RHCS) cluster for high availability.
   - Deployment of a farm of RHEV host systems (either in the form of RHEV Hypervisors or as RHEL+KVM) to host tenants' VMs.
   - Demonstrate sample RHEL application(s), JBoss application(s) and MRG Grid application(s) respectively in the tenant VMs.
2. Describe the procedures for deploying **Red Hat Cloud Foundations: Hybrid IaaS Clouds**
   - Deploy MRG Grid application instances in RHEL VMs in a configuration with a single hypervisor host (of each type – RHEL+ KVM host and RHEV-H host) in a private cloud.
   - As demand increases, deploy additional MRG Grid application instances in RHEL VMs in a configuration with a multiple hypervisor host (of each type – RHEL+ KVM host and RHEV-H host) in a private cloud.
   - As demand increases even more, deploy even more MRG Grid application instances in RHEL VMs in a configuration including Amazon's Elastic Compute Cloud (EC2) instances in a hybrid cloud configuration.

The execute nodes that comprise the MRG Grid environment continuously de-queue and execute items off a work queue. The more MRG execute nodes there are, the shorter the elapsed time to empty the work queue and complete the application.

Initially the configuration starts with MRG Grid execute nodes running in RHEV VMs in a RHEV-based Private IaaS Cloud. As more VMs are added, the application completes in lesser time (i.e., the work queue is emptied faster) as illustrated in Figure **Error: Reference source not found**.

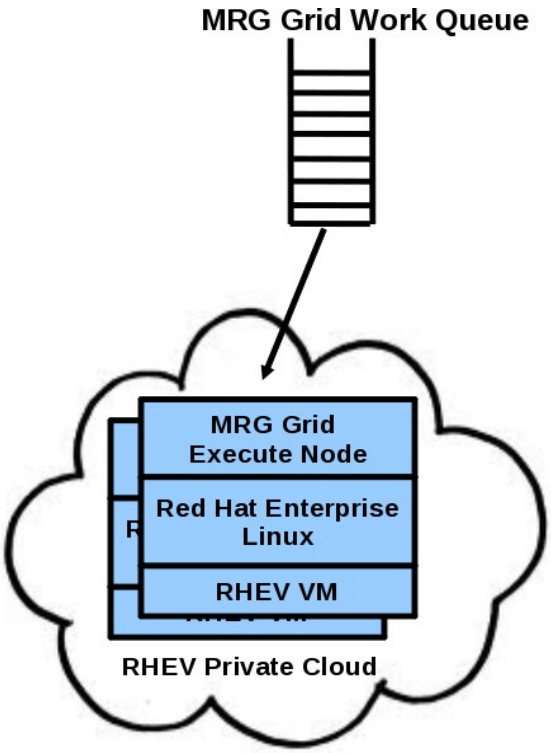


Figure 1

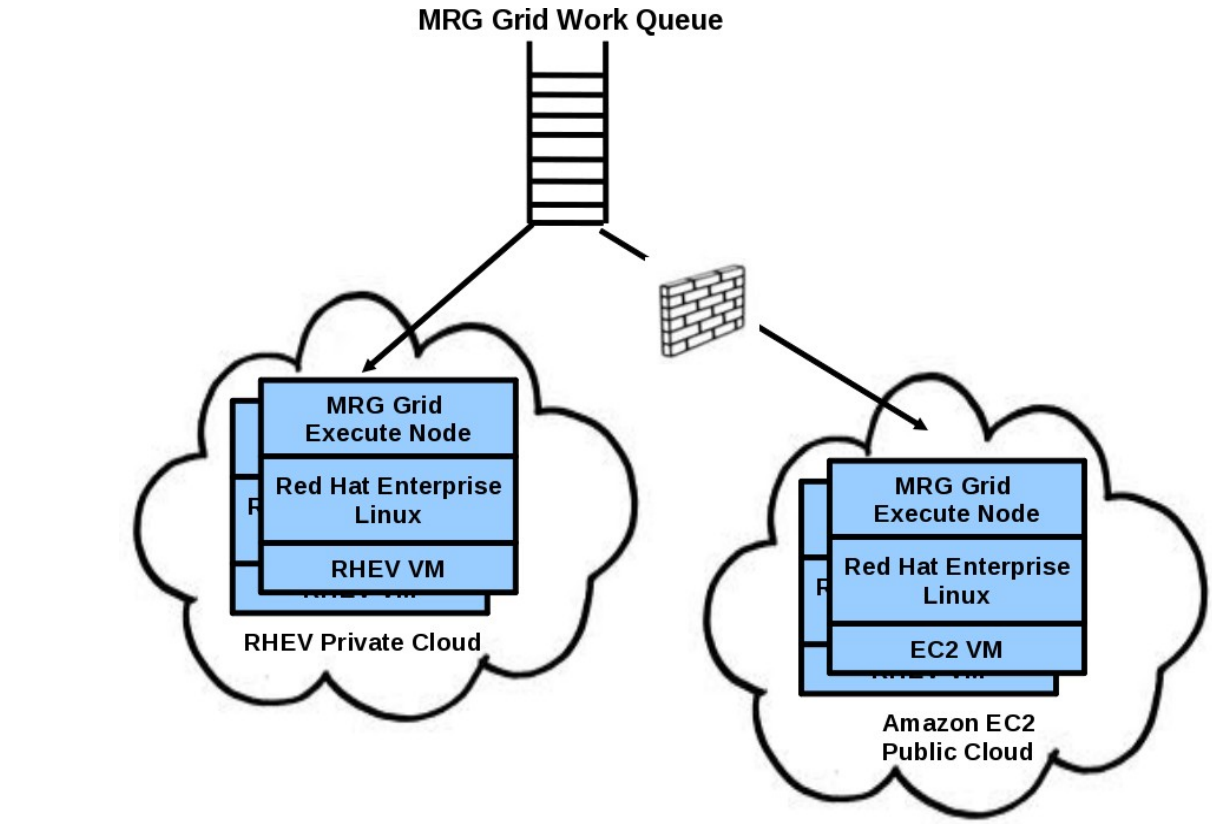


Figure 2

If the allocated resources in the private cloud are insufficient, a hybrid cloud configuration can be implemented using Amazon EC2 VMs in conjunction with RHEV VMs to further reduce the application time to completion as illustrated in Figure **2**.

Section 2 presents some commonly used definitions of cloud computing.

Section 3 discusses the phased adoption of cloud computing by enterprises from the use of virtualization, to the deployment of internal clouds and leading to full-functional utility computing using private and public clouds.

Section 4 describes the software infrastructure for Red Hat Cloud Foundations.

Section 5 describes the software, server, storage, and network configuration used for this proof-of-concept.

Section 6 is a quick overview of Red Hat Cloud Foundation: Private IaaS Cloud published in a separate reference architecture paper.

Sections 7 , 8 and 11 describe the detailed steps for deploying Red Hat Cloud Foundations: Hybrid IaaS Clouds.

Section 9 provides an overview of Amazon's Elastic Compute Cloud (EC2) including procedures to configure an AWS account, configure an EC2 environment, and start EC2

instances.

Section 10 defines the workload used.

Section 12 lists referenced documents.

Future versions of the Red Hat Cloud Reference Architecture take these concepts further:

- Red Hat Cloud Reference Architecture: Adding self-service
- Red Hat Cloud Reference Architecture: Managing mixed private clouds
- Red Hat Cloud Reference Architecture: Adding public clouds
- Red Hat Cloud Reference Architecture: Creating large-scale clouds

2 Cloud Computing: Definitions

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. This cloud model promotes availability and is composed of five essential **characteristics,** three **service models**, and four **deployment models**. The following definitions have been proposed by National Institute of Standards and Technology *(*NIST) in the document found at http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

2.1 Essential Characteristics

Cloud computing creates an illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning.

2.1.1 *On-demand Self-Service*

A 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.

2.1.2 Resource Pooling

The computing resources of the provider are pooled to serve 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 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.

2.1.3 *Rapid Elasticity*

Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released 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.

2.1.4 Measured Service

Cloud systems automatically control and optimize resource use 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). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

2.2 Service Models

2.2.1 Cloud Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and invoke arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

2.2.2 Cloud Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

2.2.3 Cloud Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

2.2.4 Examples of Cloud Service Models

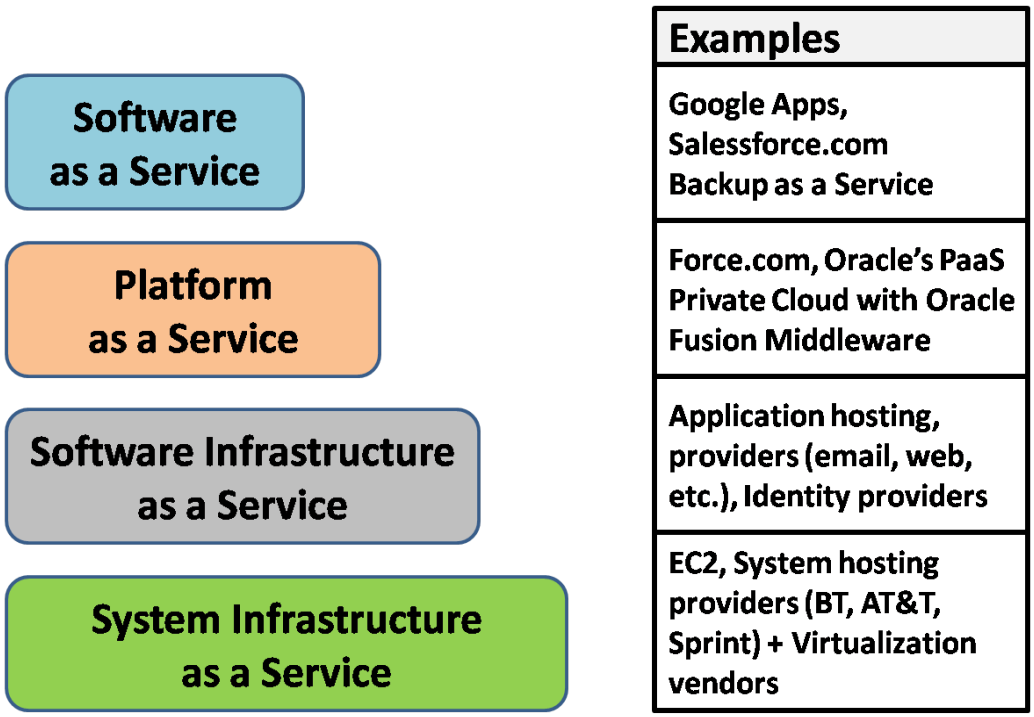


Figure 3

2.3 Deployment Models

2.3.1 Private Cloud

The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on or off premise.

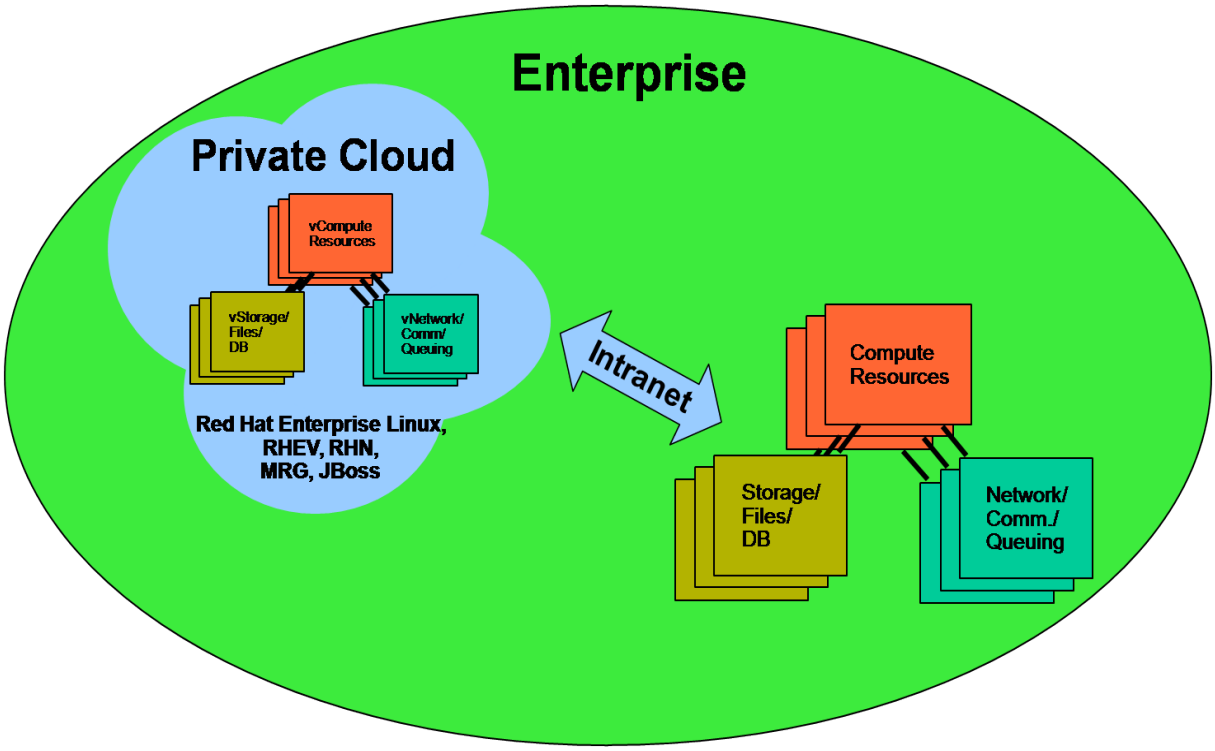


Figure 4

2.3.2 Public Cloud

The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

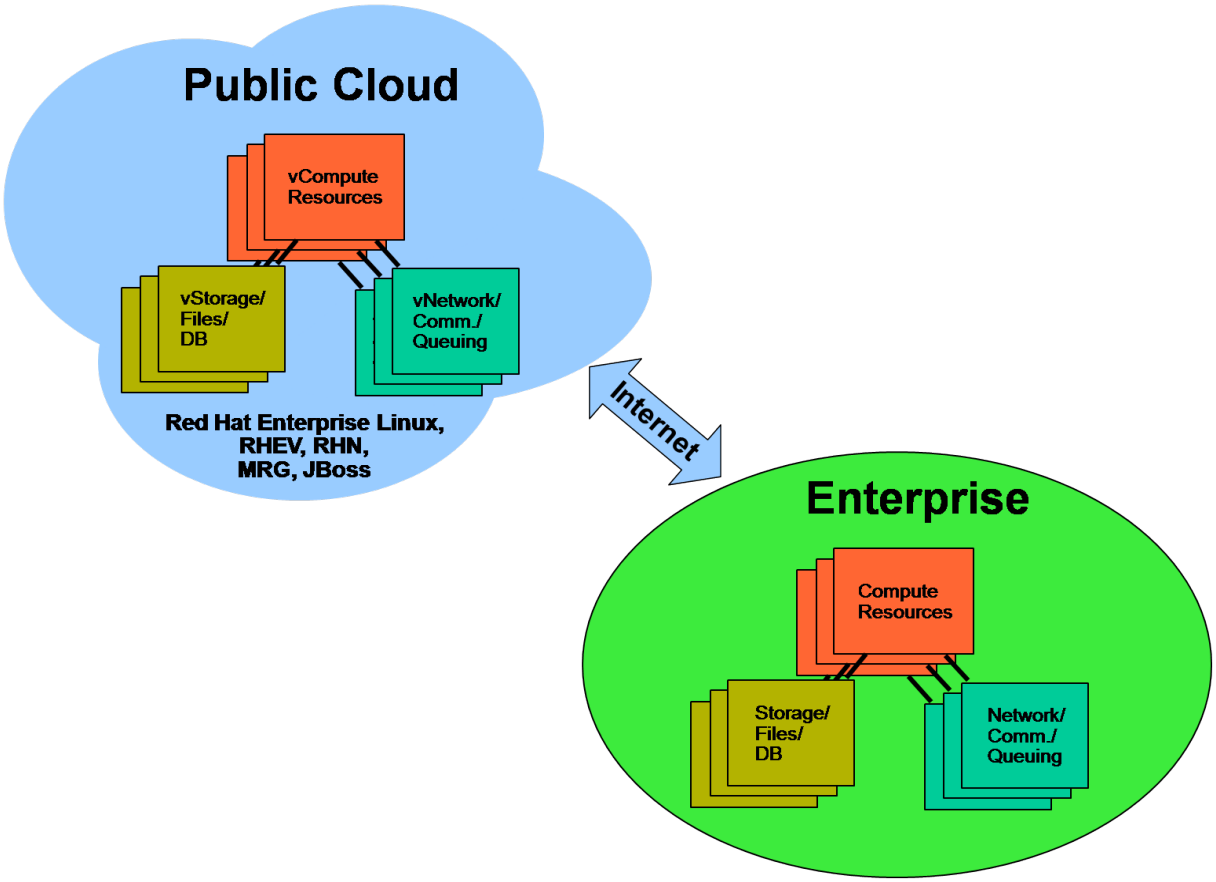


Figure 5

2.3.3 Hybrid Cloud

The cloud infrastructure 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., load-balancing between clouds).

