



Red Hat Reference Architecture Series

OpenShift Enterprise

Deploying and Managing a Private PaaS with OpenShift Enterprise

Scott Collier, Sr. Principal Software Engineer

RHCA

Steve Reichard, Sr. Principal Software Engineer

RHCE

Version 1.0

December 2012





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

Linux is a registered trademark of Linus Torvalds. Red Hat, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group.

Intel and the Intel logo are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

© 2012 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

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

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

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

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

Send feedback to refarch-feedback@redhat.com



Table of Contents

1 Executive Summary.....	1
2 Components Overview.....	3
2.1 OpenShift Enterprise.....	3
2.1.1 Broker.....	3
2.1.2 Node.....	3
2.2 Load Balancer Add-On for Red Hat Enterprise Linux 6.....	4
2.3 Red Hat Enterprise Virtualization.....	4
2.4 Microsoft Windows.....	4
2.4.1 Windows Server 2008 R2	4
2.4.2 Active Directory Domain Services (AD DS)	4
2.5 BIND.....	5
2.6 Red Hat Enterprise Linux.....	5
3 Reference Architecture Configuration.....	7
3.1 Environment.....	7
3.2 Virtual Machines.....	9
3.3 Software Configuration.....	10
3.3.1 Configuration Files.....	10
3.3.2 Virtual Machine Sizing.....	10
3.3.3 Virtual Machine IP Addresses.....	10
3.3.4 Required Channels.....	11
3.3.5 Software Revisions.....	13
3.3.6 Security.....	14
3.4 Hardware Configuration.....	15
3.4.1 Servers.....	15
3.4.2 Storage.....	15
4 Deploying the Infrastructure.....	16
4.1 Deploy Red Hat Enterprise Virtualization.....	16
4.2 Deploy BIND.....	16
4.3 Deploy Broker Support Nodes.....	21
4.3.1 ActiveMQ.....	21
4.3.2 MongoDB.....	28
4.4 Deploy Brokers.....	33



4.4.1 Configure LDAP Authentication.....	40
4.4.2 Deploy OpenShift Enterprise Console.....	41
4.5 Deploy Nodes.....	44
4.5.1 Handling SSH Host Keys.....	50
4.6 Deploy Red Hat Load Balancer.....	51
5 Operational Management.....	67
5.1 Client Tools Configuration.....	67
5.1.1 Installation.....	67
5.2 Infrastructure Operation	70
5.2.1 Confirm Hosts.....	70
5.2.2 Log Files.....	74
5.3 Applications	74
5.3.1 Deploy PHP Application via Command Line.....	74
5.3.2 Deploy PHP Application via OpenShift Console.....	79
5.3.3 Deploy JBoss Application with JBoss Developer Studio 5.....	83
5.3.4 Deploy JBoss Application with JBoss Developer Studio 6.....	95
5.3.5 Collecting Information from the Application (user).....	104
5.3.6 Scaling (user).....	106
5.3.7 Jenkins Enablement (user).....	109
5.3.8 Clean Up.....	118
5.4 Node Gear & District Management.....	121
5.4.1 Add a Node to the Infrastructure.....	122
5.4.2 Enabling Node Profiles.....	122
5.4.3 Districts.....	122
5.4.3.1 Create Districts.....	123
5.4.3.2 Move Applications and Add a Node to District.....	124
5.5 Users.....	130
6 Conclusion.....	132
Appendix A: Revision History.....	133
Appendix B: Contributors.....	134
Appendix C: ActiveMQ Configuration Files.....	135
Appendix D: Configuration Files.....	143



Appendix E: Configure Microsoft Windows 2008 R2 Active Directory Server 144



1 Executive Summary

A Platform-as-a-Service, or PaaS, is a cloud application platform in which the application configuration, build, deployment, hosting, and administration is automated in an elastic cloud environment along with their supported stacks. OpenShift is a Platform as a Service (PaaS) solution from Red Hat. OpenShift provides a multi-language, auto-scaling, self-service, elastic cloud application platform built on a proven stack of Red Hat technologies. The OpenShift PaaS services are available as both a hosted service and on-premise PaaS product offering. OpenShift Enterprise is Red Hat's on-premise PaaS software solution that enables customers to deploy their own Private or Hybrid PaaS environment.

OpenShift Enterprise enables administrators to more quickly serve the needs of application developers by deploying a PaaS platform that streamlines the application service delivery process. OpenShift Enterprise enables administrators and developers to more quickly serve their needs by standardizing and accelerating developer work-flows. OpenShift Enterprise does this in a highly-efficient, fine-grained, multi-tenant cloud architecture that maximizes infrastructure utilization. This provides application developers with self-service access so they can easily deploy applications on-demand leveraging the languages, frameworks, and tools of their choosing. It ultimately increases developer efficiency and agility by allowing them to focus on what they do best, writing code and in turn enables them to better meet the demands of the business.

OpenShift Enterprise is built on the strength of a proven stack of Red Hat technologies. It provides the freedom for developers and administrators to work the way they want to work. For developers, OpenShift Enterprise supports Java, Ruby, PHP, Python, and Perl programming languages with associated frameworks as well as additional platform services like MySQL and PostgreSQL databases, Jenkins Continuous Integration server, and more. OpenShift Enterprise "cartridges" provide these platform services. Cartridges are extensible, enabling customers to extend OpenShift to add their own custom services. Developers can access OpenShift Enterprise via a rich command line interface, web console, RESTful API, or Eclipse integrated development environment (IDE). For IT administrators, OpenShift Enterprise runs on the trusted Red Hat Enterprise Linux platform and leverages technologies such as SELinux and Cgroups to provide a secure and scalable multi-tenant environment, that allows efficient infrastructure utilization supporting application development and deployment. OpenShift also provides standards-based application run-time systems like JBoss Enterprise Application Platform, Tomcat, and Apache. OpenShift Enterprise can be deployed on-premise in a customer's data center or private cloud on their choice of virtualization platform or cloud provider. In addition, developers can work from a variety of workstations including Red Hat Linux, Mac and Windows. OpenShift Enterprise is open source, based on the upstream OpenShift Origin project, that helps drive innovation and eliminate lock-in.

This reference architecture shows how to deploy OpenShift Enterprise in a distributed fashion, separating domain name services, ActiveMQ, and MongoDB from the OpenShift Enterprise broker host. Other items that are included in this reference architecture are how to



deploy both PHP and Java applications, enable applications with Jenkins continuous integration services, use JBoss Developer Studio within a OpenShift Enterprise Environment, and other operational management features. In addition, a Microsoft Active Directory lightweight directory access protocol (LDAP) service is used for authentication.



2 Components Overview

2.1 OpenShift Enterprise

2.1.1 Broker

The broker is the single point of contact for all application management activities. It is responsible for the following tasks:

- Managing user logins
- DNS
- Application state
- Application orchestration
- Services and operations

In addition to being able to contact the broker directly via a RESTful API, the following methods may also be used

- Web console
- Command line interface (CLI) tools
- Eclipse JBoss Developer Studio

OpenShift Enterprise is composed of several components that make the above tasks possible. This reference architecture focuses on distributing OpenShift Enterprise for the purposes of high availability. In order to do that, the following services are configured on separate systems, which are typically provided by the broker:

- MongoDB
- ActiveMQ
- Authentication Services
- BIND

Mongo is set up in a replicated configuration over three hosts. The same hosts provide ActiveMQ services for messaging. In this configuration, if one broker or broker support node (bsn) host goes down, the OpenShift Enterprise infrastructure is still operational. Product documentation for OpenShift Enterprise can be found on Red Hat's customer portal¹.

2.1.2 Node

An OpenShift node is a host that runs the applications. There can be many nodes that are deployed that a broker manages. A node supports gears that contain applications. One of the features of OpenShift is that it is a true multi-tenancy environment. This is accomplished by using SELinux and Cgroup restrictions so as to separate each applications' resources and data.

A node and a broker can be on the same host but that is not recommended. This reference



architecture has deployed the node services onto a set of dedicated node hosts.

2.2 Load Balancer Add-On for Red Hat Enterprise Linux 6

The Load Balancer Add-On for Red Hat Enterprise Linux provides support for Transmission Control Protocol (TCP) load balancing independent of applications. It is composed of two major components: the Linux Virtual Server (LVS) and the Piranha Configuration Tool, a management tool with a graphical user interface (GUI).

This reference architecture shows how to set up a load balancing cluster that consists of a active / backup configuration that is load balancing Hypertext Transfer Protocol Secure (https) traffic to a set of OpenShift Enterprise brokers. The configuration is set up in a direct routing method and a round robin load balancing algorithm. In addition, the LVS cluster uses least connections with persistence enabled to ensure clients are sent to the same broker when possible. For more information on administering the Load Balancer Add-on for Red Hat Enterprise Linux, please see the product documentation².

2.3 Red Hat Enterprise Virtualization

The Red Hat Enterprise Virtualization portfolio is an end-to-end virtualization solution, with use cases for both servers and desktops, designed to overcome current IT challenges for consolidation, enable pervasive data center virtualization, and unlock unprecedented capital and operational efficiency. The Red Hat Enterprise Virtualization portfolio builds upon the Red Hat Enterprise Linux platform that is trusted by thousands of organizations on millions of systems around the world for their most mission-critical workloads. Combined with KVM, the latest generation of virtualization technology, Red Hat Enterprise Virtualization delivers a secure, robust virtualization platform with unmatched performance and scalability for Red Hat Enterprise Linux and Windows guests. Red Hat Enterprise Virtualization shares the same platform certification as Red Hat Enterprise Linux for compatibility and performance.

2.4 Microsoft Windows

2.4.1 Windows Server 2008 R2

Windows Server 2008 R2 is Microsoft's enterprise operating system for businesses and provides features for virtualization, power savings, manageability and mobile access. Windows Server 2008 R2 is available in several editions – Foundation, Standard, Enterprise, Datacenter, Web, and HPC (High Performance Computing). Windows Server 2008 R2 Enterprise Edition is used for the configuration described in this reference architecture.

2.4.2 Active Directory Domain Services (AD DS)

Active Directory Domain Services is a suite of directory services developed by Microsoft. Active Directory utilizes customized versions of industry standard protocols including:

- Kerberos
- Domain Name System (DNS)
- Lightweight Directory Access Protocol (LDAP)



Active Directory allows Windows system administrators to securely manage directory objects from a scalable, centralized database infrastructure. Directory objects (users, systems, groups, printers, applications) are stored in a hierarchy consisting of nodes, trees, forests and domains. Prior to Windows Server 2008 R2, Active Directory Domain Services was known as Active Directory. Active Directory Domain Services is included with Windows Server 2008 R2. This reference architecture only used Microsoft Windows 2008 R2 for authentication services.

2.5 BIND

Berkeley Internet Name Daemon (BIND) is a implementation of the Domain Name System (DNS) protocols. BIND enables a human or computer to look-up another computer on the basis of a name. BIND contains three parts:

- DNS Server – answers queries that are sent to it
- DNS Resolver Library – library that can be added to other programs that provides the ability to resolve names ensuring DNS standards are being followed
- Tools for Testing Servers – Tools used for testing, such as **dig**, **nslookup** and **host**.

BIND is an integral part of a successful OpenShift Enterprise deployment / environment. OpenShift Enterprise uses a RubyGem called **dnsmruby** to issue dynamic updates to the DNS environment. The requirement that allows this functionality is that the DNS environment supports RFC 2136 (Dynamic DNS Updates). Any DNS implementation that supports RFC 2136 will work in a OpenShift Environment, however, this reference architecture uses BIND.

2.6 Red Hat Enterprise Linux

Red Hat Enterprise Linux 6, the latest release of Red Hat's trusted datacenter platform, delivers advances in application performance, scalability, and security. With Red Hat Enterprise Linux 6, physical, virtual, and cloud computing resources can be deployed within the data center. Red Hat Enterprise Linux 6.3 provides the following features and capabilities:

Reliability, Availability, and Security (RAS):

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

File Systems:

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

High Availability:

- Extends the current clustering solution to the virtual environment allowing for high availability of virtual machines and applications running inside those virtual machines



- Enables NFSv4 resource agent monitoring
- Introduction of Cluster Configuration System (CCS). CCS is a command line tool that allows for complete CLI administration of Red Hat's High Availability Add-On

Resource Management:

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

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



3 Reference Architecture Configuration

3.1 Environment

This reference architecture environment consists of all the components required to build a OpenShift Enterprise PaaS in a distributed fashion. The best practices that the reference architecture team learned in the lab are shared throughout this document.



There are many different ways to deploy a PaaS with OpenShift Enterprise. One solution would be to deploy everything on a single server which includes the broker, DNS, node, MongoDB, ActiveMQ, and other services. This could be used for development purposes or a proof of concept. Another way might be to distribute every service onto a different host – and then cluster those hosts to ensure maximum availability. As depicted in **Figure 3.1.1: Distributed Infrastructure**, this reference architecture tries to meet those two solutions in the middle by separating the MongoDB and ActiveMQ services onto three Broker Support Node's (BSN) as well as the DNS, node and load balanced broker services onto separate virtual machines.

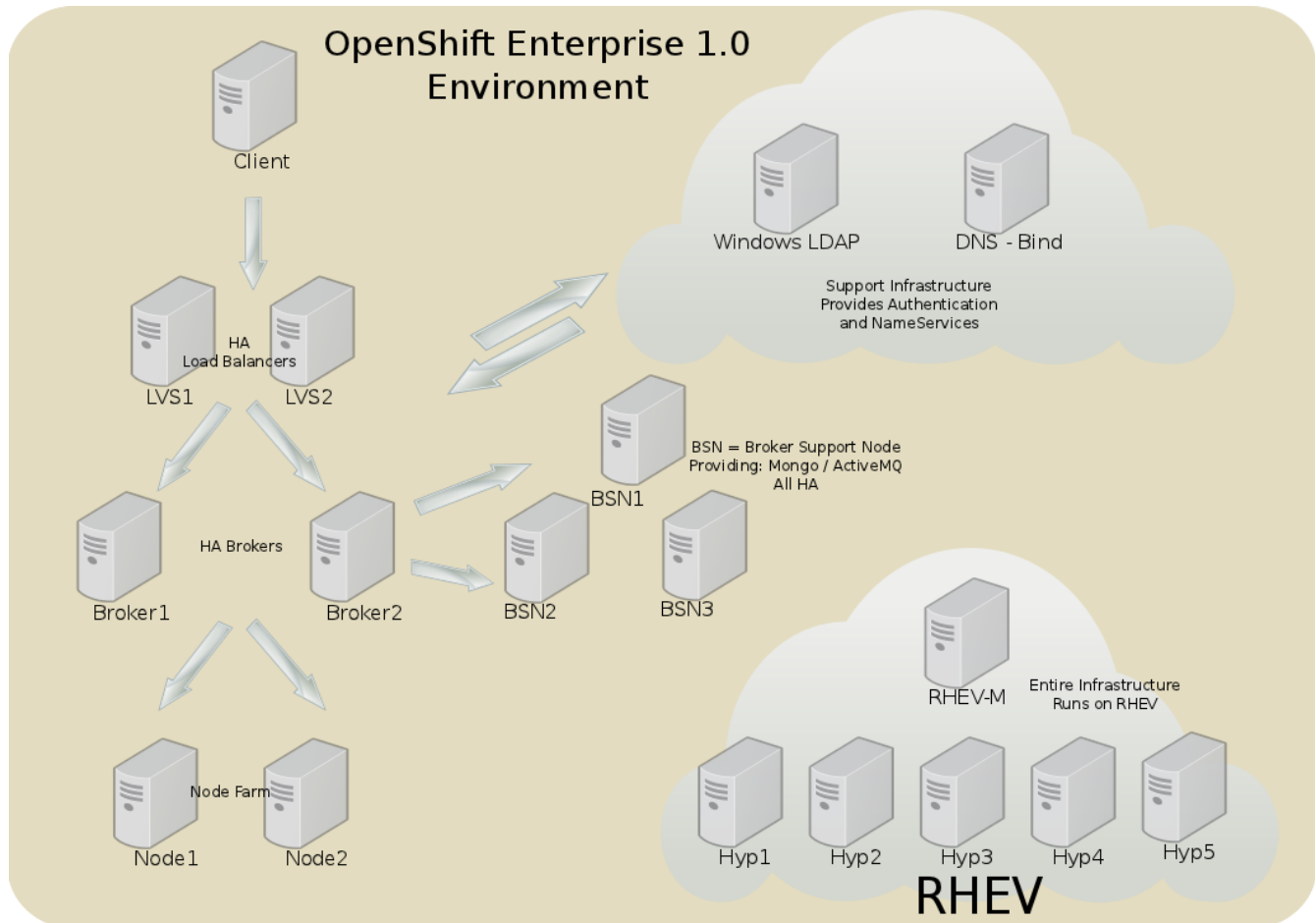


Figure 3.1.1: Distributed Infrastructure



3.2 Virtual Machines

All systems in this reference architecture are virtual machines except for the Red Hat Enterprise Virtualization hypervisors. The virtual machines are listed in **Table 3.2.1: Virtual Machines**. To create a more redundant infrastructure, the virtual machines of the same type were configured with Red Hat Enterprise Virtualization hypervisor preferences. The reason for doing this is two fold: distribute resources and high availability.