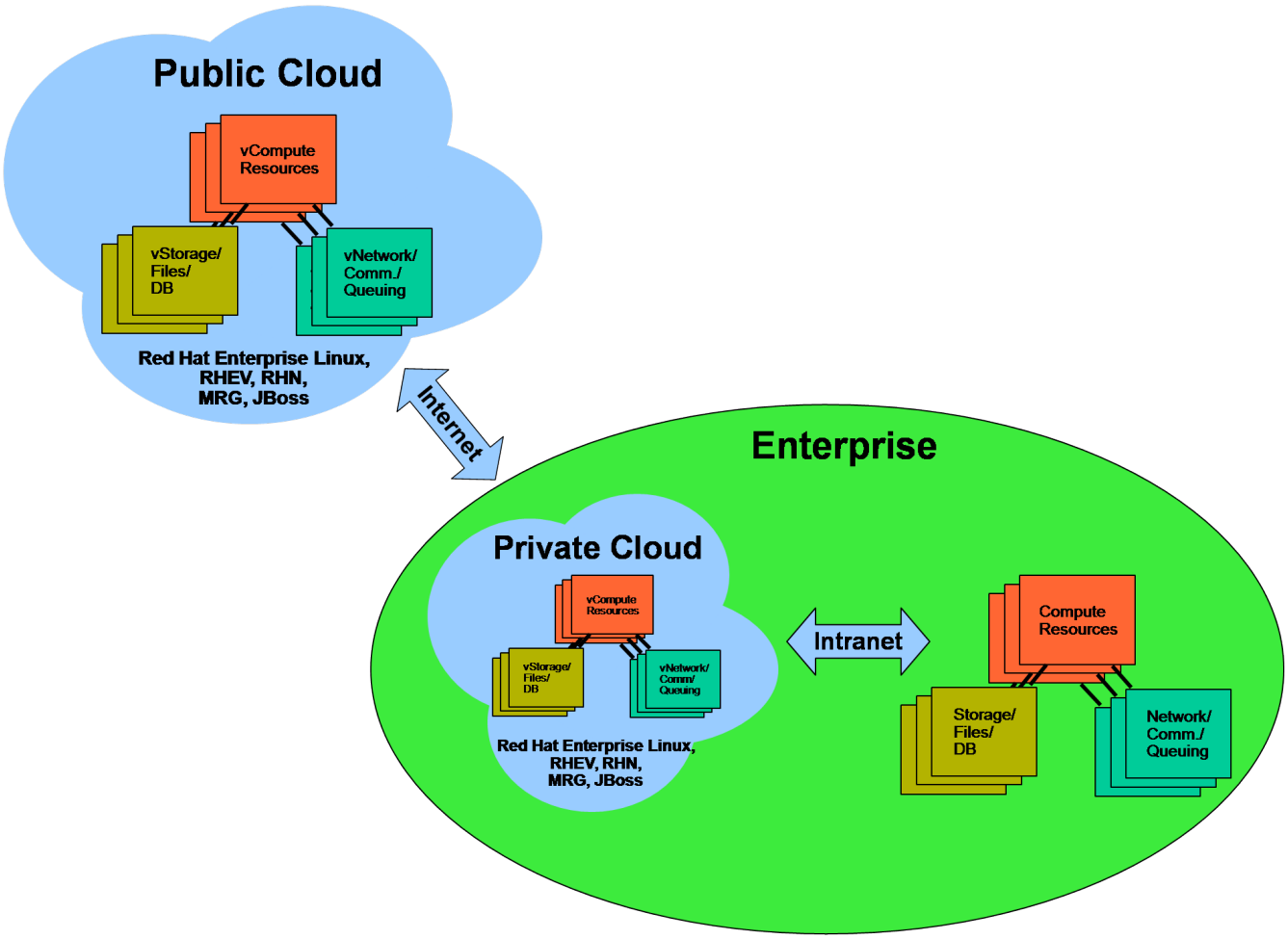


Figure 6

2.3.4 Community Cloud

The cloud infrastructure 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 organizations or a third party and may exist on or off premise.

3 Red Hat and Cloud Computing

3.1 Evolution, not Revolution – A Phased Approach to Cloud Computing

While cloud computing requires virtualization as an underlying and essential technology, it is inaccurate to equate cloud computing with virtualization. The figure below displays the different levels of abstraction addressed by virtualization and cloud computing respectively.

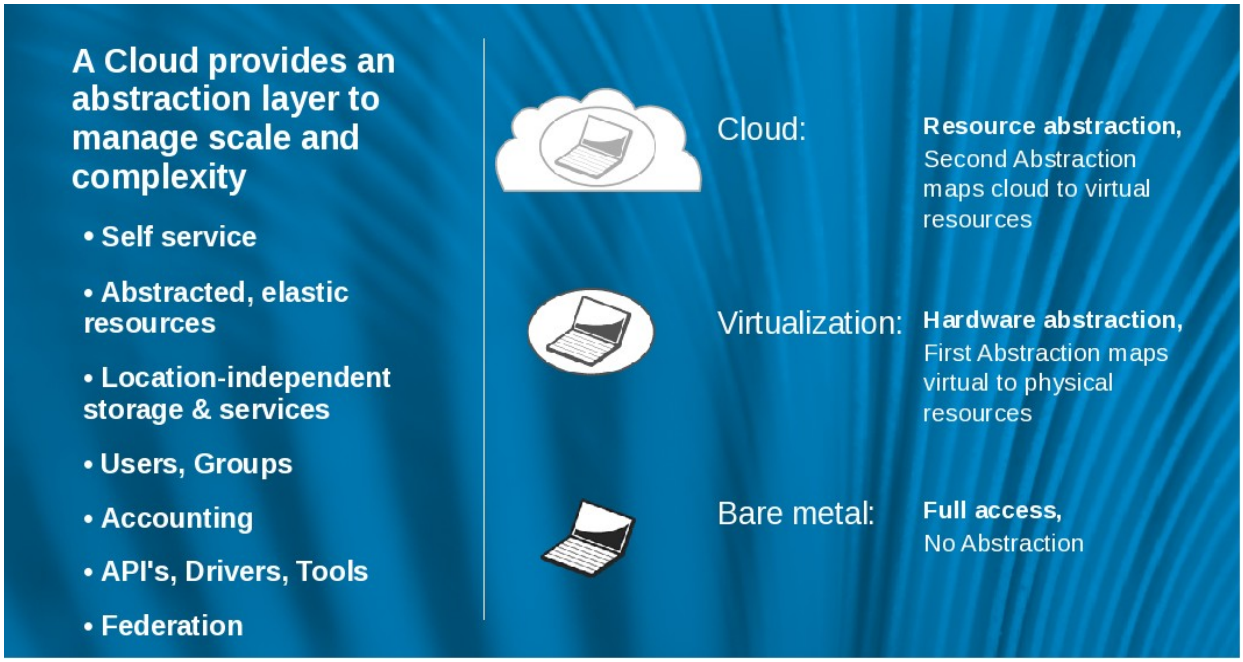


Figure 7: Levels of Abstraction

The following figure 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 8: Phases of Technology Adoption in the Enterprise

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 partners, and
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 existing investment and manages diverse investments as one cloud -- whether Linux or 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.

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.

Open source 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 leading work on 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:

- Deltacloud - Abstracts the differences between clouds
- BoxGrinder - A set of projects to build appliances for a multitude of virtualization fabrics
- Cobbler - Installation server for rapid set up of network installation environment
- Condor - Batch system managing millions of machines worldwide
- CoolingTower - Simple application-centric tool for deploying applications in the cloud
- Hail - Umbrella cloud computing project for cloud services
- Infinispan - Extremely scalable, highly available data grid platform
- Libvirt - Common, generic, and scalable layer to securely manage domains on a node
- Spice - Open remote computing solutions for interaction with virtualized desktop devices
- Thincrust - Tools to build appliances for the cloud

3.3.1 Deltacloud

The goal of Deltacloud is simple: making many clouds function as one. Deltacloud strives to bridge the differences between diverse silos of infrastructure, allowing them to be managed as one. Organizations today may have different clouds built on, for example, RHEV-M or VMware vCloud. The Deltacloud project is designed to make them manageable as one cloud, one pool of resources. Organizations may wish to use internal cloud capacity, as well as public clouds like Amazon's EC2, and perhaps capacity from other IaaS providers.

Today each IaaS cloud presents a unique API to which developers and ISVs need to write in order to consume the cloud service. The Deltacloud effort is creating a common, REST-based API, such that developers can write once and manage anywhere. Deltacloud is cloud broker, so to speak, with drivers that map the API to both public clouds like EC2 and private virtualized clouds based on VMware, vCloud, or RHEV-M as depicted in Figure **13**.

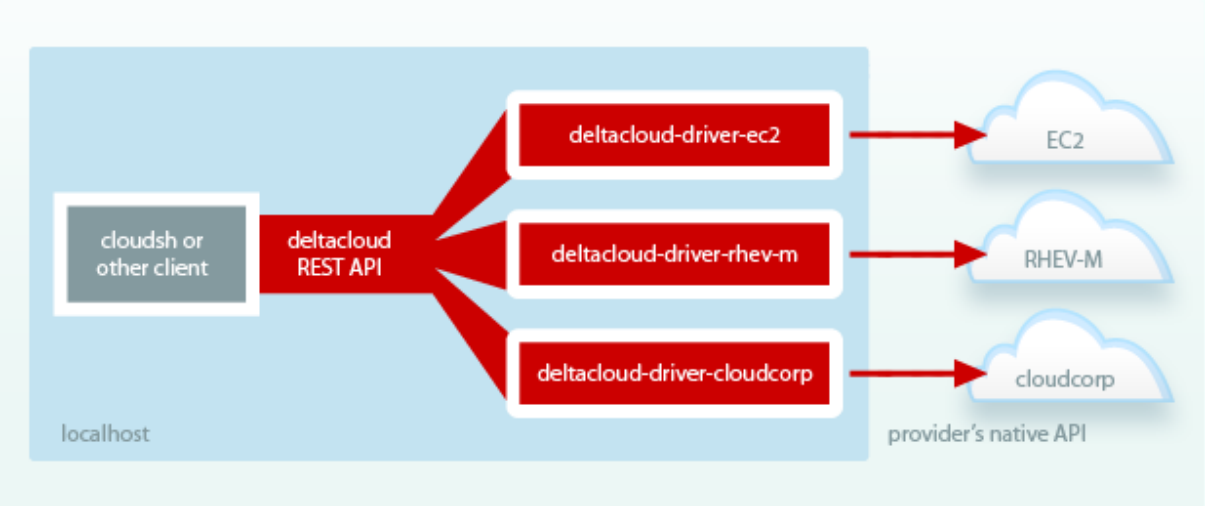


Figure 9: Deltacloud Overview

One level up, *Deltacloud Aggregator* provides a web UI in front of the Deltacloud API. With Deltacloud Aggregator users can:

- View image status and stats across clouds, all in one place
- Migrate instances from one cloud to another
- Manage images locally and provision them on any cloud

To learn more about the Deltacloud project, visit http://deltacloud.org.

4 Red Hat Cloud: Software Stack and Infrastructure Components

Figure **1** depicts the software stack of Red Hat Cloud Foundation components.

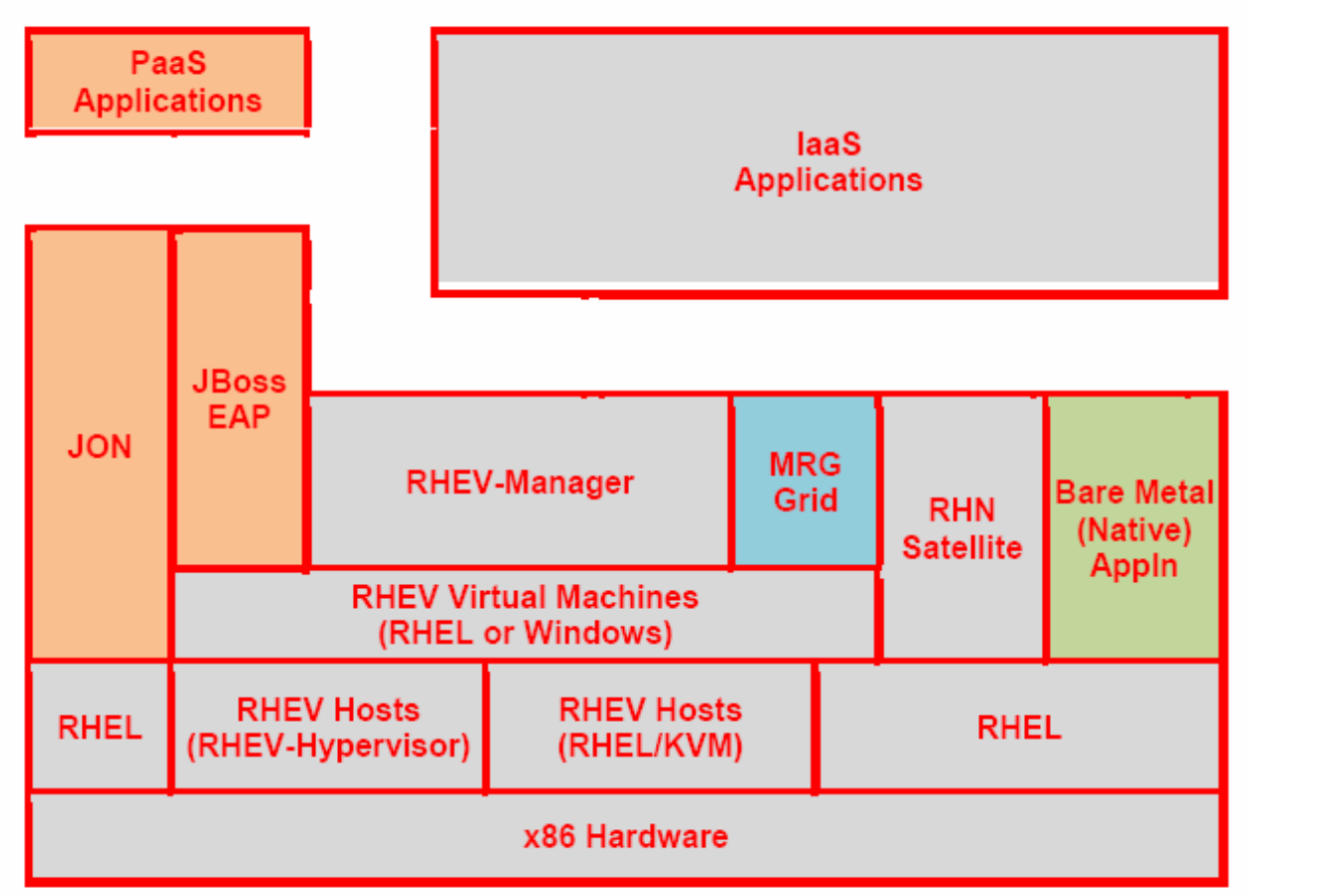


Figure 10: Red Hat Software Stack

4.1 Red Hat Enterprise Linux

Red Hat Enterprise Linux (RHEL) 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 fifth update to RHEL 5: Red Hat Enterprise Linux 5.5.