Hostname	Software	Role	Version
client	Red Hat Enterprise Linux	Client	2.6.32-279.11.1.el6
lvs{1,2}	Red Hat Enterprise Linux	Load Balancer	2.6.32-279.14.1.el6
broker{1,2}	Red Hat Enterprise Linux	Broker	2.6.32-279.14.1.el6
node{1,2}	Red Hat Enterprise Linux	Node	2.6.32-279.14.1.el6
bsn{1,2,3}	Red Hat Enterprise Linux	Broker Support Node	2.6.32-279.14.1.el6
ldapwin	Windows 2008 R2	Authentication	Version 6.1 (Build 7601: Service Pack 1)
rhev	Red Hat Enterprise Linux	Red Hat Enterprise Virtualization Manager	2.6.32-279.11.1.el6 RHEV-M: 3.1.0-26.el6ev
hyper{1-4}	Red Hat Enterprise Linux / KVM	Red Hat Enterprise Virtualization Hypervisor	6.3.0.3.el6 VDSM Version: 4.9.6-41.0.el6_3
hyper5	RHEV Hypervisor	Red Hat Enterprise Virtualization Hypervisor	2.6.32 – 279.14.1.el6 VDSM Version: 4.9.6-41.0.el6_3
dns	Red Hat Enterprise Linux	Dynamic DNS Update Server	BIND 9.8.2-0.10.rc1.el6_3

Table 3.2.1: Virtual Machines



3.3 Software Configuration

3.3.1 Configuration Files

All of the configuration files needed to reproduce this reference architecture are included on the Red Hat customer portal³. A log-in is required to access the configuration files. A list of all the files include can be seen in Appendix D: **Configuration Files**.

3.3.2 Virtual Machine Sizing

Hostname	Disk	Memory	CPU
client	30 GB	4GB	1 Virtual Core
lvs{1,2}	15 GB	2GB	1 Virtual Core
broker{1,2}	15GB	4GB	1 Virtual Core
node{1,2,3}	20GB	8GB	2 Virtual Core
bsn{1,2,3}	15 GB	4GB	1 Virtual Core
ldapwin	25GB	4GB	1 Virtual Core
rhevm	50GB	8GB	4 Virtual Cores
dns	15	1GB	1 Virtual Core

Table 3.3.1: Virtual Machine Sizing



3.3.3 Virtual Machine IP Addresses

Host	IP Address
client	10.16.138.20
broker1	10.16.138.25
broker2	10.16.138.26
broker	10.16.138.34
bsn1	10.16.138.28
bsn2	10.16.138.27
bsn3	10.16.138.31
node1	10.16.138.29
node2	10.16.138.30
node3	10.16.138.57
dns	10.16.138.14
lvs1	10.16.138.18
lvs2	10.16.138.19
broker	10.16.138.34
ldapwin	10.16.138.21

Table 3.3.2: Host IP Addresses



3.3.4 Required Channels

Host	Channels	Purpose
broker{1,2}	rhel-x86_64-server-6	RHEL Base
	rhel-x86_64-server-6-osop-1-infrastructure	OpenShift Enterprise Infrastructure
	rhel-x86_64-rhev-agent-6-server	RHEV Agent
bsn{1,2,3}	rhel-x86_64-server-6	RHEL Base
	rhel-x86_64-server-6-osop-1-infrastructure	OpenShift Enterprise Infrastructure
	rhel-x86_64-rhev-agent-6-server	RHEV Agent
dns	rhel-x86_64-server-6	RHEL Base
node{1,2,3}	rhel-x86_64-server-6-osop-1-node	OpenShift Enterprise Node
	rhel-x86_64-server-6-osop-1-jbosseap	JBoss EAP 6 Cartridge Support
	jbappplatform-6-x86_64-server-6-rpm	JBoss EAP 6
	jb-ews-2-x86_64-server-6-rpm	JBoss Enterprise Web Server
lvs1	rhel-x86_64-server-6	RHEL Base
	rhel-x86_64-server-lb-6	Red Hat Load Balancer
	rhel-x86_64-rhev-agent-6-server	RHEV Agent

Table 3.3.3: Required Software Channels



3.3.5 Software Revisions

The software running on each host is:

Host	Software	Version
Broker	openshift-origin-broker	1.0.1-1.el6op
	openshift-origin-broker-util	1.0.5-1.el6op
	rubygem-openshift-origin-dns-bind	1.0.1-1.el6op
	rubygem-openshift-origin-auth-remote-user	1.0.3-1.el6op
	rubygem-openshift-origin-msg-broker-mcollective	1.0.2-1.el6op
Node	rubygem-openshift-origin-node	1.0.7-1.el6op
	rubygem-passenger-native	3.0.17-2.el6op
	openshift-origin-port-proxy	1.0.1-1.el6op
	openshift-origin-node-util	1.0.5-1.el6op
BSN	activemq	5.6.0-1.el6op
	mongodb-server	2.0.2-2.el6op
DNS	bind	9.8.2-0.10.rc1

Table 3.3.4: Software Versions



3.3.6 Security

SELinux

SELinux is enabled and enforcing on all Red Hat machines in this reference architecture environment.

Additional Services

Each system has the **rhev-agent** and **ntp** running as well.

Ports

The network services ports are shown in **Table 3.3.5: Ports**.

Port	Host	Protocol
22,443,539,3636	LVS1	TCP
22,443,539,3636	LVS2	TCP
22,80,443	Broker1	TCP
22,80,443	Broker2	TCP
22,53	DNS	TCP,UDP
22,443,27017,8161,61613,61616	BSN1	TCP
22,443,27107,8161,61613,61616	BSN2	TCP
22,443,27017,8161,61613,61616	BSN3	TCP
22	Client	TCP
22,80,35531-65535	Node1	TCP
22,80,35531-65535	Node2	TCP
389,636	LDAP	TCP

Table 3.3.5: Ports



3.4 Hardware Configuration

3.4.1 Servers

All five hypervisors are configured with the same hardware platform type:

Hardware	Specifications
Red Hat Enterprise Linux Hypervisors [5 x DELL BladeServer – PowerEdge M520]	2 x Intel(R) Xeon(R) CPU E5-2450 0 @ 2.10GHz GB (8 core)
	2 x Broadcom NetXtreme II BCM57810 10 Gigabit Ethernet 4 x Broadcom NetXtreme BCM5720 Gigabit Ethernet
	8 x DDR3 4096 MB @1600 MHZ DIMMs
	2 x 146GB SAS internal disk drives

Table 3.4.1: Server Hardware

3.4.2 Storage

Non-local storage is provided to the hypervisors by an EMC Celerra NS-120

System	Disk Usage
RHEV Hypervisors	750 GB

Table 3.4.2: Storage



4 Deploying the Infrastructure

4.1 Deploy Red Hat Enterprise Virtualization

This reference architecture uses Red Hat Enterprise Virtualization to host the virtual machines that constitute the OpenShift Enterprise 1.0 infrastructure as shown in **Figure 4.1.1: RHEV Hypervisors** and **Figure 4.1.2: Virtual Machines**. The only virtual machines that are not hosted on the Red Hat Enterprise Virtualization cluster are the DNS and RHEV-M virtual machines. The Red Hat Enterprise Virtualization environment consists of five Red Hat Enterprise Virtualization hosts, EMC iSCSI storage, one RHEV-M host, and several virtual machines.

Name	Hostname/IP	Cluster	Data Center	Status	Load	Memory	CPU	Network	SPM
hyper1.osop	10.16.138.41	Default	Default	Up	4 VMs	22%	1%	0%	Normal
hyper2.osop	10.16.138.42	Default	Default	Up	3 VMs	22%	0%	0%	Normal
hyper3.osop	10.16.138.43	Default	Default	Up	7 VMs	80%	1%	0%	Normal
hyper4.osop	10.16.138.44	Default	Default	Up	6 VMs	47%	0%	0%	Normal
hyper5.osop	10.16.138.55	Default	Default	Up	7 VMs	58%	1%	0%	SPM

Figure 4.1.1: RHEV Hypervisors

Name	Host	IP Address	Cluster	Data Center	Memory	CPU	Network	Display	Status	Uptime	Logged-in U
broker1	hyper3.osop	10.16.138.25	Default	Default	16%	2%	0%	Spice	Up	6 days	
broker2	hyper4.osop	10.16.138.26	Default	Default	15%	2%	0%	Spice	Up	5 days	
broker3	hyper1.osop	10.16.138.45	Default	Default	24%	2%	0%	Spice	Up	6 days	
broker-spr	hyper3.osop	10.16.138.13	Default	Default	0%	3%	0%	Spice	Up	6 days	
bsn1	hyper2.osop	10.16.138.28	Default	Default	0%	10%	0%	Spice	Up	17 days	
bsn2	hyper5.osop	10.16.138.27	Default	Default	13%	3%	0%	Spice	Up	17 days	
bsn3	hyper4.osop	10.16.138.31	Default	Default	13%	2%	0%	Spice	Up	17 days	
client.osop	hyper3.osop	10.16.138.20	Default	Default	14%	0%	0%	Spice	Up	7 days	
lvs1	hyper4.osop	10.16.138.18	Default	Default	18%	2%	0%	Spice	Up	7 days	
lvs2	hyper3.osop	10.16.138.19	Default	Default	18%	1%	0%	Spice	Up	7 days	
nagios	hyper4.osop	10.16.138.22	Default	Default	30%	2%	0%	Spice	Up	52 days	
node0-spr	hyper5.osop		Default	Default	0%	1%	0%	Spice	Up	6 days	
node1	hyper3.osop		Default	Default	0%	1%	0%	Spice	Up	7 days	
node2	hyper4.osop	10.16.138.30	Default	Default	11%	1%	0%	Spice	Up	3 days	
weba	hyper5.osop	10.16.138.36	Default	Default	22%	1%	0%	Spice	Up	24 days	
webb	hyper4.osop	10.16.138.37	Default	Default	22%	1%	0%	Spice	Up	26 days	
Windows-LDAP	hyper5.osop	10.16.138.21	Default	Default	22%	6%	0%	Spice	Up	26 days	admin@inte

Figure 4.1.2: Virtual Machines



4.2 Deploy BIND

This reference architecture uses BIND for name resolution for the OpenShift Enterprise 1.0 infrastructure. The reference architecture lab has an existing DNS server that is used for name resolution. OpenShift Enterprise 1.0 requires that a name server is able to process dynamic updates. Rather than use our existing DNS server for this purpose another BIND based DNS server was deployed and configured to process dynamic updates. This section demonstrate how the DNS infrastructure was configured as shown in **Figure 4.2.1: Bind Configuration**.

The only requirements here are that the upstream DNS server is configured to accept zone transfers from the DNS server that is taking dynamic updates.

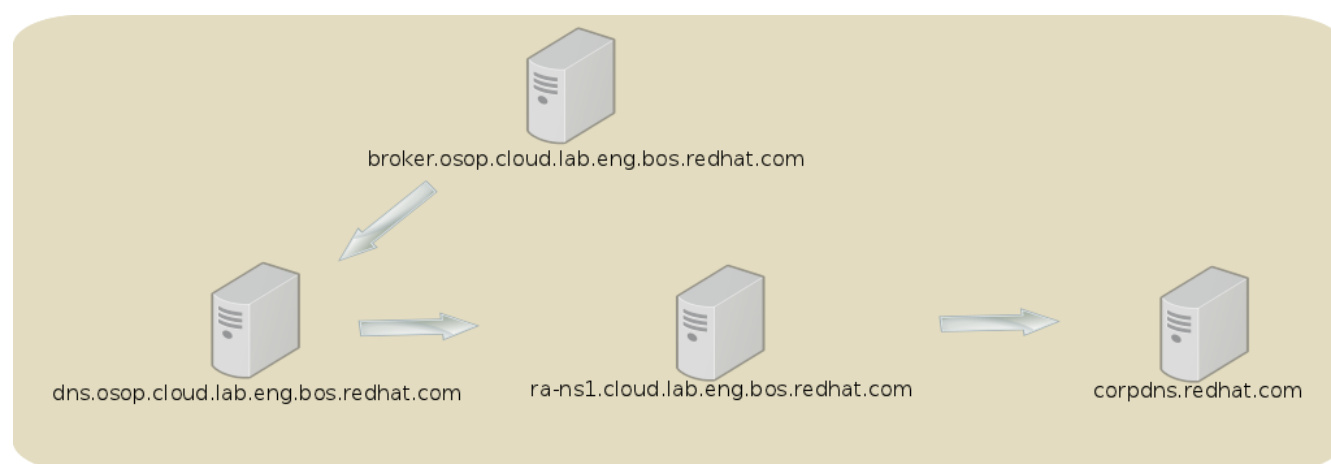


Figure 4.2.1: Bind Configuration

To configure BIND to process dynamic updates, the following steps need to take place:

- Deploy a Red Hat Enterprise Linux in a virtual machine
- Install the BIND packages
- Configure the BIND software
- Test dynamic updates

Deploy a Red Hat Enterprise Linux server

- Follow the Red Hat Enterprise Linux Installation Guide⁴ for more details.

On the DNS Server, install and configure the BIND packages

1. Install the BIND packages and backup the `/etc/named.conf` file

```
# yum install bind.x86_64 bind-utils.x86_64bind-devel.x86_64
# cp /etc/named.conf{, .orig}
```

2. Modify the `/etc/named.conf` file, note the highlighted sections below

```
//
```



```
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

options {
    listen-on port 53 { any; };
    // listen-on-v6 port 53 { ::1; };
    forwarders { 10.16.143.248 port 53; 10.16.143.247 port 53; };
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query { any; };
    recursion yes;
    allow-transfer { any; };

    dnssec-enable yes;
    dnssec-validation yes;
    dnssec-lookaside auto;

    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";

    max-journal-size 50k;
    managed-keys-directory "/var/named/dynamic";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

zone "osop.cloud.lab.eng.bos.redhat.com." IN {
    type master;
    notify yes;
    file "slaves/osop.cloud.lab.eng.bos.redhat.com";
    allow-transfer { any; };
    allow-update { key osop.cloud.lab.eng.bos.redhat.com; };
};

zone "138.16.10.in-addr.arpa." IN {
    type master;
    notify yes;
    file "slaves/138.16.10.in-addr.arpa";
};
```



```

    allow-transfer { any; };
    allow-update { key osop.cloud.lab.eng.bos.redhat.com; };
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
include "osop.cloud.lab.eng.bos.redhat.com.key";

```

3. Configure the DNS server to allow traffic over port 53

```

# iptables --insert INPUT --protocol tcp --dport 53 --jump ACCEPT
# service iptables save

```

4. Configure the DNS update key and configure the *osop.cloud.lab.eng.bos.redhat.com*. +157+35791.private file as shown below

```

# cd /var/named
# dnssec-keygen -a HMAC-MD5 -b 512 -n USER -r /dev/urandom
osop.cloud.lab.eng.bos.redhat.com
Private-key-format: v1.3
Algorithm: 157 (HMAC_MD5)
Key:
w+/dpA05yoYkASK+7HAXmUQyEV30FR0r1MepdH/jjEC3XS0mU/TTtqyUcQXFK1flzt2HN0eFQWh5
hEn4YM4bPg==
Bits: AAA=
Created: 20121119103407
Publish: 20121119103407
Activate: 20121119103407

```

5. Place the key hash that is in the *osop.cloud.lab.eng.bos.redhat.com.key* key file into the keyfile that is included in the */etc/named.conf* file

```

key osop.cloud.lab.eng.bos.redhat.com {
    algorithm HMAC-MD5;
    secret
"w+/dpA05yoYkASK+7HAXmUQyEV30FR0r1MepdH/jjEC3XS0mU/TTtqyUcQXFK1flzt2HN0eFQWh5
hEn4YM4bPg==";
};

```

6. Create the forward zone file for the domain as displayed here in the */var/named/slaves/osop.cloud.lab.eng.bos.redhat.com*

```

$ORIGIN .
$TTL 86400 ; 1 day
osop.cloud.lab.eng.bos.redhat.com IN SOA osop.cloud.lab.eng.bos.redhat.com.
refarch-tech.osop.cloud.lab.eng.bos.redhat.com. (
    2012092608 ; serial
    10800      ; refresh (3 hours)
    3600       ; retry (1 hour)
    604800     ; expire (1 week)
    600        ; minimum (10 minutes)
)
NS dns.osop.cloud.lab.eng.bos.redhat.com.
$ORIGIN osop.cloud.lab.eng.bos.redhat.com.
dns A 10.16.138.14

```




7. Create the reverse zone file for the domain as shown in the file
`/var/named/slaves/138.16.10.in-addr.arpa`

```
$ORIGIN .
$TTL 300      ; 5 minutes
138.16.10.in-addr.arpa IN SOA      dns.osop.cloud.lab.eng.bos.redhat.com.
nobody.example.com. (
                        2012092638 ; serial
                        600        ; refresh (10 minutes)
                        1800       ; retry (30 minutes)
                        604800     ; expire (1 week)
                        300        ; minimum (5 minutes)
                        )
                        NS      dns.osop.cloud.lab.eng.bos.redhat.com.
$ORIGIN 138.16.10.in-addr.arpa.
$TTL 86400   ; 1 day
```

8. Test updates with **nsupdate** to confirm dynamic updates are functioning

```
# nsupdate -k Krefarchdns.cloud.lab.eng.bos.redhat.com.+157+54668.key
> server 10.16.138.46
> update add 105.138.16.10.in-addr.arpa 86400 IN PTR
testnode.refarchdns.cloud.lab.eng.bos.redhat.com.
> show
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:      0
;; flags:;; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; UPDATE SECTION:
105.138.16.10.in-addr.arpa. 86400 IN    PTR
      testnode.refarchdns.cloud.lab.eng.bos.redhat.com.

> send
> ^D

# rndc freeze; rndc thaw

# grep testnode /var/named/slaves/138.16.10.in-addr.arpa
105          PTR    testnode.refarchdns.cloud.lab.eng.bos.redhat.com.
```



4.3 Deploy Broker Support Nodes

The broker support nodes host two critical services for the OpenShift Enterprise infrastructure. ActiveMQ is set up in a highly available configuration. ActiveMQ is deployed in a three node pool so there is no single point of failure in the messaging infrastructure. The first is ActiveMQ and the second is MongoDB. MongoDB is also configured in a highly available replica set. There are three instances of MongoDB which ensures that there is no single point of failure in the environment.

4.3.1 ActiveMQ

ActiveMQ is set up in a Network-of-Brokers configuration. This provides the ability to distribute queues and topics among ActiveMQ brokers. This allows a client to connect to any broker in the network and in the event of a failure – fail over to another broker. More information is provided on the apache.org⁵ website.

To configure ActiveMQ the following steps need to take place:

- Deploy Red Hat Enterprise Linux on 3 servers
- Install ActiveMQ software on bsn{1,2,3} nodes
- Configure ActiveMQ on all BSN nodes
- Start the ActiveMQ service

Deploy a Red Hat Enterprise Linux server

- Follow the Red Hat Enterprise Linux Installation Guide⁶ for more details.

Install ActiveMQ on each BSN host.

1. The reference architecture systems use DHCP to assign a static IP address. Ensure that the DHCP client points to the correct DNS server and has the appropriate host / domain name entries. This can be done by adding the following entries (in bold) to the `/etc/dhcp/dhclient-eth0.conf` file. Change this on each BSN host.

```
send vendor-class-identifier "anaconda-Linux 2.6.32-279.el6.x86_64
x86_64";
timeout 45;
prepend domain-name-servers 10.16.138.14;
supersede host-name "bsn1";
supersede domain-name "osop.cloud.lab.eng.bos.redhat.com";
```

Ensure that the “host-name” entry is correct for the host this file is being put on.

2. Subscribe to the appropriate Red Hat Network channels and install ActiveMQ on each BSN host.

```
# rhn-channel --add --channel rhel-x86_64-server-6-osop-1-infrastructure
Username: admin
Password:

# yum install activemq
```



- Perform steps 3 and 4 on bsn1 to configure ActiveMQ and then copy the modified files to the other BSN hosts. Ensure when the files are copied over that they contain the proper hostname and information that is specific to that host.

3. Configure the `/etc/activemq/activemq.xml`

(a) Create a backup of each BSN's configuration file.

```
# cp /etc/activemq/activemq.xml{, .orig}
```

- There are quite a few changes in the `/etc/activemq/activemq.xml` file. This section attempts to explain each change that is made. In order to do this the file is broken up into sections. A complete copy of each BSN hosts `/etc/activemq/activemq.xml` file can be found in **Appendix C: ActiveMQ Configuration Files**.

(b) Change the brokerName, data directory, and add useJmx=true in the `/etc/activemq/activemq.xml` file.

```
<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="localhost" dataDirectory="${activemq.data}">
```

To:

```
<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="bsn1.osop.cloud.lab.eng.bos.redhat.com" useJmx="true"
dataDirectory="${activemq.base}/data">
```

Change the hostname parameter to correspond to the host that the `/etc/activemq/activemq.xml` file is on.

(c) Change the destinationPolicy element from:

```
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <policyEntry topic="" producerFlowControl="true"
memoryLimit="1mb">
        <pendingSubscriberPolicy>
          <vmCursor />
        </pendingSubscriberPolicy>
      </policyEntry>
      <policyEntry queue="" producerFlowControl="true"
memoryLimit="1mb">
      </policyEntry>
    </policyEntries>
  </policyMap>
</destinationPolicy>
```

To:

```
<destinationPolicy>
  <policyMap>
    <policyEntries>
      <!-- <policyEntry topic="" producerFlowControl="true"
```



```
memoryLimit="1mb"> -->
    <policyEntry topic=">" producerFlowControl="false"/>
    <policyEntry queue="*.reply.>" gcInactiveDestinations="true"
inactiveTimeoutBeforeGC="300000" />
    </policyEntries>
  </policyMap>
</destinationPolicy>
```

(d) Comment out the <persistenceAdapter> element:

```
<persistenceAdapter>
  <kahaDB directory="${activemq.data}/kahadb"/>
</persistenceAdapter>
```

To:

```
<!--
<persistenceAdapter>
  <kahaDB directory="${activemq.data}/kahadb"/>
</persistenceAdapter>
-->
```



- (e) The networkConnectors provide one way message paths to the other ActiveMQ brokers. In order to have a completely fault tolerant configuration, each ActiveMQ broker's networkConnector must point to the other ActiveMQ brokers as shown in **Figure 4.3.1.1: Network of Brokers**.

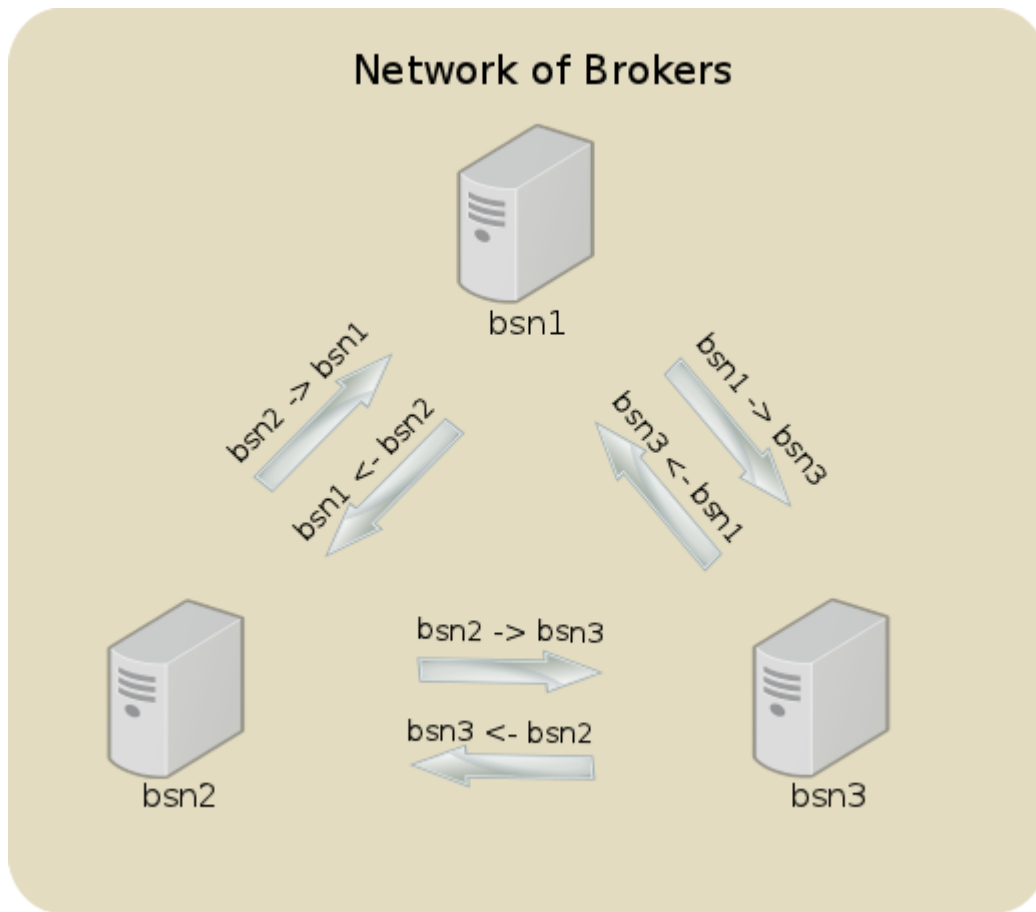


Figure 4.3.1.1: Network of Brokers

This section of the file is specific for each BSN host, so an example `<networkConnectors>` section is provide for each one. The `<networkConnectors>` section goes after the `</persistenceAdapter>` section. In addition, complete activemq configuration files can be found on the reference architecture website. Each `<networkConnector>` needs a unique name for each element on each ActiveMQ broker. The names used here are in the “localhost -> remotehost” format. For example, the bsn1 ActiveMQ host has bsn1-bsn2 and the static URI IP address corresponds with the bsn2 host. Likewise, bsn1 also has a `<networkConnector>` set up from bsn1-bsn3, and the static URI IP address corresponds with the bsn3 host. The `<topics>` are the same for each `<staticallyIncludedDestination>`, and the `<topics>`.



bsn1:

```
<networkConnectors>
  <networkConnector name="bsn1-bsn2" duplex="true" uri="static:
(tcp://10.16.138.27:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.openshift.reply"/>
      <topic physicalName="mcollective.discovery.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>

  <networkConnector name="bsn1-bsn3" duplex="true" uri="static:
(tcp://10.16.138.31:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.openshift.reply"/>
      <topic physicalName="mcollective.discovery.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>
</networkConnectors>
```

bsn2:

```
<networkConnectors>
  <networkConnector name="bsn2-bsn1" duplex="true" uri="static:
(tcp://10.16.138.28:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>

  <networkConnector name="bsn2-bsn3" duplex="true" uri="static:
(tcp://10.16.138.31:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>
</networkConnectors>
```

bsn3:

```
<networkConnectors>
  <networkConnector name="bsn3-bsn1" duplex="true" uri="static:
(tcp://10.16.138.28:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>

  <networkConnector name="bsn3-bsn2" duplex="true" uri="static:
(tcp://10.16.138.27:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>
</networkConnectors>
```



```

    </staticallyIncludedDestinations>
  </networkConnector>
</networkConnectors>

```

- (f) Add the plugins section to allow mcollective authorization. The <plugins> element goes after the </networkConnectors> element. This is on each BSN host.

```

<plugins>
  <statisticsBrokerPlugin/>
  <simpleAuthenticationPlugin>
    <users>
      <authenticationUser username="mcollective"
password="marionette" groups="mcollective,everyone"/>
      <authenticationUser username="amq" password="password"
groups="admins,everyone"/>
      <authenticationUser username="admin" password="secret"
groups="mcollective,admin,everyone"/>
    </users>
  </simpleAuthenticationPlugin>
  <authorizationPlugin>
    <map>
      <authorizationMap>
        <authorizationEntries>
          <authorizationEntry queue=">" write="admins" read="admins"
admin="admins" />
          <authorizationEntry topic=">" write="admins" read="admins"
admin="admins" />
          <authorizationEntry topic="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
          <authorizationEntry queue="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
          <authorizationEntry topic="ActiveMQ.Advisory.>"
read="everyone,all" write="everyone,all" admin="everyone,all"/>
        </authorizationEntries>
      </authorizationMap>
    </map>
  </authorizationPlugin>
</plugins>

```

- (g) Add the stomp transportConnector to the <transportConnectors> element. The following <transportConnectors> element should contain the following. Leave the openwire connector as is. That is used for intra-activemq broker communications.

```

<transportConnectors>
  <transportConnector name="openwire" uri="tcp://0.0.0.0:61616"/>
  <transportConnector name="stomp" uri="stomp://0.0.0.0:61613"/>
</transportConnectors>

```

4. Secure the ActiveMQ console by configuring the */etc/activemq/jetty.xml* . This file is copied to each BSN host.

- (a) Enable authentication and restrict the console to localhost

```

# cp /etc/activemq/jetty.xml{,.orig}
# sed -i -e '/name="authenticate"/s/false/true/' /etc/activemq/jetty.xml
# sed -i -e '/name="port"/a<property name="host" value="127.0.0.1" />'

```



```
/etc/activemq/jetty.xml
```

(b) Change the default admin password in the */etc/activemq/jetty-realm.properties* file, choose a password that is appropriate. This file is copied to each BSN host.

```
# cp /etc/activemq/jetty-realm.properties{,.orig}
# sed -i -e '/admin:/s/admin,/badpassword,/' /etc/activemq/jetty-
realm.properties
```

5. Copy the XML files to the other BSN hosts

```
# for HOST in 2 3; do for FILES in /etc/activemq/jetty-realm.properties
/etc/activemq/activemq.xml /etc/activemq/jetty.xml; do scp $FILES bsn$HOST:
$FILES; done; done
```

After copying the file to each host, make sure to change the hostname in the *activemq.xml* file to the appropriate hostname. Also change the `<networkConnectors>` element to match the appropriate host.

Complete the following tasks on each BSN host.

6. Modify the firewall to allow ActiveMQ stomp and openwire traffic

```
# lokkit --port=61613:tcp
# lokkit --port=61616:tcp
```

7. Start ActiveMQ and ensure ActiveMQ starts on boot

```
# service activemq restart
Starting ActiveMQ Broker...

# chkconfig activemq on
```

8. Confirm that authentication is working, look for a “200” return value

```
# curl --head --user admin:PASSWORD
http://localhost:8161/admin/xml/topics.jsp
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=2nm3kfarmvl7ww73vmitvb9m;Path=/admin
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/xml; charset=ISO-8859-1
Content-Length: 28
Server: Jetty(7.6.1.v20120215)
```

9. The topics will be empty, but confirm the topics are available. Right now a response is adequate.

```
# curl --user admin:PASSWORD --silent
http://localhost:8161/admin/xml/topics.jsp | grep -A 4 topic

<topics>

</topics>
```




4.3.2 MongoDB

To configure MongoDB the following steps need to take place:

- Install MongoDB software on bsn{1,2,3} nodes
- Configure MongoDB on each BSN node
- Start Mongo on each node
- Set up replication – this is only done on one host

Perform the following task on each node.

1. Install MongoDB server on each BSN node

```
# yum install mongodb-server
```

2. Create a backup of the */etc/mongodb.conf* file

```
# cp /etc/mongodb.conf{, .orig}
```

3. Make the following modifications to the */etc/mongodb.conf* file. The following instructions can be done on bsn1 and copied to the other BSN hosts.

Comment out *bind_ip* flag, enabling remote access.

```
#bind_ip = 127.0.0.1
```

Add the following flags, explanations below

```
auth = true
rest = true
smallfiles = true
keyFile = /etc/mongodb.keyfile
replSet = ose
journal = true
```

- **auth** enables the mongodb authentication system. Forces a login to access databases or other information.
- **rest** enables the simple rest API
- **replSet** enables replication. As long as replSet is consistent among all the members, replication can take place
- **keyFile** enables authentication for the replica set and specifies which key is to be used for each member when authenticating to each other
- **journal** enables writes to be journaled. This enables faster recoveries from crashes
- **smallfiles** reduces the initial size of data files and limits the files to 512MB and reduces the size of the journal from 1GB to 128MB

4. Create the MongoDB key file with the appropriate permissions. Do this only on bsn1.

```
# echo "OSEnterprise" > /etc/mongodb.keyfile
```

Change the “OSEnterprise” key to something that satisfies your corporate security needs.