RHEL 5.5 is designed to support newer Intel Xeon® Nehalem-EX platform as well as the AMD Opteron™ 6000 Series platform (formerly code named “Magny-Cours”). The new platforms leverage 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. To increase the reliability of these systems, Red Hat supports Intel’s expanded machine check architecture, CPU fail-over and memory sparing.

Red Hat also continues to make enhancements to our virtualization platform. New to the RHEL 5.5 is support for greater guest density, meaning that more virtual machines can be supported on each physical server. Our internal testing to date has shown that this release can support significantly more virtual guests than other virtualization products. The new hardware and protocols included in the latest release significantly improve networking scaling by providing direct access from the guest to the network.

RHEL 5.5 also introduces improved interoperability with Microsoft Windows 7 with an update to Samba. This extends the Active Directory integration to better map users and groups on Red Hat Enterprise Linux systems and simplifies managing file systems across platforms.

An important feature of any RHEL update is that kernel and user application programming interfaces (APIs) remain unchanged, ensuring RHEL 5 applications do not need to be rebuilt or re-certified. The unchanged kernel and user APIs also extend to virtualized environments. With a fully integrated hypervisor, the application binary interface (ABI) consistency offered by RHEL means that applications certified to run on RHEL on physical machines are also certified when run on virtual machines. With this, the portfolio of thousands of certified applications for Red Hat Enterprise Linux applies to both environments.

4.2 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 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 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): Windows 2003 server, Windows 2008 server, 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.

4.3 Red Hat Network (RHN) Satellite

All RHN 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:

- An embedded database to store packages, profiles, and system information.
- Instantly update systems for security fixes or to provide packages or applications needed immediately.
- API layer allows the creation of scripts to automate functions or integrate with existing management applications.
- Distribute custom or 3rd party applications and updates.
- Create staged environments (development, test, production) to select, manage and test content in a structured manner.
- Create errata for custom content, or modify existing errata to provide specific information to different groups.
- 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-premises 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.

In September 2009, Red Hat released RHN Satellite 5.3, the first fully open source version of the product. This latest version offers opportunities for increased flexibility and faster provisioning setups for customers with the incorporation of open source Cobbler technology in its provisioning architecture.

4.3.1 Cobbler

Cobbler is a Linux installation server that allows for rapid setup of network installation environments. It binds and automates many associated Linux tasks, eliminating the need for many various commands and applications when rolling out new systems and, in some cases, changing existing ones. With a simple series of commands, network installs can be configured for PXE, re-installations, media-based net-installs, and virtualized installs (supporting Xen and KVM).

Cobbler can also optionally help with managing DHCP, DNS, and yum package mirroring infrastructure. In this regard, it is a more generalized automation application, rather than just dealing specifically with installations. There is also a lightweight built-in configuration management system as well as support for integrating with other configuration management systems. Cobbler has a command line interface as well as a web interface and several API access options.

4.4 JBoss Enterprise Middleware

The following JBoss Enterprise Middleware Development Tools, Deployment Platforms and Management Environment 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.

4.4.1 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

4.4.2 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 Red Hat Enterprise MRG Grid

MRG Grid provides high throughput and high performance computing. Additionally, it enables enterprises to move to a utility model of computing to help enterprises achieve both higher peak computing capacity and higher IT utilization 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, utilization, 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, utilization, 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 baremetal hardware to cloud-based infrastructure.

5 Reference Architecture System Configuration

This reference architecture in deploying the Red Hat infrastructure for a private cloud used the configuration shown in Figure **11** comprised of:

1. Infrastructure management services, e.g., Red Hat Network (RHN) Satellite, Red Hat Enterprise Virtualization Manager (RHEV-M), DNS service, DHCP service, PXE server, NFS server for ISO images, JON, MRG Manager - most of which were installed in virtual machines (VMs) in a RHCS cluster for high availability.
2. A farm of RHEV host systems (either in the form of RHEV Hypervisors or as RHEL+KVM) to host tenants' VMs.
3. Sample RHEL application(s), JBoss application(s) and MRG Grid application(s) deployed in the tenant VMs.

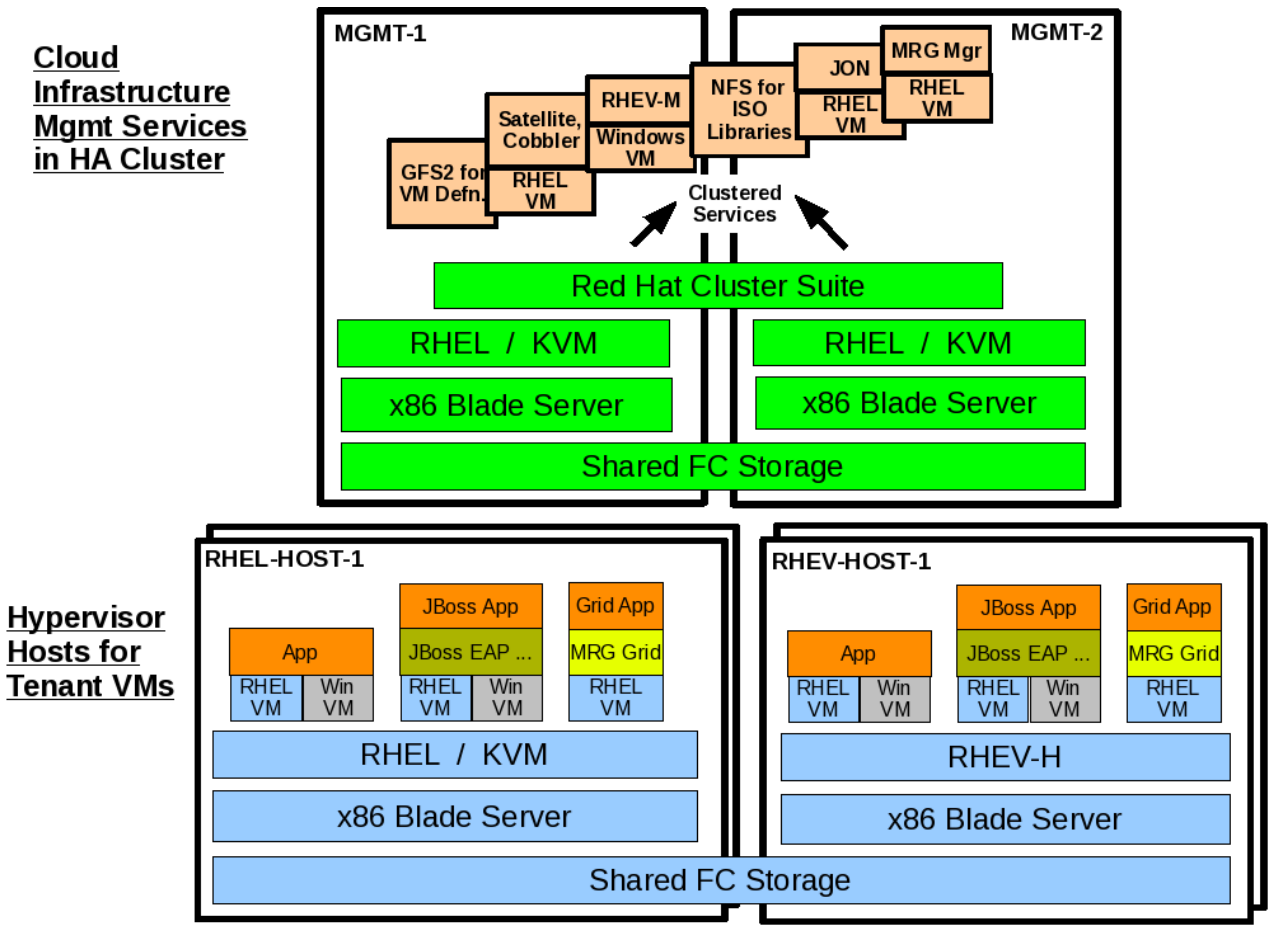


Figure 11

5.1 Server Configuration

Hardware Systems	Specifications
Management Cluster Nodes [2 x HP ProLiant BL460c G6]	Quad Socket, Quad Core (16 cores) Intel® Xeon® CPU X5550 @2.67GHz, 48GB RAM
	2 x 146 GB SATA SSD internal disk drive (mirrored)
	2 x QLogic ISP2532-based 8Gb FC HBA
	2 x Broadcom NetXtreme II BCM57711E Flex-10 10Gb Ethernet Controller
Hypervisor Host Systems [2+ x HP ProLiant BL460c G6]	Quad Socket, Quad Core, (16 cores) Intel® Xeon® CPU W5550 @2.67GHz, 48GB RAM
	2 x 146 GB SATA SSD internal disk drive (mirrored)
	2 x QLogic ISP2532-based 8Gb FC HBA
	2 x Broadcom NetXtreme II BCM57711E Flex-10 10Gb Ethernet Controller

Table 1: Hardware Configuration

5.2 Software Configuration

Software	Version
Red Hat Enterprise Linux (RHEL)	5.5 (2.6.18-194.8.1.el5 kernel)
Red Hat Enterprise Virtualization Manager (RHEV-M)	2.2.0.46267
Red Hat Enterprise Virtualization Hypervisor (RHEV-H)	5.5-2.2 - 5.2
Red Hat Enterprise Linux / KVM (RHEL / KVM)	5.5.0.2 / 83-164
Red Hat Network (RHN) Satellite	5.3.0
JBoss Enterprise Application Platform (EAP)	5.0
JBoss Operations Network (JON)	1.4
Red Hat Enterprise MRG Grid	1.2 (local)
	1.3b (EC2)

Table 2: Software Configuration

5.3 Blade and Virtual Connect Configuration

All the blades are using logical Serial Numbers, MAC addresses and FC WWNs. A single 10Gb network and two 8 Gb FC connections are presented to each host.

Complete details of the blade and virtual connection configuration can be found in the **Cloud Foundations: Private IaaS Clouds – Automating Deployment** reference architecture.

5.4 Storage Configuration

Hardware	Specifications
1 x HP StorageWorks MSA2324fc Fibre Channel Storage Array + HP StorageWorks 70 Modular Smart Array with Dual Domain IO Module [total 49 x 146GB 10K RPM SAS disks]	Storage Controller: Code Version: M110R28 Loader Code Version: 19.009
	Memory Controller: Code Version: F300R22
	Management Controller Code Version: W441R13 Loader Code Version: 12.015
	Expander Controller: Code Version: 1106
	CPLD Code Version: 8
	Hardware Version: 56
1 x HP StorageWorks 4/16 SAN Switch	Firmware: v6.2.2c
1 x HP StorageWorks 8/40 SAN Switch	Firmware: v6.4.0a

Table 3: Storage Hardware

The MSA2324fc array was configured with four 11-disk RAID6 vdisks, with online replacement spares.

LUNs were created and presented as outlined in the following table.

Volume	Size	Presentation	Purpose
MgmtServices	1 TB	Management Cluster	Volume Group for Logical Volumes: • SatVMvol (300GB) • JonVMvol (40GB) • MRGVMvol (40GB) • RHEVMVMvol (30GB) • RHEVNFSvol (300GB)
GFS2	50 GB	Management Cluster	VM Configuration File Shared Storage
RHEVStorage1	1 TB	Hypervisor Hosts	RHEV-M Storage Pool

Table 4: LUN Configuration

5.5 Network Configuration

All the systems in the test environment were assigned to an unique VLAN. Its use allows local control of the network while supplying gateway access to the corporate network. DHCP, DNS, and PXE were all controlled within the VLAN. The VLAN was assigned 10.16.136/21 providing approximately 2000 addresses.

6 Red Hat Cloud Foundations: Private IaaS Clouds

This section provides a set of detail actions required to configure Red Hat products that constitute the infrastructure for Red Hat Cloud Foundations.

The goal is to create a set of highly available cloud infrastructure management services. These cloud management services are used to configure the cloud hosts, create the VMs within those hosts, and load applications onto those VMs.

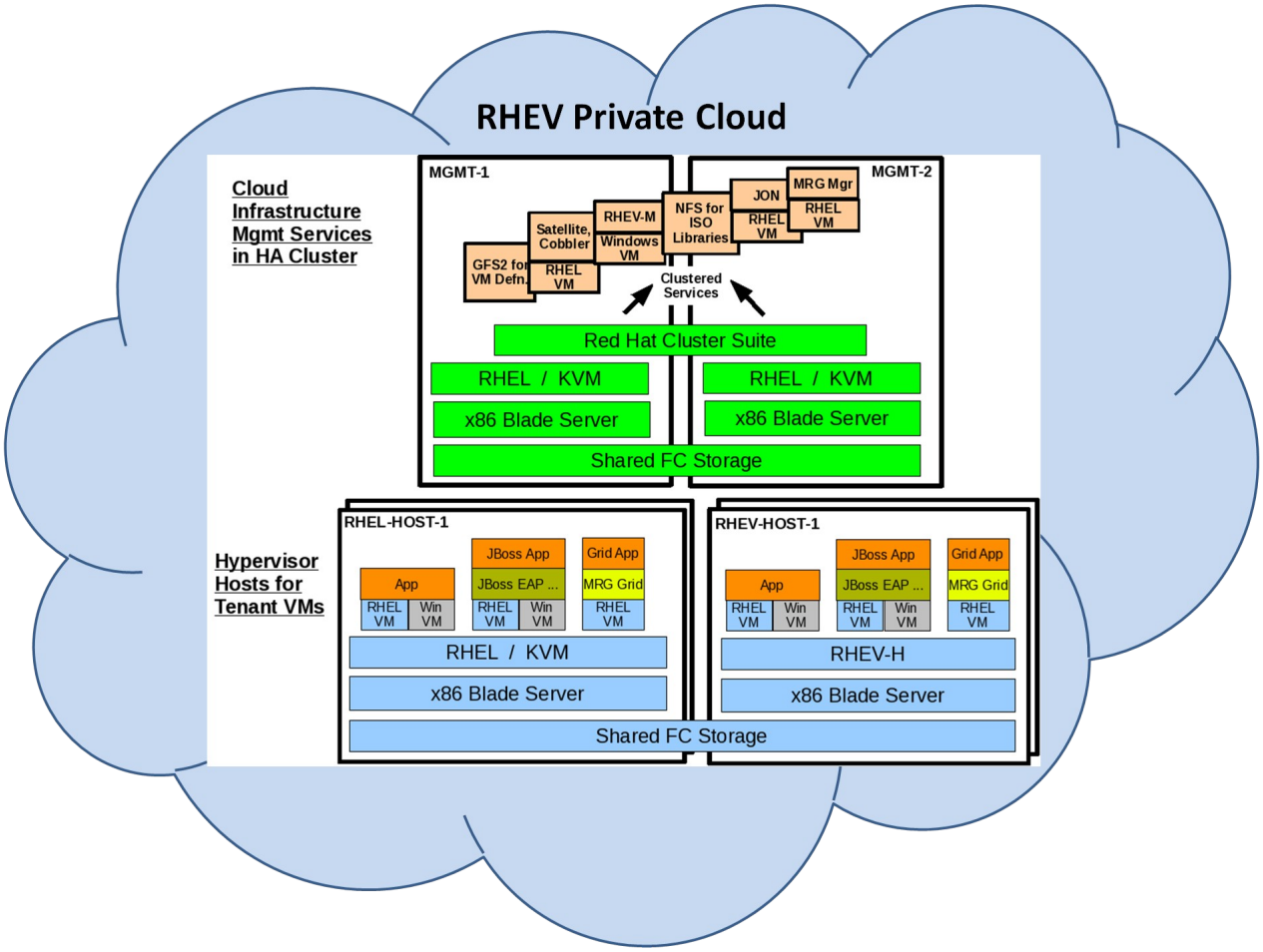


Figure 12: Private Cloud

High availability is achieved by clustering two RHEL nodes (active / passive) using RHCS. Each of the cluster nodes is configured to run RHEL 5.5 with the bundled KVM hypervisor. For most management services, a VM is created using the KVM hypervisor and configured as a RHCS service. Then the management service in installed in the VM (e.g., RHN Satellite VM, MRG VM, etc.).