5. Copy the */etc/mongodb.keyfile* and the */etc/mongodb.conf* to the other BSN hosts

```
# for HOST in 2 3; do for FILES in /etc/mongodb.conf /etc/mongodb.keyfile;
do scp $FILES bsn$HOST:$FILES; done; done
```



6. Ensure the correct permissions and ownership are on the `/etc/mongodb.keyfile`. This should be performed on all BSN hosts.

```
# chown --verbose mongodb.mongod /etc/mongodb.keyfile
changed ownership of '/etc/mongodb.keyfile' to mongodb:mongod

# chmod --verbose 700 /etc/mongodb.keyfile
mode of '/etc/mongodb.keyfile' changed to 0700 (rwx-----)
```

7. Configure the firewall for MongoDB traffic on each BSN host to only allow traffic from the broker hosts and the other BSN hosts, no other. Even with this traffic being restricted, the version of MongoDB shipped with this solution is not complied with SSL support. So all traffic is transmitted via clear text. One other option is to also implement SSH tunnels in-between the hosts to secure the traffic even further.

bsn1:

```
# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.25 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.26 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.27 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.31 --dport 27017
--jump ACCEPT

# service iptables save
```

bsn2:

```
# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.25 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.26 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.28 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.31 --dport 27017
--jump ACCEPT

# service iptables save
```

bsn3:

```
# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.25 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.26 --dport 27017
--jump ACCEPT
```



```
# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.27 --dport 27017
--jump ACCEPT

# iptables -I INPUT -i eth0 -p tcp --source 10.16.138.28 --dport 27017
--jump ACCEPT

# service iptables save
```

There are different ways to open and close firewall ports on Red Hat Enterprise Linux, this method uses **lokkit**, one other method is via **iptables**.

8. Start the MongoDB service on each BSN host and ensure that it starts when the node boots

```
# service mongod start
Starting mongod: [ OK ]

# chkconfig mongod on
```

9. Initialize the replica set by opening a MongoDB shell and issuing the following commands. This only needs to be performed on the node that is going to be set as primary.

```
# mongo

> rs.initiate()
PRIMARY> rs.add("bsn2.osop.cloud.lab.eng.bos.redhat.com:27017")
{ "ok" : 1 }

PRIMARY> rs.add("bsn3.osop.cloud.lab.eng.bos.redhat.com:27017")
{ "ok" : 1 }

PRIMARY> rs.status()
{
  "set" : "osop",
  "date" : ISODate("2012-11-02T17:11:34Z"),
  "myState" : 1,
  "members" : [
    {
      "_id" : 0,
      "name" : "bsn1.osop.cloud.lab.eng.bos.redhat.com:27017",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "optime" : {
        "t" : 1351876256000,
        "i" : 1
      },
      "optimeDate" : ISODate("2012-11-02T17:10:56Z"),
      "self" : true
    },
    {
      "_id" : 1,
      "name" : "bsn3.osop.cloud.lab.eng.bos.redhat.com:27017",
      "health" : 1,
```



```

        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 161,
        "optime" : {
            "t" : 1351876256000,
            "i" : 1
        },
        "optimeDate" : ISODate("2012-11-02T17:10:56Z"),
        "lastHeartbeat" : ISODate("2012-11-02T17:11:33Z"),
        "pingMs" : 58
    },
    {
        "_id" : 2,
        "name" : "bsn2.osop.cloud.lab.eng.bos.redhat.com:27017",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 10,
        "optime" : {
            "t" : 1351876256000,
            "i" : 1
        },
        "optimeDate" : ISODate("2012-11-02T17:10:56Z"),
        "lastHeartbeat" : ISODate("2012-11-02T17:11:34Z"),
        "pingMs" : 19615112
    }
],
"ok" : 1
}

```

10. Create user for the openshift_broker_dev database. Do this only on bsn1.

```

# mongo openshift_broker_dev --eval 'db.addUser("openshift", "mo00")'
MongoDB shell version: 2.0.2
connecting to: openshift_broker_dev
{
  "n" : 0,
  "lastOp" : NumberLong("5806265282716499969"),
  "connectionId" : 31,
  "err" : null,
  "ok" : 1
}
{
  "user" : "openshift",
  "readOnly" : false,
  "pwd" : "8f5b96dbda3a3cd0120d6de44d8811a7",
  "_id" : ObjectId("5093ff83a547154b482000c8")
}

```

The password of “mo00” is a default password here, make sure to change it to something that meets corporate IT policy.

11. Confirm replication is taking place on bsn{2,3}. Issue the following command on both bsn{2,3}.

```
# mongo
```



```
MongoDB shell version: 2.0.2  
connecting to: test
```

```
SECONDARY> show dbs  
admin (empty)  
local 1.06201171875GB  
openshift_broker_dev 0.0625GB
```



4.4 Deploy Brokers

This section demonstrates deploying the software using the Red Hat Network (RHN).

To configure a OpenShift Enterprise broker the following steps are required:

- Ensure the proper OpenShift channels are accessible
- Install the broker software
- Configure the Broker Software
- Configure LDAP authentication

Perform the following tasks on each broker

1. Add the infrastructure channel to each broker

```
# rhn-channel --add --channel rhel-x86_64-server-6-osop-1-infrastructure  
Username: admin  
Password:
```

2. Ensure the date/time is correct, the lab systems have been configured with NTP.

```
# date
```

3. The reference architecture systems use DHCP to assign a static IP address. Ensure that the DHCP client points to the correct DNS server and has the appropriate host / domain name entries. This can be done by adding the following entries (in bold) to the `/etc/dhcp/dhclient-eth0.conf` file. Change this on each broker.

```
send vendor-class-identifier "anaconda-Linux 2.6.32-279.el6.x86_64 x86_64";  
timeout 45;  
prepend domain-name-servers 10.16.138.14;  
supersede host-name "broker1";  
supersede domain-name "osop.cloud.lab.eng.bos.redhat.com";
```

The IP address 10.16.138.14 is the reference architecture lab DNS server which supports dynamic updates. Ensure the host-name field is correct.

4. Install the mcollective-client package

```
# yum install mcollective-client
```

5. Make the following changes to the `/etc/mcollective/client.cfg` file on each broker host. The username and password are the same as what is specified in `/etc/activemq/activemq.xml`

```
# cp /etc/mcollective/client.cfg{,.orig}  
topicprefix = /topic/  
main_collective = mcollective  
collectives = mcollective  
libdir = /usr/libexec/mcollective  
loglevel = debug  
logfile = /var/log/mcollective-client.log  
# Plugins  
securityprovider = psk  
plugin.psk = unset
```



```
connector = stomp
plugin.stomp.pool.size = 3
plugin.stomp.pool.host1=bsn1.osop.cloud.lab.eng.bos.redhat.com
plugin.stomp.pool.port1=61613
plugin.stomp.pool.user1=mcollective
plugin.stomp.pool.password1=marionette
plugin.stomp.pool.host2=bsn2.osop.cloud.lab.eng.bos.redhat.com
plugin.stomp.pool.port2=61613
plugin.stomp.pool.user2=mcollective
plugin.stomp.pool.password2=marionette
plugin.stomp.pool.host3=bsn3.osop.cloud.lab.eng.bos.redhat.com
plugin.stomp.pool.port3=61613
plugin.stomp.pool.user3=mcollective
plugin.stomp.pool.password3=marionette
# Facts
factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
```

- In addition, the plugin.psk can be changed from unset to a pre-shared key as well to sign the messages. A random key can work here.

6. Install the broker packages on each broker host

```
# yum install openshift-origin-broker openshift-origin-broker-util rubygem-openshift-origin-auth-remote-user rubygem-openshift-origin-msg-broker-mcollective rubygem-openshift-origin-dns-bind
```

7. Configure apache to use REMOTE_USER as the trusted header for authentication.

```
# cp /etc/openshift/plugins.d/openshift-origin-auth-remote-user.conf.example /etc/openshift/plugins.d/openshift-origin-auth-remote-user.conf
```

8. Back up the broker proxy file

```
# cp /etc/httpd/conf.d/000000_openshift_origin_broker_proxy.conf{,.orig}
```

9. Change the virtual host directive

```
# sed -i -e "s/ServerName .*/ServerName `hostname`/" /etc/httpd/conf.d/000000_openshift_origin_broker_proxy.conf
```

10. Ensure the appropriate services are able to start on boot

```
# chkconfig httpd on
# chkconfig network on
# chkconfig ntpd on
# chkconfig sshd on
```

11. Set up the firewall

```
# lokkit --service=ssh
# lokkit --service=https
# lokkit --service=http
```

12. Generate the SSH keys. These keys need to be copied to each brokers */etc/openshift* directory. Only do this step on broker1.

```
# ssh-keygen -t rsa -b 2048
```



```

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
d2:71:e6:de:67:9d:97:0c:35:09:20:3a:7e:09:fc:4c
root@broker1.osop.cloud.lab.eng.bos.redhat.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           . . . . .|
|      . . . . .|
|    =.Eo      +|
|   ..*=.      . .|
|  ..S=.      . .|
|   ... .    0.0|
|    . . 0+0|
|      0  .|
+-----+

```

13. Copy the keys to `/etc/openshift`

```
# cp ~/.ssh/id_rsa* /etc/openshift/
```

14. Copy the keys to broker2

```
# scp /etc/openshift/id_rsa* broker2:/etc/openshift/.
Warning: Permanently added 'broker2,10.16.138.26' (RSA) to the list of known
hosts.
root@broker2's password:
rsync_id_rsa
100% 1675    1.6KB/s   00:00
rsync_id_rsa.pub
100%  428    0.4KB/s   00:00
```

15. Configure the SELinux booleans

```
# setsebool -P httpd_unified=on httpd_can_network_connect=on
httpd_can_network_relay=on httpd_run_stickshift=on
named_write_master_zones=on allow_yppbind=on
```

16. Relabel the appropriate files and directories

```
# fixfiles -R rubygem-passenger restore
# fixfiles -R mod_passenger restore
# restorecon -rv /var/run
# restorecon -rv /usr/share/rubygems/gems/passenger-*
```

17. Enable the mcollective messaging plugin

```
# cp /etc/openshift/plugins.d/openshift-origin-msg-broker-
mcollective.conf.example /etc/openshift/plugins.d/openshift-origin-msg-
broker-mcollective.conf
```




18. Create the bind plugin file `/etc/openshift/plugins.d/openshift-origin-dns-bind.conf` with the following contents on each broker host.

```
BIND_SERVER="10.16.138.14"
BIND_PORT=53
BIND_KEYNAME="osop.cloud.lab.eng.bos.redhat.com"
BIND_KEYVALUE="avzgSzauar4R1Gkfyzzff52Z3ct8mt7m511GM0XINphpq0ABw0lFGhQxHLZqbC
8STM7KPSi9PehVuslfBPtTXnw=="
BIND_ZONE="osop.cloud.lab.eng.bos.redhat.com"
```

The BIND_KEYVALUE is from the earlier BIND setup.

19. Configure the dns-bind SELinux policy module and install it on each broker host.

```
# pushd /usr/share/selinux/packages/rubygem-openshift-origin-dns-bind/ &&
make -f /usr/share/selinux/devel/Makefile ; popd
/usr/share/selinux/packages/rubygem-openshift-origin-dns-bind
/etc/openshift/plugins.d
Compiling targeted dhcpnamedforward module
/usr/bin/checkmodule: loading policy configuration from
tmp/dhcpnamedforward.tmp
/usr/bin/checkmodule: policy configuration loaded
/usr/bin/checkmodule: writing binary representation (version 10) to
tmp/dhcpnamedforward.mod
Creating targeted dhcpnamedforward.pp policy package
rm tmp/dhcpnamedforward.mod tmp/dhcpnamedforward.mod.fc
/etc/openshift/plugins.d

# semodule -i /usr/share/selinux/packages/rubygem-openshift-origin-dns-
bind/dhcpnamedforward.pp
```

20. Configure LDAP authentication in the `/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf` file. This setup is explained more in the following chapter, but this step is assuming that there is a existing Windows 2008 R2 Active Directory server. Refer to **4.4.1 Configure LDAP Authentication** for more information.

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
```

```
# By default the LDAPCacheTTL directive is set to 600 seconds. If you want
to
# effectively disable LDAP caching in mod_ldap, set the directive to 0.
There
# is a performance trade-off, but disabling the cache will make things like
# password changes effective immediately.
# http://httpd.apache.org/docs/2.4/mod/mod_ldap.html
# LDAPCacheTTL 0
```

```
<Location /broker>
    AuthName "OpenShift"
    AuthType Basic
    AuthBasicProvider ldap
    AuthLDAPURL
```



```
"ldap://ldapwin.osop.cloud.lab.eng.bos.redhat.com:389/DC=winldap,DC=osop,DC=
cloud,DC=lab,DC=eng,DC=bos,DC=redhat,DC=com?sAMAccountName?sub?
(objectClass=*)" NONE
  AuthLDAPBindDN "scollier@winldap.osop.cloud.lab.eng.bos.redhat.com"
  AuthLDAPBindPassword "PASSWORD_GOES_HERE"
  require valid-user

  # The node->broker auth is handled in the Ruby code
  BrowserMatchNoCase ^OpenShift passthrough
  Allow from env=passthrough

  # Console traffic will hit the local port. mod_proxy will set this
header automatically.
  SetEnvIf X-Forwarded-For "^$" local_traffic=1
  # Turn the Console output header into the Apache environment variable
for the broker remote-user plugin
  SetEnvIf X-Remote-User "(..*)" REMOTE_USER=$1
  Allow from env=local_traffic

  Order Deny,Allow
  Deny from all
  Satisfy any
</Location>

# The following APIs do not require auth:
<Location /broker/rest/application_templates*>
  Allow from all
</Location>

<Location /broker/rest/cartridges*>
  Allow from all
</Location>

<Location /broker/rest/api*>
  Allow from all
</Location>
```

Change the PASSWORD_GOES_HERE line to the password of the LDAP user.

21. Generate the broker access keys. Only do this on broker1.

```
# openssl genrsa -out /etc/openshift/server_priv.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)

# openssl rsa -in /etc/openshift/server_priv.pem -pubout >
/etc/openshift/server_pub.pem
writing RSA key
```

22. Copy the broker access keys to the other broker

```
# scp /etc/openshift/server_p* broker2:/etc/openshift/.
Warning: Permanently added 'broker2,10.16.138.26' (RSA) to the list of known
```



```
hosts.  
root@broker2's password:  
server_priv.pem  
100% 1679      1.6KB/s   00:00  
server_pub.pem  
100% 451       0.4KB/s   00:00
```

23. Bundle the ruby gems, perform this on all brokers

```
# cd /var/www/openshift/broker  
  
# bundle --local  
Using rake (0.8.7)  
Using abstract (1.0.0)  
Using activesupport (3.0.13)  
Using builder (2.1.2)  
Using i18n (0.5.0)  
Using activemodel (3.0.13)  
Using erubis (2.6.6)  
Using rack (1.3.0)  
Using regin (0.3.7)  
Using rack-mount (0.7.1)  
Using rack-test (0.6.1)  
Using tzinfo (0.3.26)  
Using actionpack (3.0.13)  
Using mime-types (1.16)  
Using polyglot (0.3.1)  
Using tree-top (1.4.10)  
Using mail (2.3.0)  
Using actionmailer (3.0.13)  
Using arel (2.0.9)  
Using activerecord (3.0.13)  
Using activerecord (3.0.13)  
Using bson (1.5.2)  
Using bundler (1.0.21)  
Using diff-lcs (1.1.2)  
Using json (1.4.6)  
Using term-ansicolor (1.0.5)  
Using trollop (1.16.2)  
Using gherkin (2.2.4)  
Using cucumber (0.9.0)  
Using dnsruby (1.53)  
Using mocha (0.9.8)  
Using mongo (1.5.2)  
Using open4 (1.3.0)  
Using openshift-origin-common (1.0.1)  
Using state_machine (1.1.2)  
Using openshift-origin-controller (1.0.7)  
Using openshift-origin-auth-remote-user (1.0.3)  
Using openshift-origin-dns-bind (1.0.1)  
Using stomp (1.1.8)  
Using systemu (1.2.0)  
Using openshift-origin-msg-broker-mcollective (1.0.2)  
Using parseconfig (0.5.2)  
Using rdoc (3.8)
```



```
Using thor (0.14.6)
Using railties (3.0.13)
Using rails (3.0.13)
Using rcov (1.0.0)
Using rest-client (1.6.1)
Using xml-simple (1.0.12)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled
gem is installed.
```

24. Backup the `/etc/openshift/broker.conf` and make the following changes to the file on both broker hosts.

```
# cp /etc/openshift/broker.conf{,.orig}
```

```
# Domain suffix to use for applications (Must match node config)
CLOUD_DOMAIN="osop.cloud.lab.eng.bos.redhat.com"
# Comma separted list of valid gear sizes
VALID_GEAR_SIZES="small,medium"

# Default number of gears to assign to a new user
DEFAULT_MAX_GEARS="100"
# Default gear size for a new gear
DEFAULT_GEAR_SIZE="small"

#Broker datastore configuration
MONGO_REPLICA_SETS=true
# Replica set example: "<host-1>:<port-1> <host-2>:<port-2> ..."
MONGO_HOST_PORT="bsn1.osop.cloud.lab.eng.bos.redhat.com:27017
bsn2.osop.cloud.lab.eng.bos.redhat.com:27017
bsn3.osop.cloud.lab.eng.bos.redhat.com:27017"
MONGO_USER="openshift"
MONGO_PASSWORD="mooo"
MONGO_DB="openshift_broker_dev"

#Enables gear/filesystem resource usage tracking
ENABLE_USAGE_TRACKING_DATASTORE="false"
#Log resource usage information to syslog
ENABLE_USAGE_TRACKING_SYSLOG="false"

#Enable all broker analytics
ENABLE_ANALYTICS="false"

#Enables logging of REST API operations and success/failure
ENABLE_USER_ACTION_LOG="true"
USER_ACTION_LOG_FILE="/var/log/openshift/user_action.log"

AUTH_SALT="ClWqe5zKtEW4CJEMyJzQ"
AUTH_PRIVKEYFILE="/etc/openshift/server_priv.pem"
AUTH_PRIVKEYPASS=""
AUTH_PUBKEYFILE="/etc/openshift/server_pub.pem"
AUTH_RSYNC_KEY_FILE="/etc/openshift/rsync_id_rsa"
```

The `MONGO_PASSWORD` listed in the `/etc/openshift/broker` file is the same one that was used during the MongoDB setup in the earlier section.



- One item to note here is that the MONGO_HOST_PORT variable contains a list of the MongoDB servers that are in the replica set. As of this writing, the current code base is not interpreting the list properly. A Bugzilla⁷ has been opened and the workaround is to perform the following step on each broker host:

```
# sed -i 's/elif/elsif/'
/var/www/openshift/broker/config/environments/production.rb
```

25. Ensure the broker service starts on reboot

```
# chkconfig openshift-broker on
```

26. Reboot the broker

27. Check that each broker is functioning and service web pages with no authentication. Authentication does not work until the nodes are deployed.

```
# curl -Ik https://localhost/broker/rest/api
HTTP/1.1 200 OK
Date: Tue, 27 Nov 2012 16:50:26 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: Phusion Passenger (mod_rails/mod_rack) 3.0.17
X-Runtime: 0.088329
ETag: "6a5924d0f3aedbce5a426b7afeb2f0e3"
X-UA-Compatible: IE=Edge,chrome=1
Cache-Control: max-age=0, private, must-revalidate
Status: 200
Content-Type: application/json; charset=utf-8
Connection: close
```

Check for the status 200 return.

28. Once both brokers are up make sure that both brokers have the same set of ssh keys. This is required when moving applications from one node to another.

```
# scp ~/.ssh/id_rsa* broker2:/root/.ssh/.
Warning: Permanently added 'broker2,10.16.138.26' (RSA) to the list of known
hosts.
root@broker2's password:
id_rsa
100% 1675      1.6KB/s   00:00
id_rsa.pub
100% 428       0.4KB/s   00:00
```

4.4.1 Configure LDAP Authentication

OpenShift Enterprise supports authentication using through the remote-user authentication plugin which relies on the **httpd** service to pass on the authenticated user. The methods of authentication that are supported are basic, LDAP, kerberos, and other industrial strength technologies. Basic authentication is fine for testing in development environments but LDAP or kerberos are the recommended authentication methods for production environments. OpenShift Enterprise uses the Apache module *MOD_AUTHNZ_LDAP* for authenticating to directory servers. Therefore any directory server that *MOD_AUTHNZ_LDAP* supports, OpenShift Enterprise supports. This creates a flexible authentication infrastructure for OpenShift



Enterprise. There are two options for integrating with Active Directory. Either an Active Directory infrastructure exists, or it doesn't and one has to be created. For the purposes of the reference architecture lab, an AD infrastructure was created from scratch. Either point it to a newly installed AD infrastructure or one that has been in the environment.

To configure *MOD_AUTHNZ_LDAP* and Microsoft Windows Server 2008 R2 Standard Active Directory integration, the following needs to take place:

- Deploy or use an existing Microsoft Windows 2008 R2 Standard Domain controller running Active Directory services
- Configure the *openshift-origin-auth-remote-user.conf* file on the broker node for both broker and console access

To Deploy the Microsoft Windows 2008 R2 Standard Domain Controller

- See **Configure Microsoft Windows 2008 R2 Active Directory Server**

4.4.2 Deploy OpenShift Enterprise Console

Install the OpenShift Console. As of the release date for this paper, the OpenShift console is in technical preview. To evaluate the console, perform the following steps. The OpenShift Console is not required to complete the configuration of the infrastructure.

To configure the OpenShift Enterprise Console the following steps need to take place:

- Install the OpenShift Enterprise Console packages
- Configure the console configuration file

1. Install the console packages on both broker hosts

```
# yum install openshift-console
```

2. Configure the */var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf* file to reflect the following contents on both broker hosts

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule ldap_module modules/mod_ldap.so
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
LDAPCacheTTL 0

# Turn the authenticated remote-user into an Apache environment variable for
the console security controller
RewriteEngine On
RewriteCond %{LA-U:REMOTE_USER} (.+)
RewriteRule . - [E=RU:%1]
RequestHeader set X-Remote-User "%{RU}e" env=RU
<Location /console>
    AuthName "OpenShift"
    AuthType Basic
    AuthBasicProvider ldap
    AuthLDAPURL
"ldap://ldapwin.osop.cloud.lab.eng.bos.redhat.com:389/DC=winldap,DC=osop,DC=
cloud,DC=lab,DC=eng,DC=bos,DC=redhat,DC=com?sAMAccountName?sub?"
```



```
(objectClass=*)" NONE
  AuthLDAPBindDN "scollier@winldap.osop.cloud.lab.eng.bos.redhat.com"
  AuthLDAPBindPassword "PASSWORD_HERE"
  require valid-user

  # The node->broker auth is handled in the Ruby code
  BrowserMatch ^Openshift passthrough
  Allow from env=passthrough

  Order Deny,Allow
  Deny from all
  Satisfy any
</Location>
```

Change the PASSWORD_HERE to the password of the LDAP user.

There is an example `/var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user-ldap.conf.sample` file to help get started.

3. Restart the OpenShift Enterprise console

```
# service openshift-console restart
Stopping openshift-console: [FAILED]
/var/www/openshift/console /
Starting openshift-console: [ OK ]
```

4. Ensure the console is enabled after boot

```
# chkconfig openshift-console on
```



5. This step is actually completed after the nodes are installed and configured. Until then, the following screenshot is not seen. Open a browser and point to the OpenShift Console for each broker. After the load balancer is set up, the IP for the virtual server broker.osop.cloud.lab.eng.bos.redhat.com can be used. For example, the following URL's can be used to access the console(s).

- <https://broker1.osop.cloud.lab.eng.bos.redhat.com/console>
- <https://broker2.osop.cloud.lab.eng.bos.redhat.com/console>
- <https://broker.osop.cloud.lab.eng.bos.redhat.com/console>

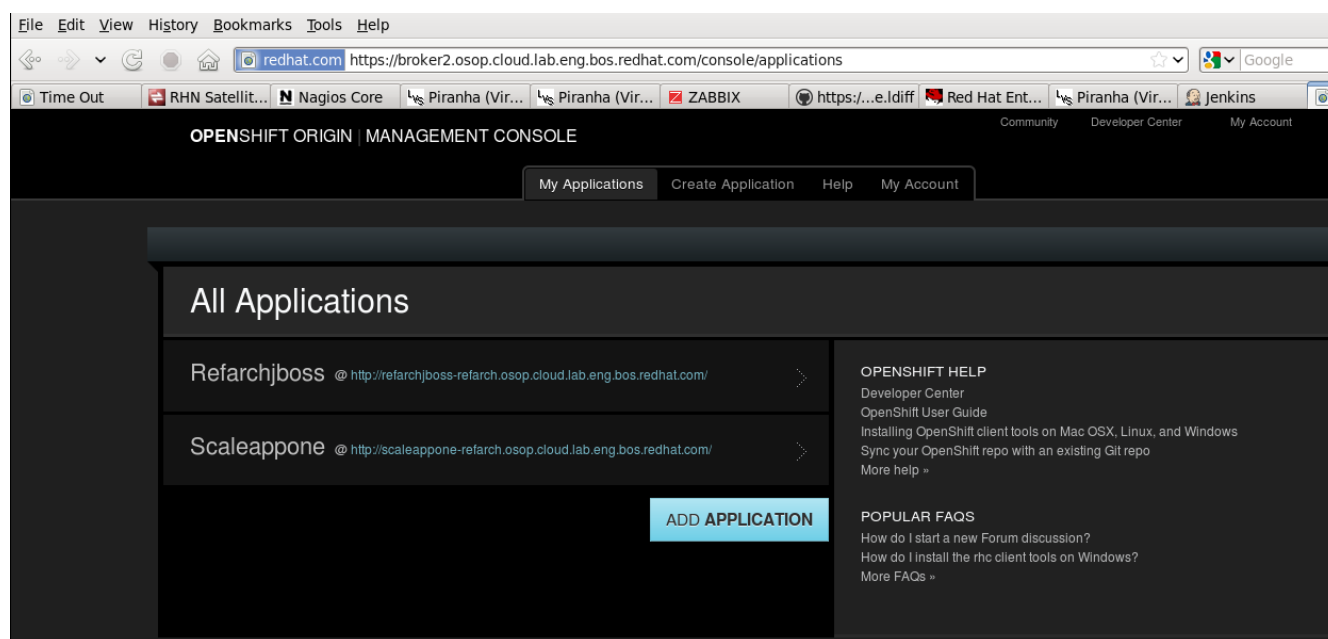


Figure 1: OpenShift Console



4.5 Deploy Nodes

This section demonstrates deploying the node software using the Red Hat Network (RHN) as well as Red Hat's content delivery network (CDN).

To configure a OpenShift Enterprise node the following steps need to take place:

- Ensure the proper OpenShift channels are accessible
- Install the node software
- Configure the node Software

Perform the following tasks on each node

1. Add the infrastructure and JBoss EAP channels to the node

```
# rhn-channel --add \  
--channel rhel-x86_64-server-6-osop-1-node \  
--channel rhel-x86_64-server-6-osop-1-jbosseap \  
--channel jbappplatform-6-x86_64-server-6-rpm \  
--channel jb-ews-2-x86_64-server-6-rpm  
Username: admin  
Password:
```

2. Configure DHCP properly by making sure the following entries are in the `/etc/dhcp/dhclient-eth0.conf` file. Perform this on each node.

```
send vendor-class-identifier "anaconda-Linux 2.6.32-279.el6.x86_64 x86_64";  
timeout 45;  
prepend domain-name-servers 10.16.138.14;  
supersede host-name "node1";  
supersede domain-name "osop.cloud.lab.eng.bos.redhat.com";
```

Make sure the host-name is correct on each node.

3. Install the mcollective packages on the node

```
# yum install openshift-origin-msg-node-mcollective
```

4. Backup the `/etc/mcollective/server.cfg` file

```
# cp /etc/mcollective/server.cfg{,.orig}
```

5. Configure the mcollective `/etc/mcollective/server.cfg` configuration file and copy it to all node hosts

```
topicprefix = /topic/  
main_collective = mcollective  
collectives = mcollective  
libdir = /usr/libexec/mcollective  
logfile = /var/log/mcollective.log  
loglevel = debug  
daemonize = 1  
direct_addressing = n  
registerinterval = 30  
# Plugins  
securityprovider = psk
```



```
plugin.psk = unset
connector = stomp
plugin.stomp.pool.size = 3
plugin.stomp.pool.host1=bsn1.osop.cloud.lab.eng.bos.redhat.com
plugin.stomp.pool.port1=61613
plugin.stomp.pool.user1=mcollective
plugin.stomp.pool.password1=marionette
plugin.stomp.pool.host2=bsn2.osop.cloud.lab.eng.bos.redhat.com
plugin.stomp.pool.port2=61613
plugin.stomp.pool.user2=mcollective
plugin.stomp.pool.password2=marionette
plugin.stomp.pool.host3=bsn3.osop.cloud.lab.eng.bos.redhat.com
plugin.stomp.pool.port3=61613
plugin.stomp.pool.user3=mcollective
plugin.stomp.pool.password3=marionette
# Facts
factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
```

6. Ensure that mcollective starts on boot and then start the service

```
# chkconfig mcollective on

# service mcollective restart
Shutting down mcollective:
Starting mcollective: [ OK ]
```

7. On the broker, test mcollective communications

```
# mco ping
node1.osop.cloud.lab.eng.bos.redhat.com  time=71.56 ms

---- ping statistics ----
1 replies max: 71.56 min: 71.56 avg: 71.56
```

8. Install the OpenShift Enterprise core packages

```
# yum install rubygem-openshift-origin-node rubygem-passenger-native
openshift-origin-port-proxy openshift-origin-node-util
```

9. Install the OpenShift Enterprise cartridges

```
# yum install openshift-origin-cartridge-*
```

10. Configure the firewall and ensure the services start when the system boots

```
# lokkit --service=ssh
# lokkit --service=https
# lokkit --service=http
# chkconfig httpd on
# chkconfig network on
# chkconfig sshd on
```

11. Configure PAM

```
# sed -i -e 's|pam_selinux|pam_openshift|g' /etc/pam.d/sshd
```

```
# for f in "runuser" "runuser-l" "sshd" "su" "system-auth-ac"; \
do t="/etc/pam.d/$f"; \
if ! grep -q "pam_namespace.so" "$t"; \
then echo -e "session\t\t\trequired\t\tpam_namespace.so no_unmount_on_close" >>
"$t" ; \
fi; \
done;
```

12. Configure cgroups

```
# cp /etc/cgconfig.conf{,.orig}

# cp -f /usr/share/doc/rubygem-openshift-origin-node-*/cgconfig.conf
/etc/cgconfig.conf
cp: overwrite `/etc/cgconfig.conf'? Y

# restorecon -v /etc/cgconfig.conf

# restorecon -v /cgroup

# chkconfig cgconfig on

# chkconfig cgred on

# chkconfig openshift-cgroups on

# service cgconfig restart
Stopping cgconfig service: [ OK ]
Starting cgconfig service: [ OK ]

# service cgred restart
Stopping CGroup Rules Engine Daemon... [ OK ]
Starting CGroup Rules Engine Daemon: [ OK ]

# service openshift-cgroups start
Initializing Openshift guest control groups:
[ OK ]Openshift cgroups initialized
```

13. Configure quotas, assuming `/var/lib/openshift` is on `/`. Add `usrquota` option to `/etc/fstab`

```
# grep myvg-rootvol /etc/fstab
/dev/mapper/myvg-rootvol /                                ext4      usrquota      1 1

# mount -o remount /

# quotacheck -cmu /

# repquota -a
*** Report for user quotas on device /dev/mapper/myvg-rootvol
Block grace time: 7days; Inode grace time: 7days

      Block limits                File limits
User      used      soft      hard      grace      used      soft      hard      grace
-----
root      -- 3197468           0           0          113763           0           0
```



daemon	--	8	0	0	3	0	0
lp	--	8	0	0	2	0	0
abrt	--	24	0	0	4	0	0
haldaemon	--	8	0	0	2	0	0
ntp	--	12	0	0	3	0	0
postfix	--	60	0	0	38	0	0
ovirtagent	--	12	0	0	3	0	0
rpc	--	4	0	0	2	0	0
avahi	--	8	0	0	3	0	0
rpcuser	--	16	0	0	5	0	0
apache	--	12	0	0	3	0	0
haproxy	--	4	0	0	1	0	0
postgres	--	16	0	0	4	0	0
mysql	--	8	0	0	3	0	0
jboss	--	6756	0	0	1458	0	0
tomcat	--	284	0	0	24	0	0
jenkins	--	12	0	0	3	0	0