The procedural walk-through to create these highly available cloud infrastructure management services is detailed in either of the reference architecture documents:

- *Red Hat Cloud Foundations: Private IaaS Clouds*
- *Red Hat Cloud Foundations: Private IaaS Clouds – Automating Deployment*

The former details the individual steps while the latter details automating the entire procedure using APIs and command line scripting. Each of them include the steps to:

1. Install RHEL + KVM on a node
2. Use `virt-manager` to create a VM
3. Install RHN Satellite in the VM (= Satellite VM)
4. Synchronize Satellite with RHN & download packages from all appropriate channels / child channels:
   - Base RHEL 5
   - Clustering (RHCS, …)
   - Cluster storage (GFS, …)
   - Virtualization (KVM, …)
   - RHN Tools
   - RHEV management agents for RHEL hosts
5. Use multi-organization support in Satellite - create a ‘Tenant’ organization and ‘Management’ organization
6. Configure cobbler
   - Configure cobbler’s management of DHCP
   - Configure cobbler’s management of DNS
   - Configure cobbler’s management of PXE
7. Provision MGMT-1 node from Satellite
8. Migrate the Satellite VM to MGMT-1
9. Provision additional cloud infrastructure management services on MGMT-1 (using Satellite where applicable = Satellite creates VM, installs OS and additional software)
   - Windows VM: RHEV-M
   - RHEL VM: JON
   - RHEL VM: MRG Manager
   - NFS service
10. Provision MGMT-2 node from Satellite
11. Build MGMT-1 and MGMT-2 into a RHCS cluster
12. Make cloud infrastructure management services clustered services
13. Balance clustered services (for better performance)
14. Configure RHEV-M including
    - RHEV data center(s)
    - RHEV cluster(s) within the data center(s)
15. Configure RHEV-H host

16. Configure RHEL/KVM host

This paper documents the procedure that continues this effort by demonstrating how to configure:

1. A private cloud with one of each type of hypervisor hosts:
   - RHEL / KVM
   - RHEV-H
2. A private cloud with added hypervisor hosts
3. A hybrid cloud with more VMs

6.1 Host Load Balancing

While not called out in the Red Hat Cloud Foundation, the maintainer of the cloud infrastructure should confirm that the load is balanced among available hosts. The RHEV Manager allows the administrator to specify load balancing parameters to determine at which point VMs should migrate from one host to another. To set the preferred load policy, in the *Clusters* tab, select/highlight the cluster name:

- Click the *Policy* tab in the lower half of the window
- Click the *Edit* button
- Select the *Even Distribution* option
- Set the *Maximum Service Level* slider to 85%
- Set the length of time to 2 minutes
- Click *OK*

These settings instruct the RHEV Manager to attempt to migrate VMs from any host that has sustained a CPU load of 85% or greater for longer than two minutes to another host, provided the load on the target host is below the specified threshold.

7 Amazon's Elastic Compute Cloud (EC2): Public IaaS Cloud

Amazon Elastic Compute Cloud (Amazon EC2) is a commercial web service that provides resizable compute capacity in a public cloud.

Amazon EC2's web service interface allows the user to obtain and configure capacity. It reduces the time required to obtain and boot new server instances to minutes, allowing the user to quickly scale capacity, both up and down, as their computing requirements change.

It is designed to make web-scale computing easier for developers and allows paying customers to rent computers on which to run applications. EC2 allows scalable deployment of applications by providing a web services interface through which customers can request an arbitrary number of Virtual Machines (i.e., server instances) on which they can load any software of their choice. Current users are able to create, launch, and terminate server instances on demand, hence the term "elastic". EC2 is one of several Web Services provided by Amazon.com under the term Amazon Web Services (AWS).

7.1 Amazon EC2 Core Concepts

7.1.1 Amazon Machine Image (AMI)

An Amazon Machine Image (AMI) is an encrypted file stored in Amazon Simple Storage Service (S3) or in a Amazon Elastic Block Storage (EBS) volume. It contains all the information necessary to boot instances of your software.

7.1.2 Amazon EC2 Instance

The running system (based on an AMI) is referred to as an instance, a virtual private server that can be one of several sizes. All instances based on the same AMI begin executing identically when created. For AMI backed by S3, any information on them is lost when the instances are terminated or if they fail. An EBS backed volume can be stopped and retain information for later startup or can also be cleared when terminated.

7.2 Amazon EC2 Functionality

Amazon EC2 presents a virtual computing environment, allowing the user to use web service interfaces to requisition machines for use, load them with the user's custom application environment, manage network access permissions, and run your image using as many or few systems as desired.

To use Amazon EC2, simply:

- Create an AMI containing the user specific applications, libraries, data and associated configuration settings. Or use preconfigured, templated images to get VMs up and running immediately.
- Upload the AMI into Amazon's Simple Storage Service (Amazon S3). EC2 provides

tools that make storing the AMI simple. S3 provides a safe, reliable and fast repository to store images.

- Use EC2 web service to configure security and network access.
- Start, terminate, and monitor as many instances of the user's AMI as needed, using the web service APIs.
- Pay only for the resources that are actually consumed, such as instance-hours or data transfer.

7.3 RHEL in Amazon EC2

Cloud computing changes the economics of IT by enabling the user to pay only for the capacity that is actually used. In partnership with Amazon, Red Hat offers Red Hat Enterprise Linux on the Amazon Elastic Compute Cloud (EC2). By offering a dynamically allocated server resource with the leading open source operating system, Amazon and Red Hat provide a complete hosting platform that is immediately accessible and secure. With a consistent subscription model, operational capabilities, and technology, Red Hat makes it simple to move applications between internal clouds and public clouds.

Use Red Hat Enterprise Linux on Amazon EC2 through:

Red Hat Cloud Access

Available to Red Hat Enterprise Linux Premium (24x7) subscribers, the Red Hat Cloud Access feature of a Red Hat Enterprise Linux subscription provides enterprise customers with the choice of leveraging the value of their subscription either on-premise or at Premier Red Hat Certified Clouds, including Amazon EC2.

With Red Hat Cloud Access, customers have the ability to use the advantages provided by public cloud providers, while continuing to benefit from the high level of support delivered directly by Red Hat.

Red Hat Enterprise Linux Hourly Beta

Capacity on-demand is a core benefit of the cloud. Red Hat Enterprise Linux Hourly Beta provides access to the the world's leading open source operating system on Amazon EC2 for customers looking to get started today with limited up-front costs.

Regardless of method above, the user has access to the same base AMIs and updates within the Amazon EC2 cloud. The variable elements are support levels and cost.

7.4 Configuring an Amazon Web Services (AWS) Account

This section walks the user through configuring an account on Amazon to use the EC2 Compute Cloud. To configure AWS access:

1. At http://aws.amazon.com, select *Sign Up Now*.
2. Use an existing amazon account, or create a new account for AWS.

3. After signing up, under *Next Steps for Using Amazon Web Services*, select *Amazon Simple Storage Services* (or go to http://aws.amazon.com/s3).
4. Select *Sign Up for This Web Service*.
5. Review Pricing and enter Credit Card Information.
6. Go to http://aws.amazon.com/ec2 and select *Sign Up for This Web Service*.
7. Complete the registration for the EC2 web service.
8. On the Thank You page, select *Create a New X.509 Certificate*.
9. Select *Yes* to create a new Certificate.
10. Download the Private Key and Certificate files.
11. Make note of the AWS Account ID (in ####-####-#### format) on http://aws.amazon.com/ec2 by following the *Account -> Security Credentials* links and scrolling down to the section entitled Access Credentials.

7.5 Obtaining Tools and Configuring Environment

1. Ensure that a 1.5 compatible JVM is installed and that the JAVA_HOME environment variable is set correctly.
2. The required tools can be downloaded from the Amazon EC2 Resource Center at http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351&categoryID=88 by clicking on the link to *Download the Amazon EC2 Command-Line Tools*.
3. Unzip the tools (assume files are expanded to ~/ec2/ec2-api-tools).
4. Setup environment variables using certs downloaded in the previous section. (which can be optionally placed in a script or shell profile):

```
export EC2_PRIVATE_KEY=\
  ~/ec2/pk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem
export EC2_CERT=~/ec2/cert-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem
export EC2_HOME=~/ec2/ec2-api-tools
export PATH=$PATH:$EC2_HOME/bin
```

5. To test the environment variables, verify if any Amazon publicly available images are displayed using the `ec2-describe-images` command.

```
ec2-describe-images -x self
IMAGE ami-b4a843dd   rhel-5.5/RHEL-5.5-ec2.i386.img.manifest.xml
309956199498   available   private   i386   machine aki-d4759dbd
ari-d6759dbf   instance-store
IMAGE ami-9aa843f3   rhel-5.5/RHEL-5.5-ec2.x86_64.img.manifest.xml
309956199498   available   private   x86_64 machine aki-de759db7
ari-d0759db9   instance-store
 [...]
```

These are the AMIs that were available from Amazon using the Cloud Access program at the time of this project. Any AMIs created by the user are also listed here. The “instance-store” informs the user that the AMI is backed by S3 storage. This field reports “ebs” for EBS backed images.

Note: Access to RHEL AMIs requires a subscription to RHEL on EC2, refer to http://www.redhat.com/solutions/cloud/, or section **7.3 .**

6. Using a public AMI requires a public/private keypair to establish a secure shell connection to EC2 instances. Use `ec2-add-keypair` to create the key.

```
ec2-add-keypair
```

The resulting private key text (everything between and including the "----- BEGIN RSA PRIVATE KEY-----" and "-----END RSA PRIVATE KEY-----") can be pasted into a text file (e.g., ~/ec2/myKeypair). Set the permissions on the file as follows.

```
chmod 400 ~/ec2/myKeypair.pem
```

7.6 Starting and Stopping Instances

In Step 5 above, the output listed public AMIs available, and the names are formatted as ami-########.

1. Using one of the AMIs listed, execute the following command.

```
ec2-run-instances ami-b4a843dd -k myKeypair
RESERVATION     r-b0f96ddb      875624895099    default
INSTANCE        i-db2ff8b1      ami-b4a843dd                    pending
myKeypair  0               m1.small        2010-07-15T17:23:47+0000
us-east-1c      aki-d4759dbd    ari-d6759dbf         monitoring-disabled
instance-store
```

There is now a RHEL 5.5 (or the selected AMI) image creation in progress.

2. The image can take a few minutes to start. Verify the status of the instance.

```
ec2-describe-instances
RESERVATION    r-70d04c1b   875624895099  default
INSTANCE i-63d90209  ami-b4a843dd  ec2-174-129-140-190.compute-
1.amazonaws.com  domU-12-31-39-07-BD-82.compute-1.internal  running
myKeypair   0m1.small  2010-07-20T18:39:55+0000  us-east-1baki-d4759dbd
ari-d6759dbf  monitoring-disabled  174.129.140.190  10.209.190.112
instance-store
```

The system is ready when the instance description resembles the above where the instance ID, hostname, and instance status are included with a status of *running*. Using the instance ID when checking status removes all other instances from the output.

```
ec2-describe-instances i-63d90209
```

3. Enable port 22 to allow secure shell access to the instances.

```
ec2-authorize default -p 22
```

`ec2-describe-group` can be used to verify which ports have been opened for access.

```
ec2-describe-group default
```

```
GROUP 875624895099     default    default group
PERMISSION  875624895099     default    ALLOWS     all              FROM
      USER  875624895099     GRPNAME    default
PERMISSION  875624895099     default    ALLOWS     tcp   22    22    FROM
      CIDR  0.0.0.0/0
PERMISSION  875624895099     default    ALLOWS     tcp   80    80    FROM
      CIDR  0.0.0.0/0
```

4. Log in as root to the running instance using the previously created private key.

```
ssh -i ~/ec2/myKeypair.pem \
 root@ec2-174-129-140-190.compute-1.amazonaws.com
```

5. When the instance is no longer required, use `ec2-terminate-instances` to remove it.

```
ec2-terminate-instances i-63d90209
INSTANCE   i-63d90209 shutting-down    shutting-down
```

   All data is lost when an instance is terminated.

7.7 Saving S3 Backed AMIs

Customizations to an existing AMI can be made and saved as a new AMI, which can then produce many instances that meet exact configuration requirements. Modifications must be applied to the running instance prior to bundling the instance into a user's own custom AMI.

1. Start an instance of an AMI that best fits the underlying needs.
2. Copy the key and certificate, downloaded in step 10 of Section **7.4** , to the /mnt dir on the EC2 instance. /mnt is used because it is not duplicated when the EC2 instance is bundled into an AMI.

```
scp -i ~/ec2/myKeypair.pem \
  ~/ec2/cert-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem \
  ~/ec2/pk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem \
  root@ec2-174-129-140-190.compute-1.amazonaws.com:/mnt
```

3. Copy the Amazon EC2 AMI tools from the AWS Developers Tool page at http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=251 to the running instance.
4. Log into the running instance.
5. Set Security Enhanced Linux (SELinux) to Permissive mode, currently necessary to work around existing EC2 bundling tools limitations (see Appendix **C** for details).
   a) Dynamically set SELinux to Permissive mode.

```
setenforce 0
```

   b) Prepare image to automatically relabel the security contexts on all files.

```
touch /.autorelabel
```

   c) Edit */etc/sysconfig/selinux* to persistently set the SELINUX variable to 'permissive'.
6. Install the AMI tools.

```
yum -y --nogpgcheck localinstall ec2-ami-tools.noarch.rpm
```

7. Create a directory to house any AMIs created by the user.

```
mkdir /mnt/ami
```

8. Perform any desired customizations to the running instance prior to creating the AMI. Refer to sections **9.1** and **9.2** for the specific customizations performed for this document.
9. Create an AMI from the running instance.

```
ec2-bundle-vol -k /mnt/pk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem \
  -c /mnt/cert-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem -u <AWS-Acct-ID> \
  -d /mnt/ami -r i386 -p <AMI-prefix>
```

   The -p option allows the user to specify a file name prefix for bundled AMI files. The -u option refers to the AWS Account ID, noted in step 11 of Section 7.4 , without hyphens.

10. Upload the previously created AMI to Amazon's S3 storage.

```
ec2-upload-bundle -b <S3-bucket> -a <AWS-Access-Key-ID> \
  -s <AWS-Secret-Access-Key> \
  -m /mnt/ami/<AMI-prefix>.manifest.xml
```

    The -b option refers to the name of a unique Amazon S3 bucket in which to store the bundle. If the bucket does not exist, it is created provided the bucket name is available. There is no need to have different buckets for the AMIs unless preferred.

    The access keys (-a and -s) refer to the AWS access credentials and can be obtained in the same location as the AWS Account ID in the previous step.

    The AMI-prefix changes depending on which AMI is being uploaded.

11. Back on the local system (not the EC2 instance), register the now uploaded AMI with EC2.

```
ec2-register <S3-bucket>/<AMI-prefix>.manifest.xml -n <AMI-name>
```

    The -n option allows the user to specify the name of the AMI that was provided during image creation.

    The *AMI-name* is optional and defaults to a system generated name, can be any name the user chooses to apply the the AMI, and is used for any future references of the AMI.

7.8 Elastic IP Addresses

When using Amazon EC2 instances, both public and private IP addresses are provided. However, theses addresses are not determined until the AMI has been instantiated. It would be troublesome to require changes to the configuration files of our MRG Manager, and EC2 MRG Grid execute nodes. Amazon provides a solution through an Elastic IP address. This address is known and controlled by the user until they decide to release it. This address can be associated to any EC2 instance. This is the address that is used in the configuration files.

Obtaining an address can be performed though the Amazon Web Service Management console or by issuing the following API command:

```
ec2-allocate-address
```

Note that while the allocated address remains allocated to the AWS user account, the address must be associated to the tunnel each time an instance is created. `ec2-describe-addresses` can be used to list the addresses allocated to the AWS user account.

8 Using MRG Grid

MRG Grid provides high throughput and high performance computing. Additionally, it enables enterprises to move to a utility model of computing to help achieve both higher peak computing capacity and higher IT utilization 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, utilization, 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, utilization, 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.

Condor is essentially a specialized batch system for managing compute-intensive jobs. Like most batch systems, Condor provides a queuing mechanism, scheduling policy, priority scheme, and resource classifications. Users submit their compute jobs to Condor, Condor puts the jobs in a queue, executes them, and then informs the user of the result.

The infrastructure in this document uses the clustered service mrg-vm to perform Condor's scheduling, data collecting, and job submission.

8.1 Submitting a Job to MRG Grid

Submitting a job consists of six main steps, further details can be found in the Red Hat Enterprise 1.2 Grid User Guide:

1. Prepare the job:

   Jobs must be able to run without interaction by the user, as MRG Grid runs unattended and in the background. All interactive input and output must be automated.

2. Choose a universe:

MRG Grid uses a runtime environment, called a *universe*, to determine how a job is handled as it is being processed.

3. Write a submit description file:
   The submit description file contains the details of the job submission.
4. Submit the job:
   Submit the program to MRG Grid for processing.
5. Monitor the progress of the job:
   Once a job has been submitted, MRG Grid executes the job. Job progress can be monitored using a variety of methods.
6. Finishing the job:
   When the job completes, MRG Grid provides the job exit status and other statistics.

8.2 Using MRG Grid to Start and Stop Instances

MRG Grid has the ability to create instances in Amazon EC2. This document uses this feature to spawn both the Grid nodes themselves and the node which tunnels the traffic from the EC2 Grid nodes through the corporate firewall. The life of the EC2 instances coincides with the specific jobs submitted to create the instances.

The following is the submit description file (e.g., *ec2_tunnel.sub*) for the EC2 system which forwards traffic from the EC2 MRG execute nodes through a tunnel in the corporate firewall. The keys referenced in step 10 of Section **7.4** are set to provide access. The ID for the AMI customized in Section **9.1** is specified. The *amazon_keypair_file* assigns the name of a file that MRG creates which can be used as the key for ssh to communicate with the spawned system.

```
# Note to submit an AMI as a job we need the grid universe
Universe = grid
grid_resource = amazon

# Executable in this context is just a label for the job
Executable = tunnel_instance
transfer_executable = false

# Keys provided by AWS
amazon_public_key = /home/ec2/cert-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem
amazon_private_key = /home/ec2/pk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem

# The AMI ID
amazon_ami_id = ami-9c45aef5

# The generated keypair file if needed for using ssh etc
amazon_keypair_file = /home/admin/tunnelKP
```

```
# The security group for the job
amazon_security_groups = default

queue
```

Similarly, the submit file for the EC2 based MRG execute nodes (e.g., *ec2_mrgexec.sub*) is listed below. This AMI ID corresponds to the AMI image customized in Section **9.2** . The *amazon_keypair_file* has the cluster and process included to differentiate between multiple guests. The number after the 'queue' keyword indicates the number of EC2 systems that is created which defaults to one in the previous example.

```
# Note to submit an AMI as a job we need the grid universe
Universe = grid
grid_resource = amazon

# Executable in this context is just a label for the job
Executable = mrg_exec_instance
transfer_executable = false

# Keys provided by AWS
amazon_public_key = /home/ec2/cert- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem
amazon_private_key = /home/ec2/pk- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.pem

# The AMI ID
amazon_ami_id = ami-949379fd

# The generated keypair file if needed for using ssh etc
amazon_keypair_file = /home/admin/mrgexecKP.$(cluster).$(process)

# The security group for the job
amazon_security_groups = default

queue 16
```

`condor_submit` is used with the job submit files above to start the instances.
`condor_rm` is used to stop the instances.

8.3 Configuring MRG Grid to use EC2 systems

The infrastructure configured in Volume I of this series established the MRG environment but several changes are required to include EC2 systems. Many companies have a firewall in place allowing outgoing traffic but protecting against unwanted incoming traffic. The incoming traffic from all EC2 nodes is forwarded to one system which uses a corporate approved method allowing specific traffic through the firewall. In this paper, the system acting as the tunnel is an EC2 system with an assigned EC2 elastic IP. This allows the EC2 MRG Grid execute nodes to have a static configuration.

The drawback to using an EC2 instance with an elastic IP is that several condor daemons attempt to contact this address but when the Elastic IP is not assigned, the daemons timeout attempting to contact the EC2 system. Using a system that always has a resolvable IP

address for a tunneling host eliminates these timeouts which can be seen on the MRG Manager when the daemons start and when each job begins processing.

The */var/lib/condor/condor_config.local* file is provided below. The EC2 specific additions have been highlighted. The majority of the changes are related to network traffic flow through the tunnel host including modifying the collector to accept TCP messages due to the fact that the tunneling solution does not allow UDP packets. See Appendix **B** for details on Condor tickets #1329 and #1405 which resolved issues related to this feature in MRG version 1.3. For this reason, MRG version 1.3 was used on the EC2 instances.

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

# What the relay looks like from the inside.
PRIVATE_HOST = <Internal IP Name/Address of Firewall Tunnel>

# What the relay looks like from the outside.
PUBLIC_HOST = <EC2 Elastic IP of tunnel host>
PUBLIC_PORT = 9618

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

# Setting CCB_ADDRESS for all daemons 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 all but the Collector.
CCB_ADDRESS = $(PUBLIC_HOST):$(PUBLIC_PORT)
COLLECTOR.CCB_ADDRESS =

# Avoid needing CCB within the VPN
PRIVATE_NETWORK_NAME = mrg-vm

# Set TCP_FORWARDING_HOST so CCB advertises its public
# address. Without this, it advertises its private address
# and the Starter is not 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)
HOSTALLOW_WRITE = *.cloud.lab.eng.bos.redhat.com,  $(PRIVATE_HOST)
```

```
CONDOR_HOST = $(FULL_HOSTNAME)
COLLECTOR_NAME = Grid On a Cloud
#COLLECTOR_HOST = $(CONDOR_HOST)
NEGOTIATOR_HOST = $(CONDOR_HOST)
UID_DOMAIN = cloud.lab.eng.bos.redhat.com
FILESYSTEM_DOMAIN = cloud.lab.eng.bos.redhat.com
START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
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 = mrg-vm.cloud.lab.eng.bos.redhat.com
```

The changes for the execute node are detailed in the section **9.2** where the AMI image is created.

Two helper scripts were created to assist managing the search. This first is used to create the desired number of submit files, evenly distributing the number to search for each submit. This script requires two arguments: the maximum number for the search and how many segments/jobs in which to divide the search.

```
#!/bin/bash

if [[ $# -ne 2 ]]
then
  echo "Usage - $0 MAX #jobs"
  exit -1
fi

/bin/rm perfect_seg*.sub 2>/dev/null

MAX=$1
NJOBS=$2

let start=1
let incr=MAX/NJOBS

stop=${incr}
let seg=1
```

```
while [[ $stop -le ${MAX} ]]
do
 fname="perfect_seg${seg}.sub"
 cat <<-EOF>${fname}
       Universe = vanilla
       Requirements = Arch =!= Undefined
       Executable = /usr/tmp/perfect
       Arguments = ${start} ${stop}
       Log = /home/admin/perfect/output/${fname%sub}log
       Output = /home/admin/perfect/output/${fname%sub}out
       Error = /home/admin/perfect/output/${fname%sub}err
       should_transfer_files = YES
       when_to_transfer_output = ON_EXIT
       transfer_executable = true
       QUEUE
       EOF
 let seg++
 let start=stop+1
 let stop+=incr
done
```

The second script cleans any previously submitted jobs and submits each of the currently existing submit job description files.

```
#!/bin/bash
# Removing all jobs in the queue ...
for job in `condor_q |grep perfect |awk '{print $1}'`
do
  condor_rm $job
done
sleep 5
echo Cleaning up log/out/err files ....
rm -f /home/admin/perfect/output/*.{log,out,err} 2>/dev/null
sleep 5
for submit in perfect*.sub
do
  condor_submit ${submit}
done
```

8.4 Authorizing MRG Access into EC2

The incoming network connections MRG required need to be authorized. For the MRG ports identified in this paper the following commands provide this access using the default group.

```
ec2-authorize default -p 40000-40005 -P tcp
ec2-authorize default -p 9618 -P tcp
```

9 Customized Amazon EC2 AMIs

As section **7.7** presented, an EC2 user can save customized AMIs so that instances spawned from their AMI are preconfigured to suit the needs of the user with respect to configuration settings, applications, data, and/or libraries. For the purpose of this paper, modifications were made to the two AMIs used and are detailed below. The changes are made to a running instance before the bundling of the customized AMI.

9.1 Firewall Tunnel Image

To configure a RHEL instance for use as a forwarding tunnel through a corporate firewall:

1. Configure the firewall to allow MRG collector traffic.

```
iptables -I RH-Firewall-1-INPUT -p tcp --dport 9618 -m state --state NEW
-j ACCEPT
```

2. Save the firewall configuration.

```
service iptables save
```

9.2 MRG Execute Image

To configure an AMI with preconfigured MRG Grid software and open the required firewall ports:

1. Set the firewall to allow MRG traffic.

```
iptables -I RH-Firewall-1-INPUT -p tcp -m tcp --dport 40000:40005 \
  -m state --state ESTABLISHED,NEW -j ACCEPT
```

2. Save the firewall configuration.

```
service iptables save
```

3. Acquire access to the MRG software packages. To use MRG 1.3 Beta, used for the EC2 instances in this paper, contact your Red Hat representative.
4. Install version 1.3 of the MRG Grid software (may be available in beta only).

```
yum -y groupinstall "MRG Grid"
```

5. Create a */var/lib/condor/condor_config.local.template* file and populate it with the following. Note that the highlighted area requires the user to specify the EC2 Elastic IP that is associated with the tunnel system. Modify COLLECTOR_NAME and ALLOW_WRITE according to the environment.

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

COLLECTOR_HOST = <EC2 Elastic IP address associated with tunnel node>
COLLECTOR_NAME = Grid On a Cloud
DAEMON_LIST = MASTER, STARTD
NEGOTIATOR_INTERVAL = 20
```

```
TRUST_UID_DOMAIN = TRUE
IN_HIGHPORT = 40005
IN_LOWPORT = 40000
UPDATE_COLLECTOR_WITH_TCP = True
USE_PROCD=FALSE

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

# specify the domain(s) allowed to access the instance
ALLOW_WRITE = $(FULL_HOSTNAME) *.redhat.com

START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE
```

6. Append the following to */etc/rc.local* for execution at instance startup. This sets the variable associated with the network traffic when the dynamic addressed have been assigned.

```
/bin/cp /var/lib/condor/condor_config.local.template /var/lib/condor/condor_config.local
echo TCP_FORWARDING_HOST = `curl -f http://169.254.169.254/latest/meta-data/public-ipv4`
>> /var/lib/condor/condor_config.local
echo PRIVATE_NETWORK_INTERFACE = `curl -f http://169.254.169.254/latest/meta-data/local-
ipv4`  >> /var/lib/condor/condor_config.local
service condor stop
pkill -9 condor
service condor start
```

10 The Perfect Number Search Workload

An easy to implement algorithm was selected to provide a sample workload. The C program below performs a search for perfect numbers within a range. A perfect number is defined as a positive integer that is the sum of its proper positive divisors, i.e., the sum of the positive divisors excluding the number itself.

```
#include <stdio.h>
#include <err.h>

int isperfect(long num) {
 long
   indx=2,
   sum=1,
   max=(num/2);

 while ( indx <= max ) {
   if ( num % indx == 0 ) {
     sum += indx;
     max = num / indx;
     sum += max;
     max -= 1;
     if ( sum > num ) break;
   }
   indx += 1;
 }
 if ((sum == num) && (num != 1))
   return(1);
 else
   return(0);
}

int main (int argc, char **argv) {

 long
   candidate,
   start,
   stop;

 if (argc != 3) {
   errx(-1, " Usage -   %s MAX #segments", argv[0]);
   exit(-1);
 }

 start = atol(argv[1]);
 stop = atol(argv[2]);

 for (candidate=start; candidate < stop; candidate++) {
   if (isperfect(candidate))
```

```
printf("%d\n",candidate);
  }
}
```

Example run output:

```
/usr/tmp/perfect 1 100
6
28
```

11 Expanding from Private to Hybrid Cloud

A search of the internet claims that the first five perfect numbers are:

- 6
- 28
- 496
- 8,192
- 33,550,336

This section details the procedure followed to verify the claim and confirm that no perfect numbers were missed.