14. Configure SELinux

```
# setsebool -P httpd_unified=on httpd_can_network_connect=on
httpd_can_network_relay=on httpd_read_user_content=on
httpd_enable_homedirs=on httpd_run_stickshift=on allow_polyinstantiation=on

# fixfiles -R rubygem-passenger restore
# fixfiles -R mod_passenger restore
# restorecon -rv /var/run
# restorecon -rv /usr/share/rubygems/gems/passenger-*
# restorecon -rv /usr/sbin/mcollectived /var/log/mcollective.log
/var/run/mcollectived.pid
# restorecon -rv /var/lib/openshift /etc/openshift/node.conf
/etc/httpd/conf.d/openshift
```

15. Configure the system control settings

```
# echo -e "kernel.sem = 250 32000 32 4096\nnet.ipv4.ip_local_port_range =
15000 35530\nnet.netfilter.nf_conntrack_max = 1048576" >> /etc/sysctl.conf

# sysctl -p /etc/sysctl.conf
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
error: "net.bridge.bridge-nf-call-ip6tables" is an unknown key
error: "net.bridge.bridge-nf-call-iptables" is an unknown key
error: "net.bridge.bridge-nf-call-arptables" is an unknown key
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
kernel.sem = 250 32000 32 4096
net.ipv4.ip_local_port_range = 15000 35530
net.netfilter.nf_conntrack_max = 1048576
```



16. Configure SSH

```
# echo "AcceptEnv GIT_SSH" >> /etc/ssh/sshd_config  
  
# perl -p -i -e "s/^#MaxSessions .*/MaxSessions 40/" /etc/ssh/sshd_config  
# perl -p -i -e "s/^#MaxStartups .*/MaxStartups 40/" /etc/ssh/sshd_config
```

17. Configure the port proxy

```
# lokkit --port=35531-65535:tcp  
# chkconfig openshift-port-proxy on  
# service openshift-port-proxy start  
Starting openshift-port-proxy: [ OK ]  
# chkconfig openshift-gears on
```

18. Configure the `/etc/openshift/node.conf` file. The four entries changed from default are in bold below. The `BROKER_HOST` variable represents the DNS name that is associated to the virtual IP address hosted by the load balancers. If there was only one broker in the solution, that broker's IP hostname is listed there.

```
# These should not be left at default values, even for a demo.  
# "PUBLIC" networking values are ones that end-users should be able to reach.  
PUBLIC_HOSTNAME="node1.osop.cloud.lab.eng.bos.redhat.com" # The node  
host's public hostname  
PUBLIC_IP="10.16.138.29" # The node  
host's public IP address  
BROKER_HOST="broker.osop.cloud.lab.eng.bos.redhat.com"  
# IP or DNS name of broker server for REST API  
  
# Usually (unless in a demo) this should be changed to the domain for your  
installation:  
CLOUD_DOMAIN="osop.cloud.lab.eng.bos.redhat.com"  
# Domain suffix to use for applications (Must match broker config)  
  
# You may want these, depending on the complexity of your networking:  
# EXTERNAL_ETH_DEV='eth0' # Specify the  
internet facing public ethernet device  
# INTERNAL_ETH_DEV='eth1' # Specify the  
internal cluster facing ethernet device  
  
# Generally the following should not be changed:  
GEAR_BASE_DIR="/var/lib/openshift" # gear root  
directory  
GEAR_SKEL_DIR="/etc/openshift/skel" # skel files to  
use when building a gear  
GEAR_SHELL="/usr/bin/oo-trap-user" # shell to use  
for the gear  
GEAR_GECOS="OpenShift guest" # Gecos  
information to populate for the gear user  
GEAR_MIN_UID=500 # Lower bound of  
UID used to create gears  
GEAR_MAX_UID=6500 # Upper bound of  
UID used to create gears  
OPENSIFT_NODE_PLUGINS="openshift-origin-node/plugins/unix_user_observer"
```



```
# Extensions to load when customize/observe openshift-origin-node models
CARTRIDGE_BASE_PATH="/usr/libexec/openshift/cartridges" # Locations where
cartridges are installed
LAST_ACCESS_DIR="/var/lib/openshift/.last_access" # Location to
maintain last accessed time for gears
APACHE_ACCESS_LOG="/var/log/httpd/access_log" # Location of
httpd for node
PROXY_MIN_PORT_NUM=35531 # Lower bound of
port numbers used to proxy ports externally
PROXY_PORTS_PER_GEAR=5 # Number of
proxy ports available per gear
CREATE_APP_SYMLINKS=0 # If set to 1,
creates gear-name symlinks to the UUID directories (debugging only)
OPENSIFT_HTTP_CONF_DIR="/etc/httpd/conf.d/openshift"
```

19. Ensure the broker hosts can ssh into the nodes without requiring a password.
Perform these step from broker1.

```
# for NODES in node1 node2 node3; do ssh-copy-id -i ~/.ssh/id_rsa.pub
$NODES; done
```

Warning: Permanently added 'node1,10.16.138.29' (RSA) to the list of known hosts.

root@node1's password:

Now try logging into the machine, with "ssh 'node1'", and check in:

```
ssh/authorized_keys
```

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'node2,10.16.138.30' (RSA) to the list of known hosts.

root@node2's password:

Now try logging into the machine, with "ssh 'node2'", and check in:

```
ssh/authorized_keys
```

to make sure we haven't added extra keys that you weren't expecting.

Warning: Permanently added 'node3,10.16.138.57' (RSA) to the list of known hosts.

root@node3's password:

Now try logging into the machine, with "ssh 'node3'", and check in:

```
ssh/authorized_keys
```

to make sure we haven't added extra keys that you weren't expecting.

20. Back to the nodes, update the facter database

```
# /etc/cron.minutely/openshift-facts
```



4.5.1 Handling SSH Host Keys

There are times when the administrator of the OpenShift Enterprise solution needs to move a gear to another host. A **gear** is a container for an application. When an administrator makes that move, the developer encounters a **SSH** man-in-the-middle message that something nasty has happened on the system. List here are the steps that happen and some options on dealing with it.

1. Administrator deploys OpenShift Enterprise solution.
2. Developer creates an account on the OpenShift Enterprise solution with **rhc setup** (covered later on).
 - (a) Part of this process is uploading the developers **SSH** public key.
3. Developer creates an application and the application gets deployed to node1.
 - (a) As part of **rhc app create** the application's git repository is cloned via SSH, using the application's hostname (e.g. app-domain.example.com) which is a CNAME to the node host it resides on.
 - (b) Developer verifies the host key (either manually or according to ssh config) and it is added to the `~/.ssh/known_hosts` file to verify later visits.
4. The administrator moves the **gear** to node2 so maintenance can be performed on node1. This causes the application CNAME to point to node2.
5. Developer connects to the **gear** again (either with git operation or directly via SSH), but this time because the host ID for the application has changed from what is stored in the developer's `known_hosts` file, ssh sounds an alarm.

The simplest way to handle this situation is to ensure that all nodes have the same SSH host keys.

1. On each node, make a backup of all `/etc/ssh/ssh_host_*` files.

```
# cd /etc/ssh/  
# mkdir hostkeybackup  
# cp ssh_host_* hostkeybackup/.
```

2. On node1, copy the `/etc/ssh/ssh_host_*` files to the other nodes.

```
# scp /etc/ssh/ssh_host_* node2:/etc/ssh/.  
# scp /etc/ssh/ssh_host_* node3:/etc/ssh/.
```

3. Restart **SSH** on each node and reboot

```
# service sshd restart  
# reboot
```



4.6 Deploy Red Hat Load Balancer

This reference architecture shows how to set up a two node Red Hat Load Balancer with the Load Balancer Add-On components. The load balancer is placed in front of the broker nodes and load balances all traffic going over port 443 to the two broker nodes. The load balancer is configured as an active-passive cluster. The networking topology used in this reference architecture is Direct Routing as shown in **Figure 4.6.1:Load Balancer**. This means that the traffic goes through the load balancer to a broker and then returns directly to the requesting client bypassing the load balancer. This improves performance. In addition to the Direct Routing networking setup, the load balancer uses the least-connections load balancing algorithm with persistence enabled. This helps ensure that the client connects to the broker with the least connections and stays on the same broker until the persistence timeout has expired.

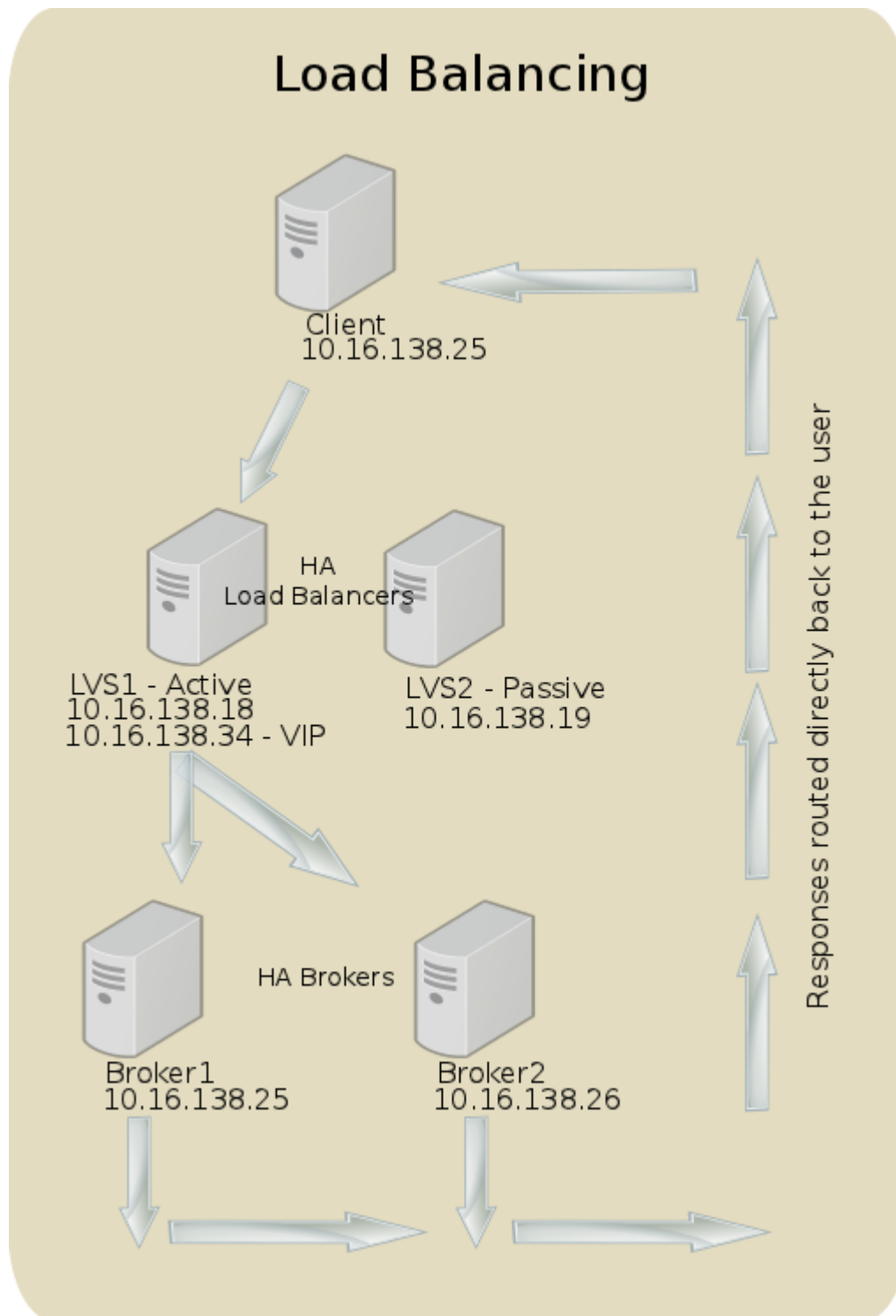


Figure 4.6.1:Load Balancer

To configure the load balancer the following steps need to take place:

- Deploy two Red Hat Enterprise Linux 6 servers and associate the *rhel-x86_64-server-*



lb-6 channel

- Install and configure the load balancer add-on software on the load balancer nodes
- Configure the brokers to allow direct network traffic
- Configure the nodes to point to the virtual server for traffic

Perform the following tasks on each load balancer node to deploy Red Hat Enterprise Linux 6

- Deploy Red Hat Enterprise Linux. Follow the Red Hat Enterprise Linux Installation Guide⁸ for more details.

Perform the following tasks on each load balancer node to install and configure the load balancer add-on software.

1. Installation

(a) Add the channel to install piranha and set the piranha gui password

```
# rhn-channel --add --channel=rhel-x86_64-server-lb-6
```

```
Username: admin
```

```
Password:
```

```
# yum install piranha
```

```
# piranha-passwd
```

```
New Password:
```

```
Verify:
```

```
Adding password for user piranha
```

(b) Start the piranha GUI and ensure the load balancer software starts on boot

```
# service piranha-gui restart
```

```
Shutting down piranha-gui:
```

```
[ OK ]
```

```
Starting piranha-gui:
```

```
[ OK ]
```

```
# chkconfig piranha-gui on
```

For more information on configuring the Load Balancer Add-on for Red Hat see the Load Balancer guide⁹ for more information

(c) Enable packet forwarding on both load balancers by issuing the following command, and make it permanent by editing the */etc/sysctl.conf* file

```
# sysctl -w net.ipv4.ip_forward=1
```

(d) Configure firewall to allow ports 3636,539 and restart **httpd**

```
# lokkit --port=3636:tcp
```

```
# lokkit --port=539:udp
```

```
# service httpd restart
```



2. Configuration. This configuration only needs to happen on one of the load balancers, then copy the `/etc/sysconfig/ha/lvs.cf` file to the other LVS host.

(a) Log into the load balancer node that is the primary LB. To access the load balancer nodes, the following URL's can be used:

<http://lvs1.osop.cloud.lab.eng.bos.redhat.com:3636> - primary

<http://lvs2.osop.cloud.lab.eng.bos.redhat.com:3636> - secondary

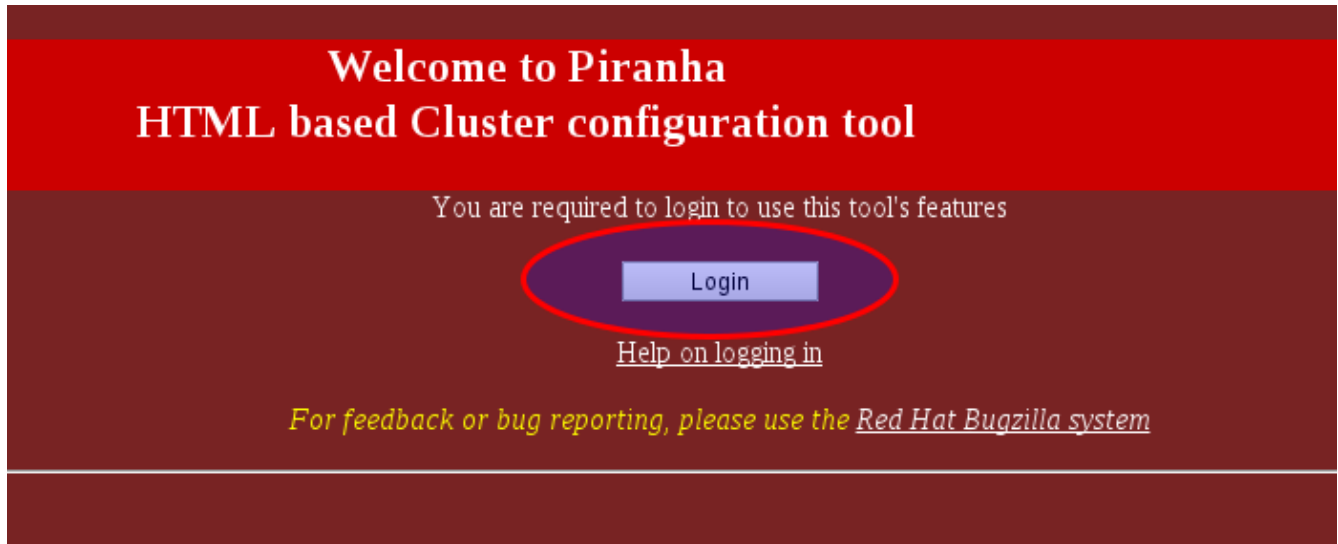


Figure 4.6.2: Piranha Login

The server 10.16.138.39:3636 requires a username and password. The server says: access to the piranha web GUI.

User Name:

Password:

Figure 4.6.3: Password



- (b) Ensure that the network type is direct and that the **Primary server public IP** is set.
- Click on **GLOBAL SETTINGS** and make sure **Current type is: direct**.