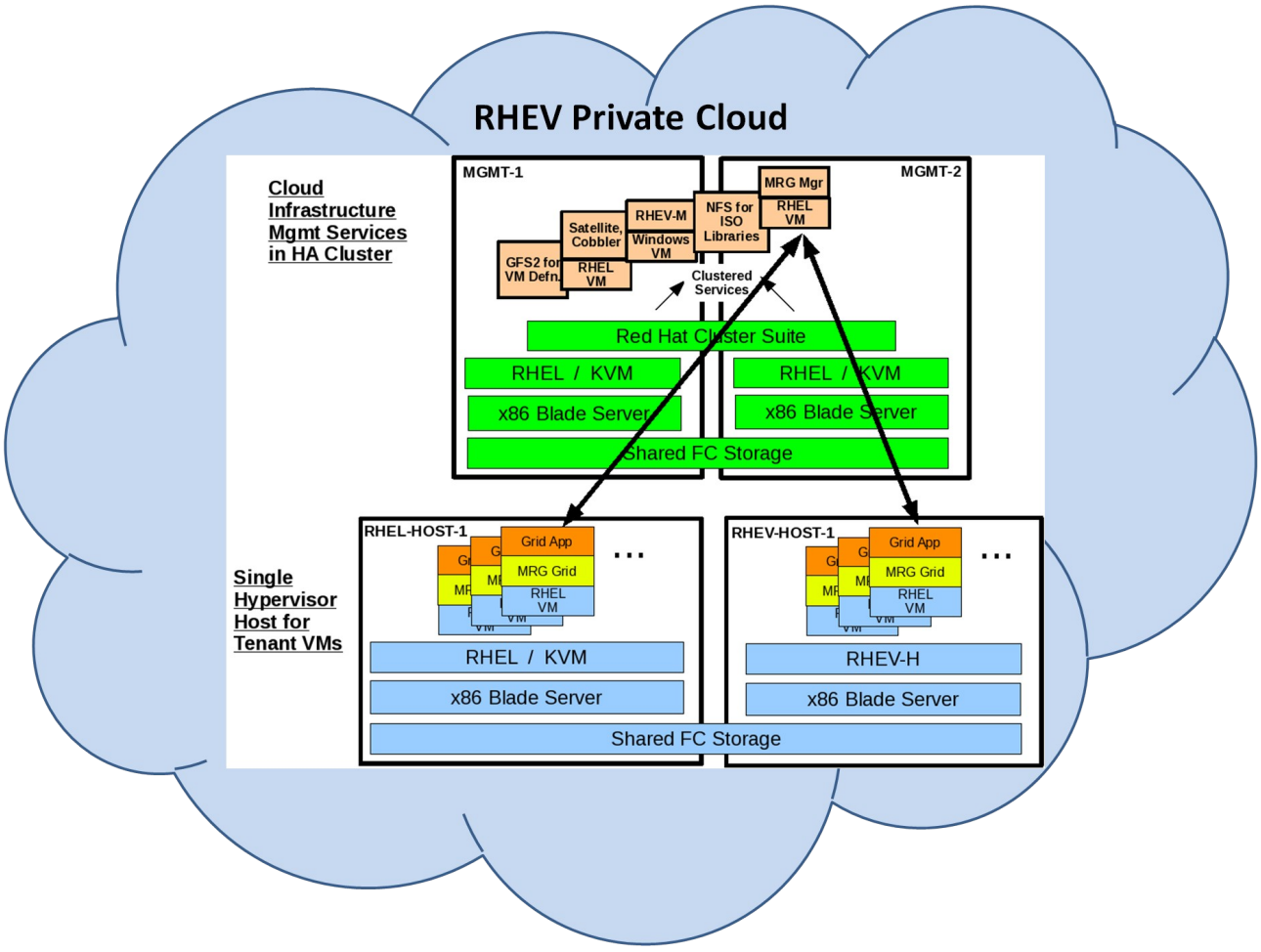


Figure 13: Single Hosts

11.1 Private Cloud Infrastructure

This section assumes that the procedures detailed in the referenced documents have been followed to create the infrastructure as it had completed in the previous reference architectures. As depicted, the pair of RHEL hosts (one RHEV-H, one RHEL5.5 w/KVM) and the MRG Manger VM is used for the initial load.

1. In RHEV-M, create the first MRG VM (mrg1) using the associated kickstart (rhel55_mrg_exec) in PXE. This VM has a single CPU with 1 GB memory and a 5GB logical drive.
2. Prepare a template from this VM. As documented in the **Red Hat Cloud Foundations: Private IaaS Clouds** document, make the necessary modifications to the */var/sysconfig/network* and */etc/rc.d/rc.local* files.
3. ssh into the MRG Manager clustered VM service (mrg-vm).
4. Assume the identity of the admin user and change to the workload directory.

```
su – admin
cd perfect/
```

5. Verify no jobs are currently executing.

```
condor_q
condor_status
```

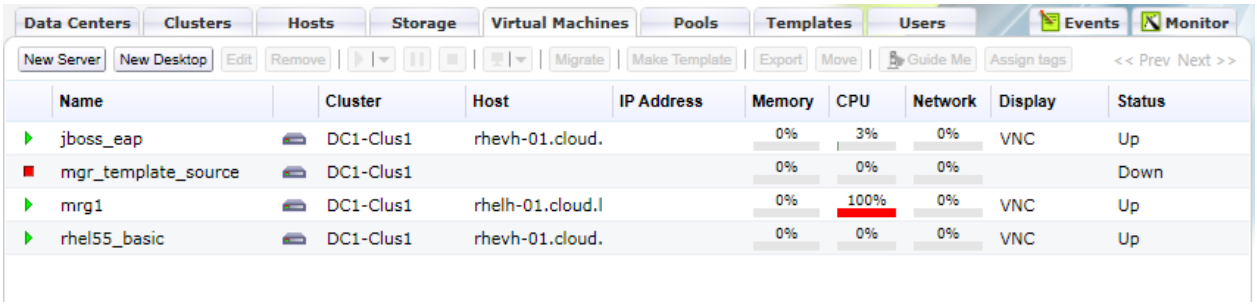
6. Create the jobs that execute the perfect number workload passing parameters that determine the upper limit of numbers to search (34,000,000) and the number of individual execute jobs (680).

```
./mk_jobs.sh 34000000 680
```

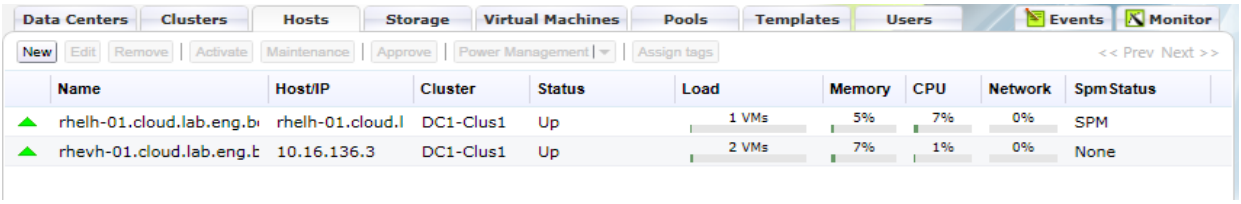
7. Start the jobs.

```
./submit_jobs.sh
```

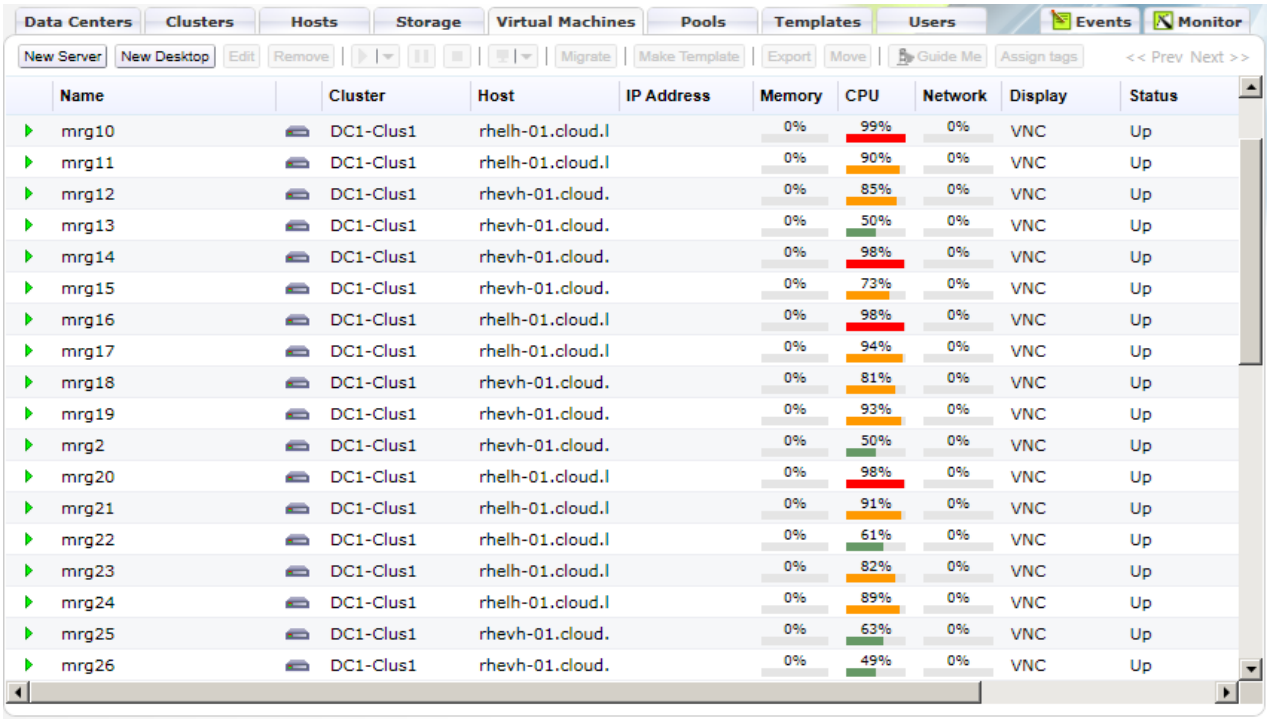
8. Use `condor_status` to monitor the state of the execute node.
9. Note the CPU load on the MRG VM (mrg1).



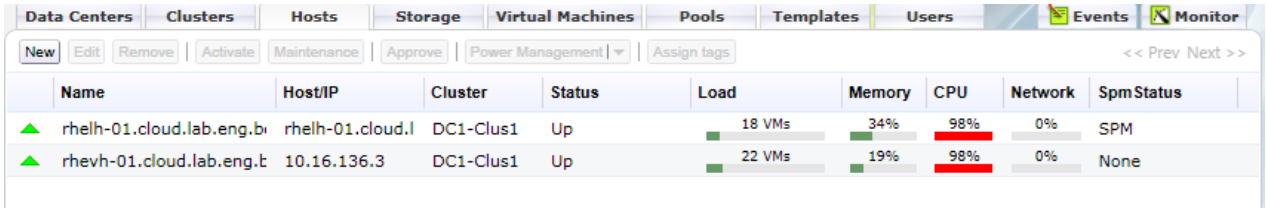
10. Observe the minimal load applied to the corresponding host, rhelh-01 in this case.



11. Add more VMs to total 40 in order to utilize the host compute capacity. This actually oversubscribes the CPU capacity of the two hosts and can be accomplished using the PowerShell script *add-vms.ps1* to create the 39 additional MRG VMs using the template created in the previous step.
    - On the RHEV Manager select from the Start menu: *All Programs -> Red Hat -> RHEV Manager -> RHEV Manager Scripting Library*
        a) In the power shell window, log in with a superuser account.
        ```
        Login-User -user admin -p <password> -domain ${env:computername}
        ```
        b) Change directories to the saved directory.
        ```
        cd C:\saved
        ```
        c) Call the script to asynchronously create the desired VMs.
        ```
        .\add-vms.ps1 -tempName <mrg_template> -baseName mrg -num 39
        ```
        d) As the VMs finish creating, the operator can select all of the desired VMs and press *Run* to start.
12. Use '`condor_status`', '`condor_q`' and '`condor_q -run`' to monitor the load.
13. Observe the VM load levels after the job has spread across the other MRG VMs. Note that by oversubscribing the host CPU capacity, the VMs can not be allocated 100% of their assigned capacity.



14. The additional load on the hosts is reflected accordingly.



11.2 Dynamic Addition of Hosts

Even with the MRG execute nodes added in the previous section, time to completion could still be significant enough to warrant more MRG execute nodes but the hosts are already CPU bound. By introducing additional hosts to the cloud infrastructure more MRG execute nodes can be created to utilize the extra compute cycles, as depicted in Figure **14**.

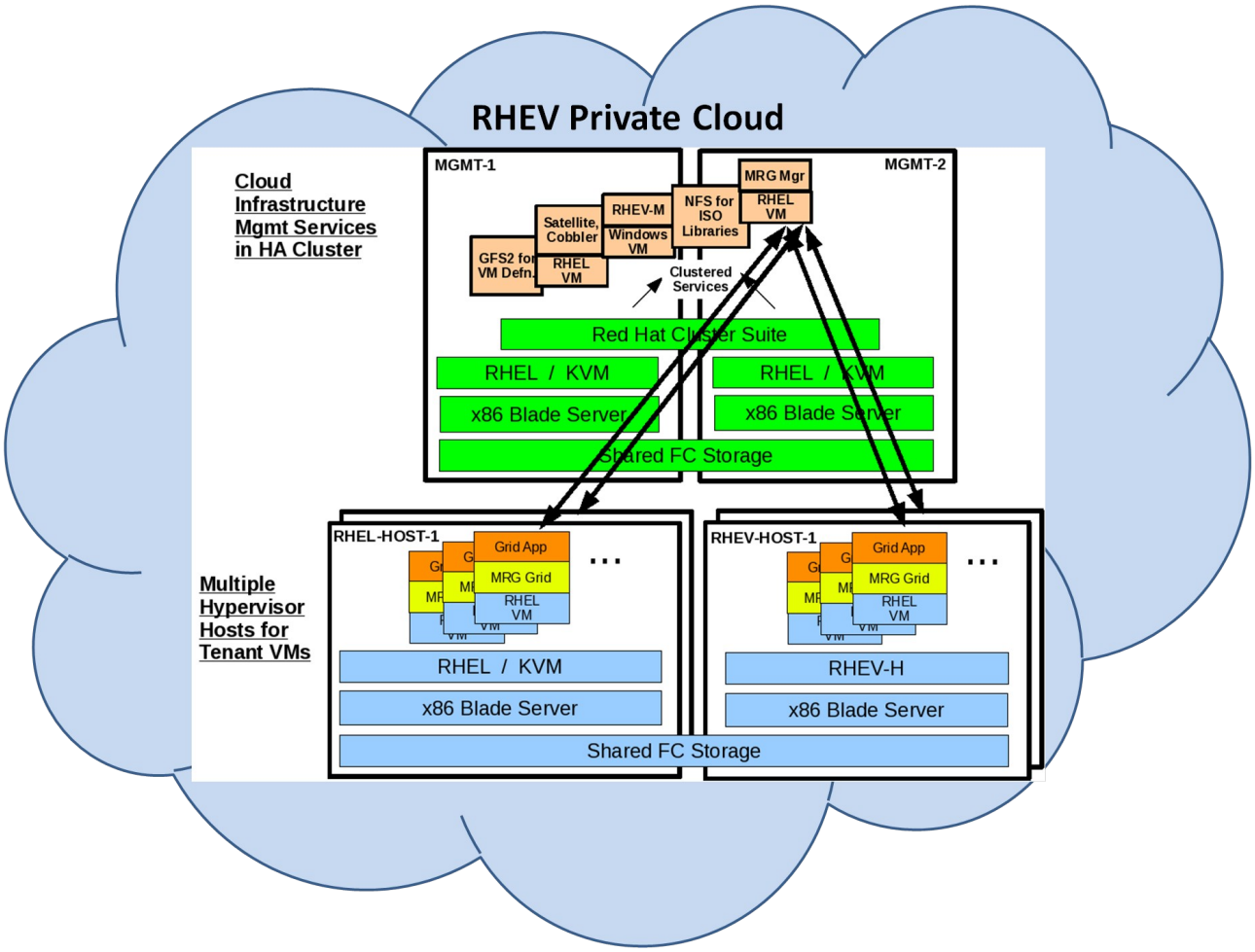


Figure 14: Multiple Hosts

In this example, two additional server blades were allocated as a RHEV-H host and a RHEL/KVM host and the appropriate HP Virtual Connect profiles have been associated with those blades.

1. Use *instRHELH.sh* and *instRHEVH.sh* to create a new host of each type.
   - The *instRHELH.sh* script requires a single passed parameter, the server blade profile name assigned in the Virtual Connect interface. The script:
     - creates the cobbler system entry
     - presents storage to the host
     - adds the new NFS export stanza to the cluster configuration file
     - sets the boot order to boot PXE first
     - registers the host with satellite after install
     - sets the boot order to boot PXE last

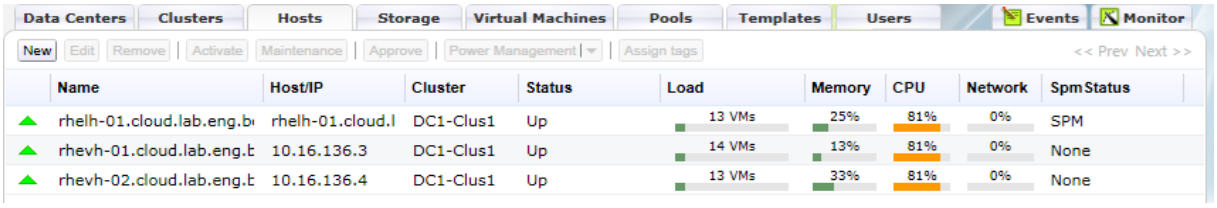
     and requires the user to add the newly created host in RHEV-M.

     ```
     ./instRHELH.sh rhelh-02
     ```

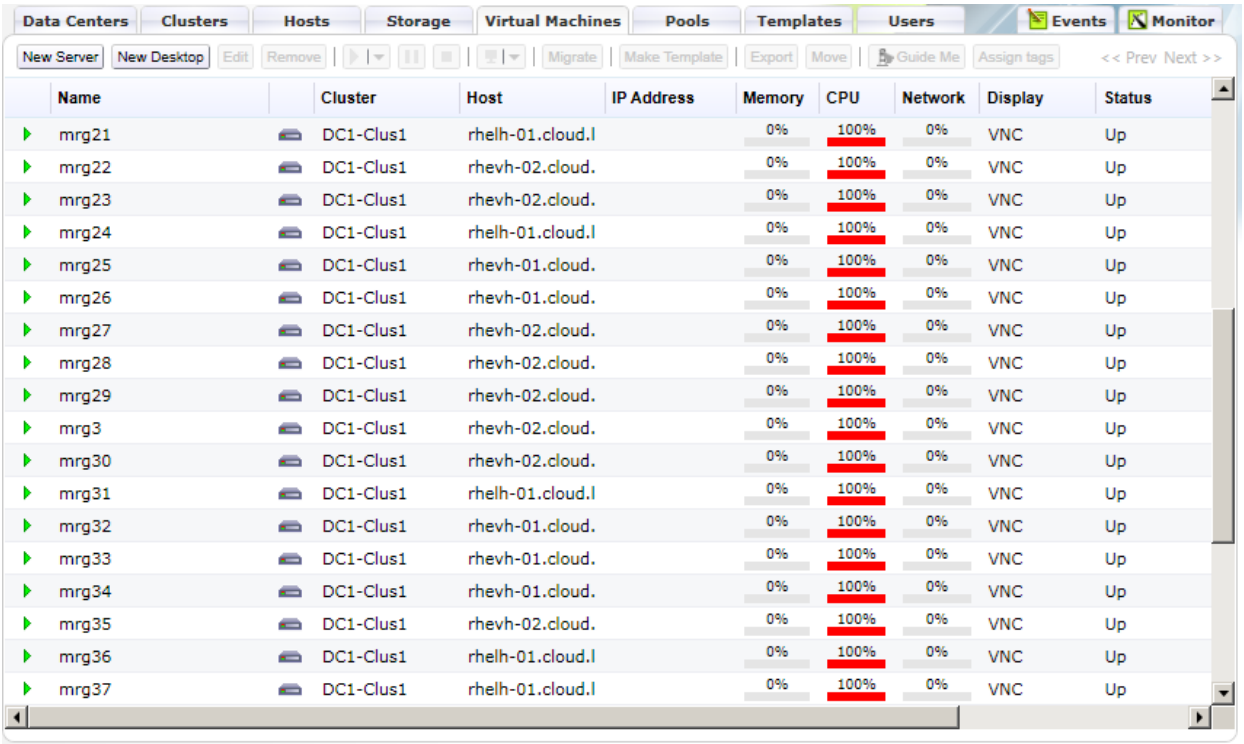
     Note: Because the kernel used for installation exhibits the issue described in BZ 602402 (which can cause network problems), ensure that no other blades in the chassis are experiencing network failures before proceeding.
   - The *instRHEVH.sh* script installs and prepare a system for use as a RHEV host. It requires a single passed parameter, the server blade profile name assigned in the HP Virtual Connect interface, and requires the user to approve the new host in RHEV-M.

     ```
     ./instRHEVH.sh rhevh-02
     ```

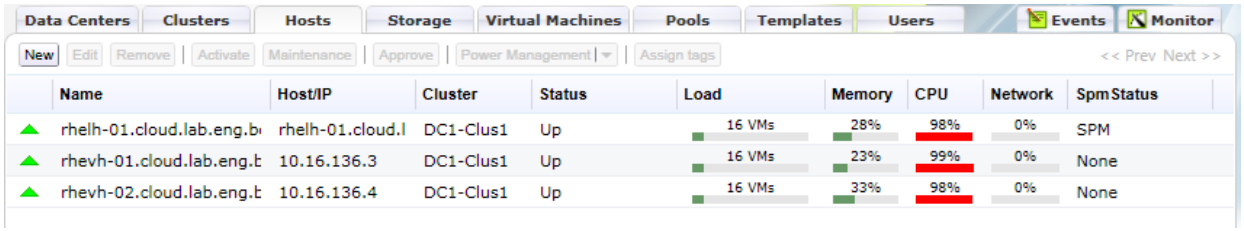
   The host install scripts are located in Appendix **A**.
2. In RHEV-M, approve the newly created RHEV-H host into the appropriate cluster. Potentially, BZ 624027 may be observed which could cause the current SPM host to reboot. Start any VMs that are not already running. BZ 620816 was entered for VMs not migrating according to the cluster's distribution policy for a new host. Performing a manual migration engaged the policy for the new host. Additional migrations were performed to completely balance the number of active VMs on each host.



3. With the hosts no longer over subscribed, the VMs return to utilizing 100% of capacity.



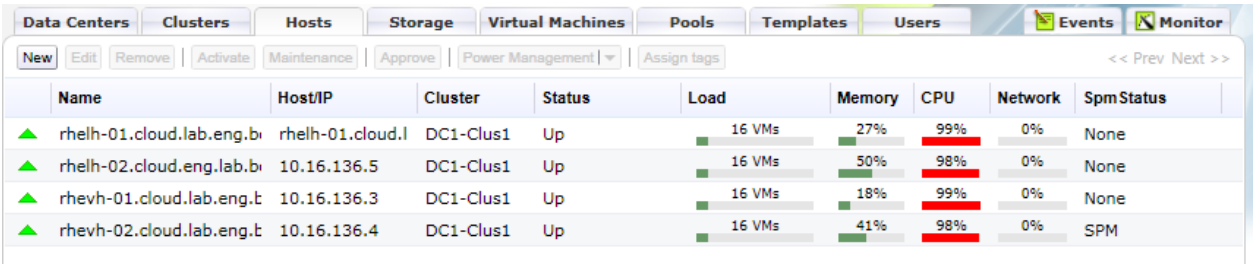
4. Adding eight more VMs brought all three hosts to capacity.



5. In RHEV-M, add the newly created RHEL host. Potentially, the same issues mentioned in step 2 could occur. If so, verify all MRG VMs are booted and distributed.
6. With a new host, there is capacity for 16 more MRG Grid execute node VMs. The "-run" option to the add-vms.ps1 script boots the VM after it is added.

```
./add-vms.ps1 -tempName <mrg_template> -baseName mrg -num 16 -run
```

7. Again, with a new host the distribution policy may not take effect. If so, then manually migrate one VM. More hosts should automatically migrate until the new host is loaded over the set maximum (85%). Manually migrate any remaining VMs that are not balanced.



8. Use '`condor_status`', '`condor_q`' and '`condor_q -run`' to monitor the load.

11.3 Adding Hybrid VMs

With completion time still farther off than desired and all available hosts have been utilized, public cloud resources can be added to the resource pool. This is depicted in Figure **15**.

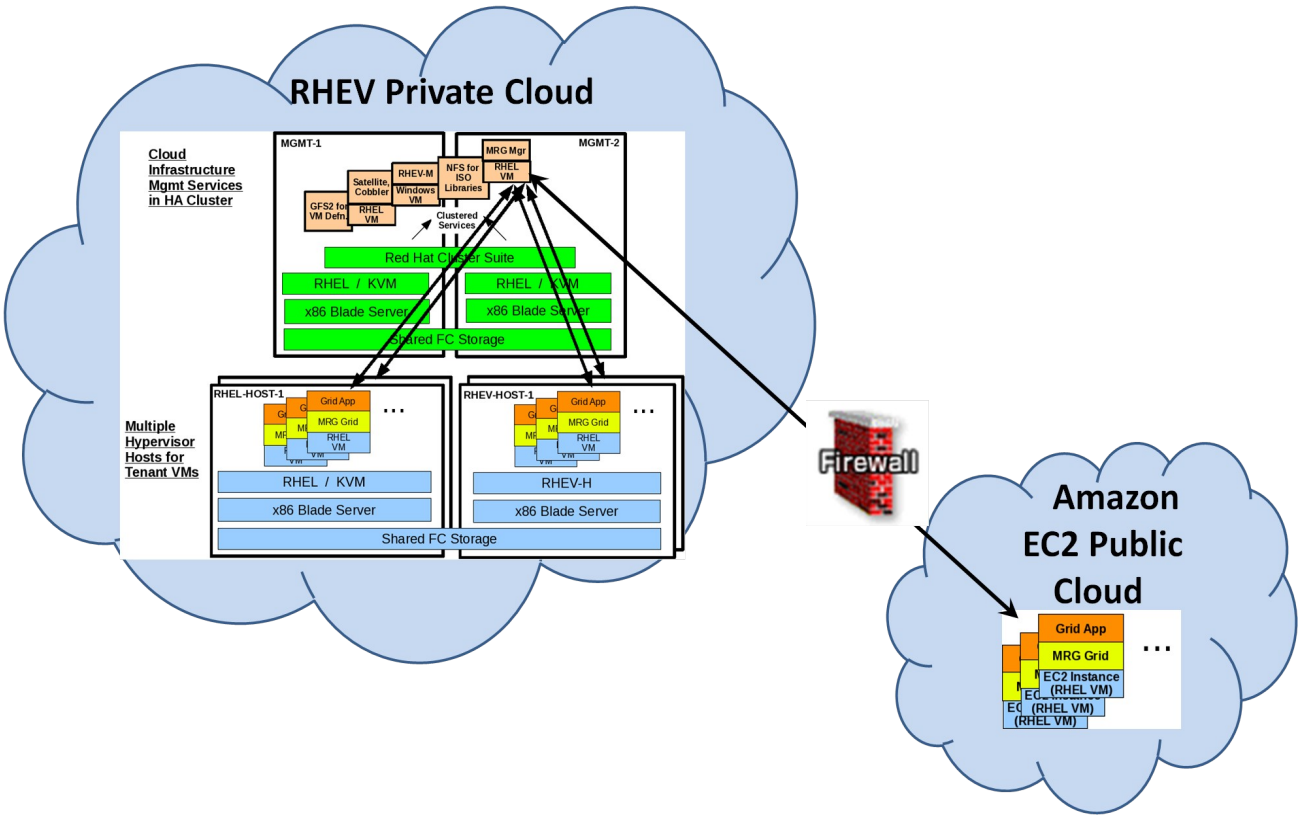


Figure15: Additional Tenant VMs

1. Start the EC2 instance that serves as the tunnel forwarding host.

```
condor_submit ec2_tunnel.sub
Submitting job(s).
1 job(s) submitted to cluster 4471.
```

2. Associate the elastic IP address to the tunnel instance. The instance id can be obtained from condor.

```
condor_q -l 4471 | grep GridJobId
GridJobId = "amazon SSH_mrg-vm.cloud.lab.eng.bos.redhat.com:9618_mrg-
vm.cloud.lab.eng.bos.redhat.com#4471.0#1281985603 i-cbf068a1"
```

Once the `ec2-associate-address` command has been issued, the tunnel instance is no longer accessible via its original IP address. `ec2-describe-instances` indicates when the elastic IP has been applied and the VM hostname changes accordingly.

```
ec2-describe-addresses
ADDRESS    184.72.227.171
ec2-associate-address -i i-cbf068a1 184.72.227.171
ADDRESS    184.72.227.171   i-cbf068a1
ec2-describe-instances i-cbf068a1
RESERVATION r-b4a5eadf 778562456303     default
INSTANCE   i-cbf068a1 ami-9c45aef5     ec2-184-72-227-171.compute-
1.amazonaws.com  domU-12-31-39-06-31-66.compute-1.internal    running
     SSH_mrg-vm.cloud.lab.eng.bos.redhat.com:9618_mrg-
vm.cloud.lab.eng.bos.redhat.com#4471.0#1281985603  0           m1.small
     2010-08-16T19:06:55+0000    us-east-1d aki-d4759dbd     ari-
d6759dbf         monitoring-disabled    184.72.227.171   10.208.54.148
          instance-store
```

3. Log into tunnel instance. Until BZ 621902 is addressed, the permissions on the private key file need to be modified.

```
chmod 400 /home/admin/tunnelKP
ssh -i ~/tunnelKP root@ec2-184-72-227-171.compute-1.amazonaws.com
```

4. Open a tunnel through the firewall. For this example, ssh is used to establish a forwarding port.

```
ssh -L 0.0.0.0:9618:mrg-vm.cloud.lab.eng.bos.redhat.com:9618
<username>@<VPNnode.domain.com>
```

5. With the tunnel open, EC2 execute node can be spawned to join in the workload.

```
condor_submit ec2_mrgexec.sub
Submitting job(s).............
16 job(s) submitted to cluster 5152.
```

6. Use '`condor_status`', '`condor_q`' and '`condor_q -run`' to monitor the load. Note the additional jobs that correspond to EC2 instances.