CONTROL/MONITORING **GLOBAL SETTINGS**

ENVIRONMENT

Primary server public IP:

Primary server private IP:

(May be blank)

TCP timeout (seconds):

TCP FIN timeout (seconds):

UDP timeout (seconds):

Use network type:
(Current type is **direct**)

-- Click here to apply changes on this page

Figure 4.6.4: Direct Network

- (c) Enable redundancy between the two load balancer nodes
- Click **REDUNDANCY** on the top navigation bar

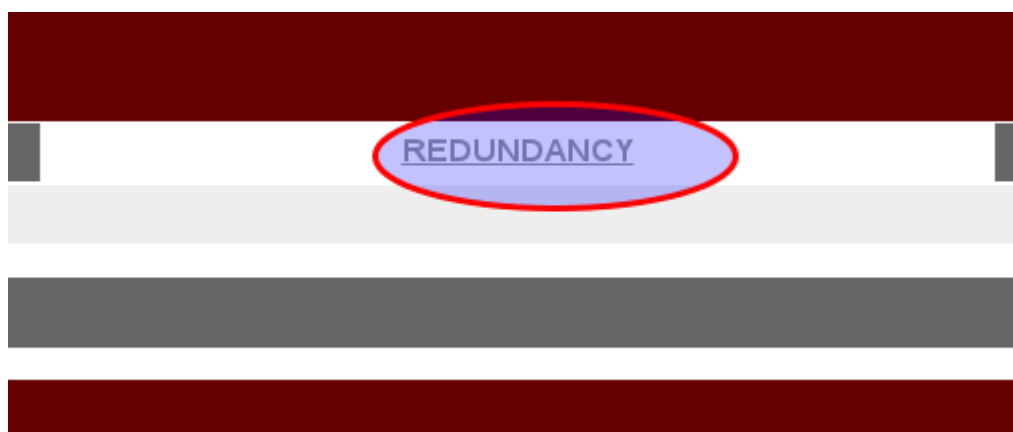


Figure 4.6.5: Piranha Redundancy



A. Click **ENABLE**

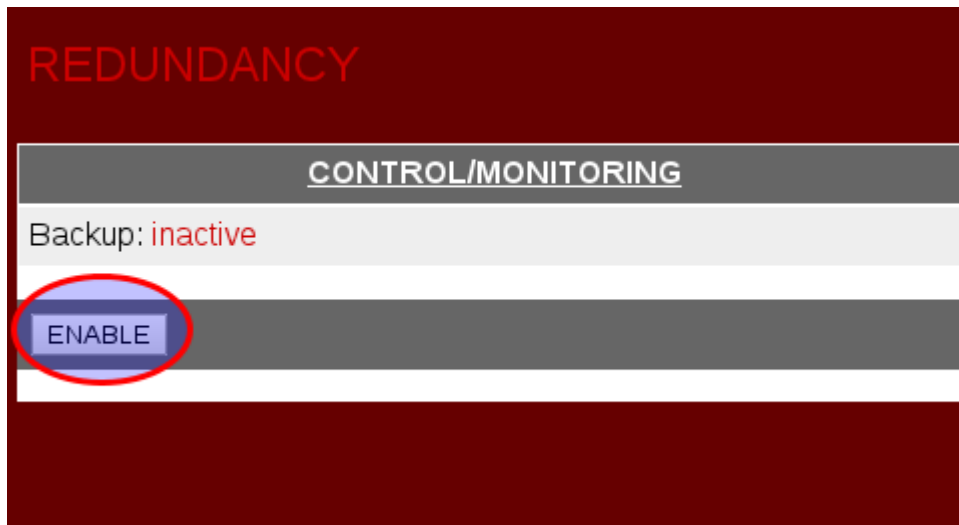


Figure 4.6.6: Enable Redundancy



- B. Input the **Redundant server public IP** address and enable **Monitor NIC links for failures** and **Use sync daemon**. Also, add *eth0* for **Sync daemon interface** and provide a **Sync daemon ID** of 0 and click **ACCEPT**

REDUNDANCY

CONTROL/MONITORING

Backup: active

Redundant server public IP: 10.16.138.19

Heartbeat interval (seconds): 6

Assume dead after (seconds): 18

Heartbeat runs on port: 539

Monitor NIC links for failures: ☒

Use sync daemon: ☒

Sync daemon interface: eth0

Sync daemon ID: 0

ACCEPT -- Click here to apply changes to this page

Figure 4.6.7: Redundant IP

- (d) Add a virtual Server, click on **VIRTUAL SERVERS**



Figure 4.6.8: Virtual Servers



i. Click **ADD**

VIRTUAL SERVERS

CONTROL/MONITORING

	STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERFACE
<input checked="" type="radio"/>							
<input type="radio"/>							

☒ **ADD** ☐ DELETE ☐ EDIT ☐ (DE)ACTIVATE

Note: Use the radio button on the side to select which virtual service you wish



Figure 4.6.9: Control / Monitoring

ii. Select the radio button beside the newly created virtual server and click **EDIT**

VIRTUAL SERVERS

CONTROL/MONITORING **GLOBAL SETTINGS**

	STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERFACE
<input checked="" type="radio"/>	down	[server_name]	0.0.0.0		80	tcp	
<input type="radio"/>							

☐ **ADD** ☐ DELETE ☒ **EDIT** ☐ (DE)ACTIVATE

Note: Use the radio button on the side to select which virtual service you wish to edit before selecting 'EDIT' or 'DELETE'

Figure 4.6.10: Add Server



- iii. Provide a **Name**, **Application port** of 443, **Virtual IP Address**, **Virtual Network Mask** and choose a **Scheduling** method of **Least-connection** with a **Persistence** of 300 seconds and click **Accept**.

CONTROL/MONITORING		GLOBAL SETTINGS	
EDIT: VIRTUAL SERVER REAL SERVER MONITORING SCRIPTS			
Name:	<input type="text" value="BrokerHTTPS"/>		
Application port:	<input type="text" value="443"/>		
Protocol:	<input type="text" value="tcp"/>		
Virtual IP Address:	<input type="text" value="10.16.138.34"/>		
Virtual IP Network Mask:	<input type="text" value="255.255.248.0"/>		
Sorry Server:	<input type="text"/>		
Firewall Mark:	<input type="text"/>		
Device:	<input type="text" value="eth0:1"/>		
Re-entry Time:	<input type="text" value="15"/>		
Service timeout:	<input type="text" value="6"/>		
Quiesce server:	<input type="radio"/> Yes <input checked="" type="radio"/> No		
Load monitoring tool:	<input type="text" value="none"/>		
Scheduling:	<input type="text" value="Round robin"/>		
Persistence:	<input type="text" value="300"/>		
Persistence Network Mask:	<input type="text" value="Unused"/>		
<input type="button" value="ACCEPT"/> -- Click here to apply changes to this page			

Figure 4.6.11: Virtual Server

- iv. Add a **REAL SERVER**

CONTROL/MONITORING		GLOBAL SETTINGS	
EDIT: VIRTUAL SERVER REAL SERVER MONITORING SCRIPTS			

Figure 4.6.12: Real Server



v. Click **EDIT**

STATUS	NAME
<input checked="" type="radio"/> Down <input type="button" value="ADD"/> <input type="button" value="DELETE"/> <input type="button" value="EDIT"/> <input type="button" value="(DE)ACTIVATE"/>	[unnamed]

Figure 4.6.13: Edit Real Server

vi. Provide a **Name** (doesn't have to be hostname of broker), **Address** (IP address of eth0 on the first broker) and click **ACCEPT**

Name:

Address:

Port: (Leave blank to default to Virtual Server's Application Port)

Weight:

Figure 4.6.14: Provide Real Server Parameters



- vii. Click **REAL SERVER** again to add the second broker
 - A. Repeat steps **Add a REAL SERVER** through **Click REAL SERVER again to add the second broker** to add the second broker
- viii. Activate the virtual server and real servers
 - A. Click **VIRTUAL SERVERS** and click **(DE)ACTIVATE** to activate the virtual server

VIRTUAL SERVERS

CONTROL/MONITORING						GLOBAL SETTINGS	
	STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERFACE
<input checked="" type="radio"/>	down	brokerVS	10.16.138.35		443	tcp	

Note: Use the radio button on the side to select which virtual service you wish to edit before selecting 'EDIT' or 'DELETE'

Figure 4.6.15:

VIRTUAL SERVERS

CONTROL/MONITORING						GLOBAL SETTINGS	
	STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERFACE
<input checked="" type="radio"/>	up	brokerVS	10.16.138.35		443	tcp	

Note: Use the radio button on the side to select which virtual service you wish to edit before selecting 'EDIT' or 'DELETE'

Figure 4.6.16: Activated Virtual Server



B. Click **EDIT** and then **REAL SERVER** and **(DE)ACTIVATE** by each real server

ix. Next, ssh to each lvs node and enable the services on boot

```
# chkconfig pulse on
# chkconfig httpd on
```

3. Configure monitoring scripts in the pirahna GUI

(a) Click on **VIRTUAL SERVERS**, edit the virtual server and click **MONITORING SCRIPTS**

CONTROL/MONITORING		GLOBAL SETTINGS	
EDIT: VIRTUAL SERVER REAL SERVER MONITORING SCRIPTS			
	STATUS	NAME	ADDRESS
<input checked="" type="radio"/>	up	broker1	10.16.138.25
<input type="radio"/>	up	broker2	10.16.138.26
<input type="button" value="ADD"/> <input type="button" value="DELETE"/> <input type="button" value="EDIT"/> <input type="button" value="(DE)ACTIVATE"/>			

Figure 4.6.17: Monitoring Scripts



(b) Change the **Sending Program:** to `/usr/local/bin/check_web_https.sh %h`
The `check_web_https.sh` script is copied to the LVS file system in the next section. The `%h` parameter tells the load balancer to run the `check_web_https.sh` script against each registered “real” server.

(c) Make sure there is no entry in the **Send:** field

(d) Change the **Expect:** field to OK and click **ACCEPT**

EDIT: [VIRTUAL SERVER](#) | [REAL SERVER](#) | [MONITORING SCRIPTS](#)

	Current text	Replacement text
Sending Program:	<code>/usr/bin/check_web_https.sh %h</code>	<code>/usr/local/bin/check_web_https.sh %h</code>
Send:		
Expect:	<code>"OK"</code>	<code>OK</code>

☐ Treat expect string as a regular expression

Please note: You may either use the simple send/expect mechanism built into piranha or a custom monitoring script (send program). The send program should output a string matching the the expect string. If the argument `%h` is used in the send program to be checked.

ACCEPT

Figure 4.6.18: Monitoring Scripts Configuration

4. Copy the `/etc/sysconfig/ha/lvs.cf` file to the other LVS host.

```
# scp /etc/sysconfig/ha/lvs.cf lvs2:/etc/sysconfig/ha/lvs.cf
Warning: Permanently added 'lvs2,10.16.138.19' (RSA) to the list of known
hosts.
root@lvs2's password:
lvs.cf
100% 823    0.8KB/s   00:00
```

5. Add the monitoring scripts to both of the LVS servers.

(a) Create a `/usr/local/bin/check_web_https.sh` script with the following contents -

```
#!/bin/bash

TEST=`curl -Ik https://$1/broker/rest/api 2> /dev/null | grep -c "Status:
200"`
if [ "$TEST" == "1" ]; then
    echo "OK"
else
    echo "FAIL"
fi
```



This script uses curl to check for a status of “200” from both of the brokers. Next the script passes that results to the “if” conditional and returns “OK” or “FAIL”. If the script returns “OK”, then LVS keeps the broker in the available pool of servers. Otherwise, it removes the server and only directs traffic to the other nodes that pass with “OK”. Upon failure of a broker, messages similar to the following will show up in /var/log/messages of the active LVS host.

```
Dec 12 10:09:49 lvs1 nanny[24218]: Trouble. Received results are not what we
expected from (10.16.138.26:443)
Dec 12 10:09:49 lvs1 nanny[24218]: [inactive] shutting down 10.16.138.26:443
due to connection failure
```

6. Make the script executable on both LVS hosts

```
# chmod +x /usr/local/bin/check_web_https.sh
```

7. Start the pulse service on both LVS hosts

```
# service pulse restart
```

8. Confirm the ipvsadm routing table is working properly, the following shows on the primary LVS host

```
# ipvsadm
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  broker.osop.cloud.lab.eng.bo rr persistent 300
  -> broker1.osop.cloud.lab.eng.b Route    1      0          0
  -> broker2.osop.cloud.lab.eng.b Route    1      0          0
```

Perform the following tasks on each broker to allow for direct network traffic

1. Install *arptables_jf*

```
# yum install arptables_jf
Loaded plugins: product-id, rhnplugin, security, subscription-manager
Updating certificate-based repositories.
Unable to read consumer identity
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package arptables_jf.x86_64 0:0.0.8-20.el6 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
=====
Package                        Arch          Version
Repository                    Size
=====
Installing:
  arptables_jf                 x86_64        0.0.8-20.el6
rhel-x86_64-server-6          58 k
```

Transaction Summary



```
=====
Install          1 Package(s)
```

```
Total download size: 58 k
```

```
Installed size: 0
```

```
Is this ok [y/N]: y
```

2. Configure the ARP tables to drop any ARP requests coming in for the virtual IP.

```
# arptables -A IN -d 10.16.138.34 -j DROP
```

3. Configure the ARP tables to change any outgoing ARP responses which might otherwise contain the virtual IP so that they contain the real IP of the server instead (the broker currently being worked on)

```
# arptables -A OUT -s 10.16.138.34 -j mangle --mangle-ip-s 10.16.138.25
```

Ensure that the mangle-ip-s address is the IP address of the broker that the command is executed on.

4. Confirm that the ARP entries are correct on each broker

```
# arptables -n --list --line-numbers
```

```
Chain IN (policy ACCEPT)
```

num	target	source-ip	destination-ip	source-hw
1	DROP	0.0.0.0/0	10.16.138.34	00/00
00/00		any	0000/0000	0000/0000

```
Chain OUT (policy ACCEPT)
```

num	target	source-ip	destination-ip	source-hw
1	mangle	10.16.138.34	0.0.0.0/0	00/00
00/00		any	0000/0000	0000/0000
				--mangle-ip-s
				10.16.138.25

```
Chain FORWARD (policy ACCEPT)
```

num	target	source-ip	destination-ip	source-hw

5. Save the ARP entries on each broker

```
# service arptables_jf save
```

```
Saving current rules to /etc/sysconfig/arptables: [ OK ]
```

6. Assign the virtual IP to each broker

```
# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0:0
```

```
# sed -i 's/eth0/eth0:0/' /etc/sysconfig/network-scripts/ifcfg-eth0:0
```

```
# sed -i 's/dhcp/static/' /etc/sysconfig/network-scripts/ifcfg-eth0:0
```

```
# sed -i 's/HW/#HW/' /etc/sysconfig/network-scripts/ifcfg-eth0:0
```



```
# echo "IPADDR=10.16.138.34" >> /etc/sysconfig/network-scripts/ifcfg-eth0:0

# echo "NETMASK=255.255.248.0" >> /etc/sysconfig/network-scripts/ifcfg-eth0:0

# cat /etc/sysconfig/network-scripts/ifcfg-eth0:0
DEVICE=eth0:0
#HWADDR=00:1A:4A:10:8A:D4
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.16.138.34
NETMASK=255.255.248.0
```

7. Bring the interface up

If the following message is encountered:

```
# ifup eth0:0
Error, some other host already uses address 10.16.138.34.
```

Stop pulse on both of the load balancers and bring up the interface, then restart the pulse service.

```
# ip a s eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:1a:4a:10:8a:d1 brd ff:ff:ff:ff:ff:ff
    inet 10.16.138.25/21 brd 10.16.143.255 scope global eth0
    inet 10.16.138.34/21 brd 10.16.143.255 scope global secondary eth0:0
    inet6 fe80::21a:4aff:fe10:8ad1/64 scope link
        valid_lft forever preferred_lft forever
```

8. Perform the following tasks on each node to point to the virtual IP address for traffic

Edit the `/etc/openshift/node.conf` file and adjust the `BROKER_HOST` attribute to reflect the virtual DNS name that the load balancers forward traffic to

```
# grep -i broker_host /etc/openshift/node.conf
BROKER_HOST="broker.osop.cloud.lab.eng.bos.redhat.com"
```

9. From either broker host, test traffic to the virtual IP / DNS address

```
# curl -k --user "scollier:PASSWORD" https://broker/broker/rest/domains
{"status":"ok","supported_api_versions":
[1.0,1.1,1.2],"version":"1.2","data":[],"type":"domains","messages":[]}
```

Where ["https://broker/broker/rest/domains"](https://broker/broker/rest/domains) is the URL that points to the DNS name associated with the Virtual IP address.