   Display the status of the EC2 instances, the 'R' in the 'ST' column indicates a running status.

```
condor_q 4471 5152

-- Submitter: mrg-vm.cloud.lab.eng.bos.redhat.com : <10.16.136.50:9744> :
mrg-vm.cloud.lab.eng.bos.redhat.com
 ID      OWNER           SUBMITTED     RUN_TIME ST PRI SIZE CMD
4471.0   admin          8/16 15:06   0+01:12:57 R  0   0.0
tunnel_instance
5152.0   admin          8/16 16:17   0+00:01:39 R  0   0.0
mrg_exec_instance
5152.1   admin          8/16 16:17   0+00:01:05 R  0   0.0
mrg_exec_instance
5152.2   admin          8/16 16:17   0+00:01:22 R  0   0.0
mrg_exec_instance
```

```
5152.3   admin          8/16 16:17   0+00:00:48 R  0   0.0
mrg_exec_instance
5152.4   admin          8/16 16:17   0+00:01:22 R  0   0.0
mrg_exec_instance
5152.5   admin          8/16 16:17   0+00:01:21 R  0   0.0
mrg_exec_instance
5152.6   admin          8/16 16:17   0+00:01:22 R  0   0.0
mrg_exec_instance
5152.7   admin          8/16 16:17   0+00:00:47 R  0   0.0
mrg_exec_instance
5152.8   admin          8/16 16:17   0+00:01:05 R  0   0.0
mrg_exec_instance
5152.9   admin          8/16 16:17   0+00:01:04 R  0   0.0
mrg_exec_instance
5152.10  admin          8/16 16:17   0+00:01:05 R  0   0.0
mrg_exec_instance
5152.11  admin          8/16 16:17   0+00:01:21 R  0   0.0
mrg_exec_instance
5152.12  admin          8/16 16:17   0+00:00:48 R  0   0.0
mrg_exec_instance
5152.13  admin          8/16 16:17   0+00:01:22 R  0   0.0
mrg_exec_instance
5152.14  admin          8/16 16:17   0+00:01:22 R  0   0.0
mrg_exec_instance
5152.15  admin          8/16 16:17   0+00:01:04 R  0   0.0
mrg_exec_instance
```

The `condor_status` shows 80 Claimed workers, all of which are Busy.

```
condor_status

Name               OpSys      Arch   State     Activity LoadAv Mem
ActvtyTime

domU-12-31-39-00-B LINUX      INTEL  Claimed   Busy     1.430  1700
0+00:00:02
domU-12-31-39-00-B LINUX      INTEL  Claimed   Busy     1.020  1700
0+00:00:02
domU-12-31-39-00-B LINUX      INTEL  Claimed   Busy     0.880  1700
0+00:00:04
 [...]
cloud-143-231.clou LINUX      X86_64 Claimed   Busy     0.990  2010
0+00:02:36
cloud-143-232.clou LINUX      X86_64 Claimed   Busy     0.990  2010
0+00:03:04
                     Total Owner Claimed Unclaimed Matched Preempting
Backfill
         INTEL/LINUX    16     0      16         0       0          0
0
        X86_64/LINUX    64     0      64         0       0          0
0
               Total    80     0      80         0       0          0
0
```

The '-run' option of `condor_q` is used to show the EC2 instances status and that the load is being executed on all (ec2 & local) nodes

```
condor_q -run

-- Submitter: mrg-vm.cloud.lab.eng.bos.redhat.com : <10.16.136.50:9744> :
mrg-vm.cloud.lab.eng.bos.redhat.com
 ID      OWNER            SUBMITTED     RUN_TIME HOST(S)
4471.0   admin           8/16 15:06   0+01:15:07 amazon
4672.0   admin           8/16 15:57   0+00:06:15 cloud-139-
236.cloud.lab.eng.bos.redhat.com
4673.0   admin           8/16 15:57   0+00:06:14 cloud-143-
224.cloud.lab.eng.bos.redhat.com
 [...]
4754.0   admin           8/16 15:57   0+00:01:29 domU-12-31-39-00-B0-
B3.compute-1.internal
4755.0   admin           8/16 15:57   0+00:01:09 domU-12-31-39-14-21-
32.compute-1.internal
4756.0   admin           8/16 15:57   0+00:01:09 domU-12-31-39-00-B1-
05.compute-1.internal
4757.0   admin           8/16 15:57   0+00:00:28 cloud-142-
229.cloud.lab.eng.bos.redhat.com
4758.0   admin           8/16 15:57   0+00:00:13 cloud-143-
227.cloud.lab.eng.bos.redhat.com
5152.0   admin           8/16 16:17   0+00:03:49 amazon
5152.1   admin           8/16 16:17   0+00:03:15 amazon
5152.2   admin           8/16 16:17   0+00:03:32 amazon
5152.3   admin           8/16 16:17   0+00:02:58 amazon
5152.4   admin           8/16 16:17   0+00:03:32 amazon
5152.5   admin           8/16 16:17   0+00:03:31 amazon
5152.6   admin           8/16 16:17   0+00:03:32 amazon
5152.7   admin           8/16 16:17   0+00:02:57 amazon
5152.8   admin           8/16 16:17   0+00:03:15 amazon
5152.9   admin           8/16 16:17   0+00:03:14 amazon
5152.10  admin           8/16 16:17   0+00:03:15 amazon
5152.11  admin           8/16 16:17   0+00:03:31 amazon
5152.12  admin           8/16 16:17   0+00:02:58 amazon
5152.13  admin           8/16 16:17   0+00:03:32 amazon
5152.14  admin           8/16 16:17   0+00:03:32 amazon
5152.15  admin           8/16 16:17   0+00:03:14 amazon
```

The full output of `condor_q` lists all the jobs, both running and waiting. Note the total of 97 running jobs, one (tunnel EC2 instance) + 16 (mrgexec EC2 instances) + 64 (load on local MRG Grid nodes) + 16 (load on EC2 Grid instances).

```
condor_q

-- Submitter: mrg-vm.cloud.lab.eng.bos.redhat.com : <10.16.136.50:9744> :
mrg-vm.cloud.lab.eng.bos.redhat.com
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
4471.0   admin           8/16 15:06   0+01:15:30 R  0   0.0
```

```
tunnel_instance
4677.0   admin           8/16 15:57   0+00:06:25 R  0   0.0  perfect
14250001 1
4678.0   admin           8/16 15:57   0+00:06:21 R  0   0.0  perfect
14300001 1
4679.0   admin           8/16 15:57   0+00:06:19 R  0   0.0  perfect
14350001 1
 [...]
5152.14  admin           8/16 16:17   0+00:03:55 R  0   0.0
mrg_exec_instance
5152.15  admin           8/16 16:17   0+00:03:37 R  0   0.0
mrg_exec_instance

485 jobs; 388 idle, 97 running, 0
```

7. The results can be displayed easily.

```
sort -n output/*.out
6
28
496
8128
33550336
```

8. If no further jobs require the EC2 instances, shut down the instance by removing the jobs that started both the grid and tunnel instances.

```
condor_rm 4471.0 5152
Cluster 5152 has been marked for removal.
Job 4471.0 marked for removal
```

12 References

1. The NIST Definition of Cloud Computing
   Version 15, 07-October-2009
   http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc
2. Red Hat Cloud Foundations: Private IaaS Clouds
   http://www.redhat.com/rhel/resource_center/reference_architecture.html
3. Red Hat Cloud Foundations: Private IaaS Clouds – Automating Deployment
   http://www.redhat.com/rhel/resource_center/reference_architecture.html
4. Condor Version 7.4.2 Manual
   http://www.cs.wisc.edu/condor/manual/v7.4

Appendix A: Host Creation Scripts

These scripts were used to dynamically add RHEL / KVM and RHEV hosts.

instRHELH.sh

```
#!/bin/bash
# This script installs and prepare a system to be a RHEL host

# 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

# The blade profile must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <HP Virtual Connect profile name>'
  exit -1
else
  pname=$1
fi

vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} > /dev/null
2>/dev/null
if [[ $? -ne 0 ]]
then
  echo "HP Virtual Connect profile $pname not found!"
  exit -2
fi

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} | awk
'{print $3}'`

iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} | grep "IP
Address" | awk '{print $3}'`

rawMAC=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile ${pname} | grep
public | awk '{print $5}'`

#implement a semaphore to verify name and IP are unique
while [[ -e /tmp/AvailNameIP ]] ; do sleep 1; done
touch /tmp/AvailNameIP
```

```
IPname=`/root/resources/GetAvailRhelh.sh | awk '{print $1}'`

IPnum=`/root/resources/GetAvailRhelh.sh | awk '{print $2}'`

# Create cobbler system entry
echo -e "\nCreating cobbler system entry ...\n"
cobbler system add --name=${IPname} --profile=${RHELH_PROFILE} --mac=${rawMAC//-/:} --ip=$
{IPnum} --hostname=${IPname} --dns-name=${IPname} --kopts="console=ttyS0,115200 nostorage"
cobbler sync

#Remove semaphore
/bin/rm /tmp/AvailNameIP

# Present storage to hosts
echo -e "\nPresenting storage to host ${pname} ...\n"
/root/resources/prep_stor_host.sh ${pname}

# Update cluster configuration for NFS presentation

# create a semaphore for unique client names
while [[ -e /tmp/UpdateNFSClient ]] ; do sleep 1; done
touch /tmp/UpdateNFSClient

#Get next available client name
nfsClient=`/root/resources/GetAvailNFSClient.sh`

# add the export to the cluster configuration
addnfsexport -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient} ${IPname} rw 1

# release semaphore
/bin/rm /tmp/UpdateNFSClient

# Get the count of systems registered with this name (should be 0)
initReg=`/root/resources/listRegSystems_infra.py | grep -c ${IPname}`
echo -e "\nNumber of systems registered with this name: ${initReg} ...\n"

# Set to boot PXE first
echo -e "\nChanging system boot order to boot network (PXE) first ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=1 | grep status=0` ]]; do sleep 10; done

# Reset node (power on node in case node was off)
echo -e "\nResetting server blade (power on in case blade was off) ...\n"
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power reset
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power on

# Disable FC
echo -e "\nDisabling FC connections during install ...\n"
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 1
```

```
speed=disabled
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 2
speed=disabled

# Wait for system to register with satellite indicating installation completion
echo -e "\nWaiting for system to register with satellite ...\n"
while [[ $initReg -ge `/root/resources/listRegSystems_infra.py | grep -c ${IPname}` ]]; do sleep 15; done
echo -e "\nSatellite registration complete ...\n"

# Set to boot PXE last
echo -e "\nChanging system boot order to boot network last ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=5 | grep status=0` ]]; do sleep 2; done

# Enable FC
echo -e "\nEnabling FC connections ...\n"
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 1 speed=auto
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 2 speed=auto
```

instRHEVH.sh

```
#!/bin/bash
#
# This script installs and prepare a system for use as a RHEV host

# 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

# The blade profile must be passed
if [[ $# -ne 1 ]]
then
  echo 'Usage - $0 <HP Virtual Connect profile name>'
  exit -1
else
  pname=$1
fi

# Implement a semaphore to verify that name and IP are unique
while [[ -e /tmp/AvailNameIP ]] ; do sleep 1; done
touch /tmp/AvailNameIP
```

```
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} > /dev/null
2>/dev/null
if [[ $? -ne 0 ]]
then
  echo "HP Virtual Connect profile $pname not found!"
  exit -2
fi

nblade=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show server | grep ${pname} | awk
'{print $3}'`
iloIP=`oacommand --oaurl //${LOGIN}:${OA_PW}@${OA_IP} show server info ${nblade} | grep "IP
Address" | awk '{print $3}'`
rawMAC=`vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} show profile ${pname} | grep
public | awk '{print $5}'`
IPname=`/root/resources/GetAvailRhevh.sh | awk '{print $1}'`
IPnum=`/root/resources/GetAvailRhevh.sh | awk '{print $2}'`

# Delete any previously defined cobbler system entry
if [[ `cobbler system list | grep ${IPname}` ]]
then
  cobbler system delete --name=${IPname}
fi

# Create cobbler system entry
echo -e "\nCreating cobbler system entry ...\n"
cobbler system add --name=${IPname} --profile=${RHEVH_PROFILE} --mac=${rawMAC//-/:} --ip=$
{IPnum} --hostname=${IPname} --dns-name=${IPname} --kopts="console=ttyS0,115200 nostorage"
cobbler sync

# Remove semaphore
/bin/rm /tmp/AvailNameIP

# Present storage to host
echo -e "\nPresenting storage to host ${pname} ...\n"
/root/resources/prep_stor_host.sh ${pname}

# Update cluster configuration for NFS presentation

# create a semaphore for unique client names
while [[ -e /tmp/UpdateNFSClient ]] ; do sleep 1; done
touch /tmp/UpdateNFSClient

#Get next available client name
nfsClient=`/root/resources/GetAvailNFSClient.sh`

# add the export to the cluster configuration
addnfsexport -H ${MGMT1_IP} rhev-nfs-fs ${nfsClient} ${IPname} rw 1

# release semaphore
/bin/rm /tmp/UpdateNFSClient
```

```
# Get the count of systems registered with this name (should be 0)
initReg=`/root/resources/listRegSystems_infra.py | grep -c ${IPname}`
echo -e "\nNumber of systems registered with this name: ${initReg} ...\n"

# Change system boot order to boot network (PXE) first
echo -e "\nChanging system boot order to boot network (PXE) first ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=1 | grep status=0` ]]; do sleep 10; done

# Reset server blade (power on in case blade was off)
echo -e "\nResetting server blade (power on in case blade was off) ...\n"
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power reset
ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} power on

# Disable fc connections
echo -e "\nDisabling FC connections during install ...\n"
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 1
speed=disabled
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 2
speed=disabled

# Wait for system to register with satellite indicating installation completion
echo -e "\nWaiting for system to register with satellite ...\n"
while [[ $initReg -ge `/root/resources/listRegSystems_infra.py | grep -c ${IPname}` ]]; do sleep 5; done
echo -e "\nSatellite registration complete ...\n"

# Change system boot order to boot network last
echo -e "\nChanging system boot order to boot network last ...\n"
while [[ ! `ilocommand -i //${LOGIN}:${ILO_PW}@${iloIP} set /system1/bootconfig1/bootsource5
bootorder=5 | grep status=0` ]]; do sleep 2; done

# Enable fc connections
echo -e "\nEnabling FC connections ...\n"
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 1 speed=auto
vcmcommand --vcmurl //${LOGIN}:${VCM_PW}@${VCM_IP} set fc-connection ${pname} 2 speed=auto
```

Appendix B: Condor Tickets

The Condor problem reports below were open(ed) at the time of this exercise.

1. Ticket #1329: Private network settings ignored in file transfer
   https://condor-wiki.cs.wisc.edu/index.cgi/tktview?tn=1329
2. Ticket #1405: ConvertDefaultIPToSocketIP can break TCP_FORWARDING_HOST
   https://condor-wiki.cs.wisc.edu/index.cgi/tktview?tn=1405
3. Ticket #1594: EC2 job adds missing instance label
   https://condor-wiki.cs.wisc.edu/index.cgi/tktview?tn=1594

Appendix C: Bugzillas

The following Red Hat bugzilla reports were open(ed) issues at the time of this exercise.

1. BZ 602402 - bnx2x panic dumps with multiple interfaces enabled
   https://bugzilla.redhat.com/show_bug.cgi?id=602402
2. BZ 613123 - EC2: Custom RHEL 5.5 AMIs do not boot due to SELinux permissions issue - rsync involved in image creation
   https://bugzilla.redhat.com/show_bug.cgi?id=613123
3. BZ 615934 - VDSM should notify once friendly names are used on multipath.conf
   https://bugzilla.redhat.com/show_bug.cgi?id=615934
4. BZ 620816 - Adding host to loaded config reboots existing SPM host
   https://bugzilla.redhat.com/show_bug.cgi?id=620816
5. BZ 620828 - Explanation required needed when VM creation fails
   https://bugzilla.redhat.com/show_bug.cgi?id=620828
6. BZ 621899 - RFE: Specify a Elastic IP to associate with Amazon Grid instance
   https://bugzilla.redhat.com/show_bug.cgi?id=621899
7. BZ 621902 - MRG: Permissions not set correctly on key pair file
   https://bugzilla.redhat.com/show_bug.cgi?id=621902
8. BZ 623767 - Unable to migrate VMs with one host in maintenance mode
   https://bugzilla.redhat.com/show_bug.cgi?id=623767
9. BZ 624027 - Load balance policy fails to engage for new hosts
   https://bugzilla.redhat.com/show_bug.cgi?id=624027