5 Operational Management

This section shows how to perform some administrative tasks on the OpenShift Enterprise environment. The users involved are:

- Developer
- Administrator

The developer is a consumer of OpenShift resources who creates applications and uses the git version control system to deploy the applications onto OpenShift. The Administrator is the person who installs configures and troubleshoots the OpenShift infrastructure. Many times the two roles can overlap, but for the purposes of this reference architecture there is a distinction. This section puts it all together by showing how to set up a client station, application creation, moving applications between nodes and other administrative tasks.

5.1 Client Tools Configuration

There are three ways to interact with a OpenShift Enterprise broker, the first way is via the REST API, second is via the web based OpenShift console and finally is the CLI. At this time the OpenShift console is in technical preview for this release so the CLI is discussed and demonstrated.

5.1.1 Installation

In order to install the client tools the user needs administrative permissions. This task is performed by a administrator with root privileges.

The following tasks need to be performed to ensure client CLI connectivity:

- Install client packages
- Configure the client

Install the client packages:

1. Add the RHN channel

```
# rhn-channel --add --channel rhel-x86_64-server-6-osop-1-rhc
Username: admin
Password:
```

2. Install the client package

```
# yum install rhc
Loaded plugins: product-id, rhnplugin, security, subscription-manager
Updating certificate-based repositories.
Unable to read consumer identity
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package rhc.noarch 0:1.0.6-1.el6op will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```




```
=====
Package           Arch           Version
Repository         Size
=====
Installing:
  rhc              noarch        1.0.6-1.el6op
Client             730 k

Transaction Summary
=====
Install           1 Package(s)

Total download size: 730 k
Installed size: 2.4 M
Is this ok [y/N]: y
Downloading Packages:
rhc-1.0.6-1.el6op.noarch.rpm
| 730 kB      00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : rhc-1.0.6-1.el6op.noarch
1/1
Installed products updated.
  Verifying  : rhc-1.0.6-1.el6op.noarch
1/1

Installed:
  rhc.noarch 0:1.0.6-1.el6op

Complete!
```

Configure the client DHCP settings (Administrator):

1. Configure DHCP properly by making sure the following entries are in the `/etc/dhcp/dhclient-eth0.conf` file

```
send vendor-class-identifier "anaconda-Linux 2.6.32-279.el6.x86_64 x86_64";
timeout 45;
prepend domain-name-servers 10.16.138.14;
supersede host-name "client";
supersede domain-name "osop.cloud.lab.eng.bos.redhat.com";
```

Configure the Client (Developer):

1. Configure the `LIBRA_SERVER` environment variable. For this example, use the DNS name of the VIP that is assigned to the load balancers.

```
$ export LIBRA_SERVER=broker.osop.cloud.lab.eng.bos.redhat.com
```

2. Install the java platform – this is used for JBoss Developer Studio later

```
# yum groupinstall "Java Platform"
```

3. Run the client setup and create a namespace



```
$ # rhc setup
```

```
Starting Interactive Setup for OpenShift's command line interface
```

```
We'll help get you setup with just a couple of questions. You can skip this
in the future by copying your config's
around:
```

```
    /root/.openshift/express.conf
    /root/.ssh/
```

```
To connect to broker.osop.cloud.lab.eng.bos.redhat.com enter your OpenShift
login (email or Red Hat login id): scollier
```

```
Password: *****
```

```
Created local config file: /root/.openshift/express.conf
```

```
The express.conf file contains user configuration, and can be transferred to
different computers.
```

```
No SSH keys were found. We will generate a pair of keys for you.
```

```
    Created: /root/.ssh/id_rsa.pub
```

```
Your public ssh key must be uploaded to the OpenShift server.  Would you
like us to upload it for you? (yes/no) yes
```

```
You can enter a name for your key, or leave it blank to use the default
name. Using the same name as an existing
key will overwrite the old key.
```

```
Since you do not have any keys associated with your OpenShift account,
your new key will be uploaded as the 'default' key
```

```
type: ssh-rsa
```

```
content:
```

```
AAAAB3NzaC1yc2EAAAADAQABAAQDofe3pyltnQsrow6YPvM0LcIFHpTLd3lxubesQHoDBb/yx
PBjTE0UgTHN6JR0+Yi7RyI5P+6PyrBUdsmT3Z
+c4YrKs5kFJkMcA0EmDOxxsKz5NWPnb/0wV8R9bsdW1kvevKRURD3MVNT+EAttQ07BcWF6VVFvDR
7IYYBqUBYFploGWNwf7X8XKrL16s1L9hYcgEjB
1qnh+IJtBo/cCgyzcA9EMACwgf+m2WVDEDl3p1NiU7+H1hxeKzcpB6mhvC+fNVitlcc62ok6rDfs
WdjrnJT97smQYUJeuXG/fp1ZSYpM0wWUA/sxCC
ZNq5DHASRg92HcWo4SZGi3QF0HUKUx
```

```
fingerprint: 73:34:7f:9f:c8:32:84:81:3e:6c:88:0b:95:46:8c:85
```

```
Uploading key 'default' from /root/.ssh/id_rsa.pub
```

```
We will now check to see if you have the necessary client tools installed.
```

```
Checking for git ... found
```

```
Checking for your namespace ... not found
```

```
Your namespace is unique to your account and is the suffix of the public
URLs we assign to your applications. You
may configure your namespace here or leave it blank and use 'rhc domain
create' to create a namespace later.  You
will not be able to create applications without first creating a namespace.
```



Please enter a namespace or leave this blank if you wish to skip this step:
refarch

Your domain name 'refarch' has been successfully created

Checking for applications ... none found

Run 'rhc app create' to create your first application.

Below is a list of the types of application you can create: * diy-0.1 -

rhc app create <app name> diy-0.1

- * jbosseap-6.0 - rhc app create <app name> jbosseap-6.0
- * jbossews-1.0 - rhc app create <app name> jbossews-1.0
- * jenkins-1.4 - rhc app create <app name> jenkins-1.4
- * perl-5.10 - rhc app create <app name> perl-5.10
- * php-5.3 - rhc app create <app name> php-5.3
- * python-2.6 - rhc app create <app name> python-2.6
- * ruby-1.8 - rhc app create <app name> ruby-1.8
- * ruby-1.9 - rhc app create <app name> ruby-1.9

Thank you for setting up your system. You can rerun this at any time by calling 'rhc setup'.

4. Confirm connectivity by reviewing domain status

\$ rhc domain show

Applications in refarch

=====

No applications. You can use 'rhc app create' to create new applications.

5.2 Infrastructure Operation

This section covers some checks that can be performed to ensure proper operation of the OpenShift Enterprise infrastructure.

5.2.1 Confirm Hosts

Broker:

1. Confirm the messaging infrastructure is operational. From both of the broker hosts, check to see that the ActiveMQ hosts are passing messages to and from the nodes.

mco ping

```
node1.osop.cloud.lab.eng.bos.redhat.com  time=52.81 ms
node2.osop.cloud.lab.eng.bos.redhat.com  time=92.75 ms
node3.osop.cloud.lab.eng.bos.redhat.com  time=94.30 ms
```

2. Confirm that the REST api is functioning properly without authentication. Check for a "200" return value.

curl -Ik https://localhost/broker/rest/api

```
HTTP/1.1 200 OK
Date: Thu, 20 Dec 2012 14:58:29 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: Phusion Passenger (mod_rails/mod_rack) 3.0.17
```



```
X-Runtime: 0.016841
X-UA-Compatible: IE=Edge,chrome=1
ETag: "029bd6c90e575808973f7277e06e0f44"
Cache-Control: max-age=0, private, must-revalidate
Status: 200
Content-Type: application/json; charset=utf-8
Connection: close
```

3. Confirm that the REST api is functioning properly with authentication. If **curl** returns json output, it is working.

```
# curl -k --user scollier:PASSWORDHERE https://broker/broker/rest/domains |
python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time
Current                                  Dload  Upload   Total   Spent    Left
Speed
103 1968 103 1968    0    0  6051      0 --:--:-- --:--:-- --:--:--
7903
{
  "data": [
    {
      "id": "refarchtwo",
      "links": {
        "ADD_APPLICATION": {
          "href":
"https://broker/broker/rest/domains/refarchtwo/applications",
          "method": "POST",
          "optional_params": [
            ... < output truncated > ...
```

4. Run the **oo-accept-broker** script to ensure overall operation.

```
# oo-accept-broker
FAIL: Datastore Password is still the default
1 ERRORS
```

oo-accept-broker is very helpful for pinpointing problems with the infrastructure. For a more verbose output, run with the verbose option.

```
# oo-accept-broker -v
INFO: SERVICES: DATA: mongo, Auth: mongo, Name bind
INFO: AUTH_MODULE: rubygem-openshift-origin-auth-mongo
INFO: NAME_MODULE: rubygem-openshift-origin-dns-bind
INFO: Broker package is: openshift-origin-broker
INFO: checking packages
INFO: checking package ruby
INFO: checking package rubygems
INFO: checking package rubygem-rails
INFO: checking package rubygem-passenger
INFO: checking package rubygem-openshift-origin-common
INFO: checking package rubygem-openshift-origin-controller
INFO: checking package openshift-origin-broker
INFO: checking ruby requirements
INFO: checking ruby requirements for openshift-origin-controller
```



```
INFO: checking ruby requirements for config/application
INFO: checking firewall settings
INFO: checking services
INFO: checking datastore
INFO: datastore plugin: OpenShift::MongoDataStore
INFO: checking mongo datastore configuration
INFO: Datastore Host: bsn1.osop.cloud.lab.eng.bos.redhat.com
INFO: Datastore Port: 27017
INFO: Datastore User: openshift
FAIL: Datastore Password is still the default
INFO: Datastore DB Name: openshift_broker_dev
INFO: Datastore: mongo db service is remote
INFO: checking mongo db login access
INFO: mongo db login successful:
bsn1.osop.cloud.lab.eng.bos.redhat.com:27017/openshift_broker_dev --username
openshift
INFO: checking cloud user authentication
INFO: auth plugin = OpenShift::RemoteUserAuthService
INFO: auth plugin: OpenShift::RemoteUserAuthService
INFO: checking remote-user auth configuration
INFO: Auth trusted header: REMOTE_USER
INFO: Auth passthrough is enabled for OpenShift services
INFO: Got HTTP 200 response from https://localhost/broker/rest/api
INFO: Got HTTP 200 response from
https://localhost/broker/rest/application_templates
INFO: Got HTTP 200 response from https://localhost/broker/rest/cartridges
INFO: Got HTTP 401 response from https://localhost/broker/rest/user
INFO: Got HTTP 401 response from https://localhost/broker/rest/domains
INFO: checking dynamic dns plugin
INFO: dynamic dns plugin = OpenShift::BindPlugin
INFO: checking bind dns plugin configuration
INFO: DNS Server: 10.16.138.14
INFO: DNS Port: 53
INFO: DNS Key Name: osop.cloud.lab.eng.bos.redhat.com
INFO: DNS Key Value: *****
INFO: DNS Zone: osop.cloud.lab.eng.bos.redhat.com
INFO: DNS Domain Suffix: osop.cloud.lab.eng.bos.redhat.com
INFO: adding txt record named testrecord.osop.cloud.lab.eng.bos.redhat.com
to server 10.16.138.14: this_is_a_test
INFO: txt record successfully added
INFO: deleteing txt record named
testrecord.osop.cloud.lab.eng.bos.redhat.com to server 10.16.138.14:
INFO: txt record successfully deleted
INFO: checking messaging configuration
INFO: messaging plugin = OpenShift::MCollectiveApplicationContainerProxy
1 ERRORS
```

5. **oo-admin-chk** is another tool for verifying the infrastructure

```
# oo-admin-chk
Check failed.
Gear 2ab2b3a685774d74b521de405690cf81 exists on node
[node3.osop.cloud.lab.eng.bos.redhat.com, uid:500] but does not exist in
mongo database
Gear 1e2d45b3ebbd4f20823b667d0c950906 exists on node
```



```
[node1.osop.cloud.lab.eng.bos.redhat.com, uid:500] but does not exist in
mongo database
Gear 4728efa5a00d46748715a33cc85122b2 exists on node
[node1.osop.cloud.lab.eng.bos.redhat.com, uid:501] but does not exist in
mongo database
```

Node:

1. Run the tool **oo-accept-node** on the node hosts to ensure proper functionality.

```
# oo-accept-node
PASS
```

Run with the verbose option to see more output and get a sense of what it is checking.

```
# oo-accept-node -v
INFO: loading node configuration file /etc/openshift/node.conf
INFO: loading resource limit file /etc/openshift/resource_limits.conf
INFO: checking node public hostname resolution
INFO: checking selinux status
INFO: checking selinux origin policy
INFO: checking selinux booleans
INFO: checking package list
INFO: checking services
INFO: checking kernel semaphores >= 512
INFO: checking cgroups configuration
INFO: checking presence of /cgroup
INFO: checking presence of /cgroup/all
INFO: checking presence of /cgroup/all/openshift
INFO: checking filesystem quotas
INFO: checking quota db file selinux label
INFO: checking 2 user accounts
INFO: checking application dirs
INFO: checking system httpd configs
PASS
```

BSN:

1. Confirm that the ActiveMQ hosts are returning topics

```
# curl --user admin:badpassword --silent
http://localhost:8161/admin/xml/topics.jsp | grep -A 4 topic
<topics>

<topic
name="ActiveMQ.Advisory.Consumer.Topic.mcollective.mcollective.command">

  <stats size="0"
    consumerCount="4"
    enqueueCount="4"
  --
</topic>

... < output truncated > ...
```



LVS:

1. On the LVS hosts, ensure the routing table is set up properly.

```
# ipvsadm
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  broker.osop.cloud.lab.eng.bo rr persistent 300
  -> broker1.osop.cloud.lab.eng.b Route    1      0          0
  -> broker2.osop.cloud.lab.eng.b Route    1      0          0
```

5.2.2 Log Files

When troubleshooting the OpenShift Enterprise infrastructure, here are the locations of log files that may be helpful.

Broker:

```
/var/log/httpd/*
/var/log/openshift/user_action.log
/var/www/openshift/broker/log/production.log
/var/www/openshift/broker/httpd/logs/*
/var/www/openshift/console/log/production.log
/var/www/openshift/console/httpd/logs/*
/var/log/messages
```

Nodes:

```
/var/log/mcollective.log
/var/log/messages
```

LVS:

```
/var/log/messages
```

BSN:

```
/var/log/activemq/*.log
/var/log/mongodb/mongodb.log
/var/log/messages
```

5.3 Applications

This section describes how to create applications using both the rhc tools, JBoss Developer Studio, and the OpenShift console.

5.3.1 Deploy PHP Application via Command Line

Create a basic PHP application:



1. Using the client tools, create a basic PHP application

```
$ rhc app create phpapp -t php-5.3
```

```
Creating application 'phpapp'
```

```
=====
```

```
Cartridge: php-5.3
```

```
Namespace: refarch
```

```
Scaling: no
```

```
Gear Size: default
```

Your application's domain name is being propagated worldwide (this might take a minute)...

The authenticity of host 'phpapp-refarch.osop.cloud.lab.eng.bos.redhat.com (10.16.138.29)' can't be established.

RSA key fingerprint is 99:df:c4:c4:aa:6e:a0:44:ac:e1:ce:5c:82:11:4c:49.

Are you sure you want to continue connecting (yes/no)? **yes**

Initialized empty Git repository in /home/scollier0/phpapp/.git/
done

```
phpapp @ http://phpapp-refarch.osop.cloud.lab.eng.bos.redhat.com/
```

```
=====
```

```
Application Info
```

```
Git URL =
```

```
ssh://abbf002e267c4a399d8a49dc82acfc3@phpapp-  
refarch.osop.cloud.lab.eng.bos.redhat.com/~/.git/phpapp.git/
```

```
Gear Size = small
```

```
SSH URL = ssh://abbf002e267c4a399d8a49dc82acfc3@phpapp-  
refarch.osop.cloud.lab.eng.bos.redhat.com
```

```
Created = 10:53 AM
```

```
UUID = abbf002e267c4a399d8a49dc82acfc3
```

```
Cartridges
```

```
=====
```

```
php-5.3
```

RESULT:

Application phpapp was created.

2. Display the application

```
$ rhc app show --stat phpapp
```

RESULT:

Geargroup php-5.3 is started



3. Browse to the default page of the application using the URL that was provided during application creation.

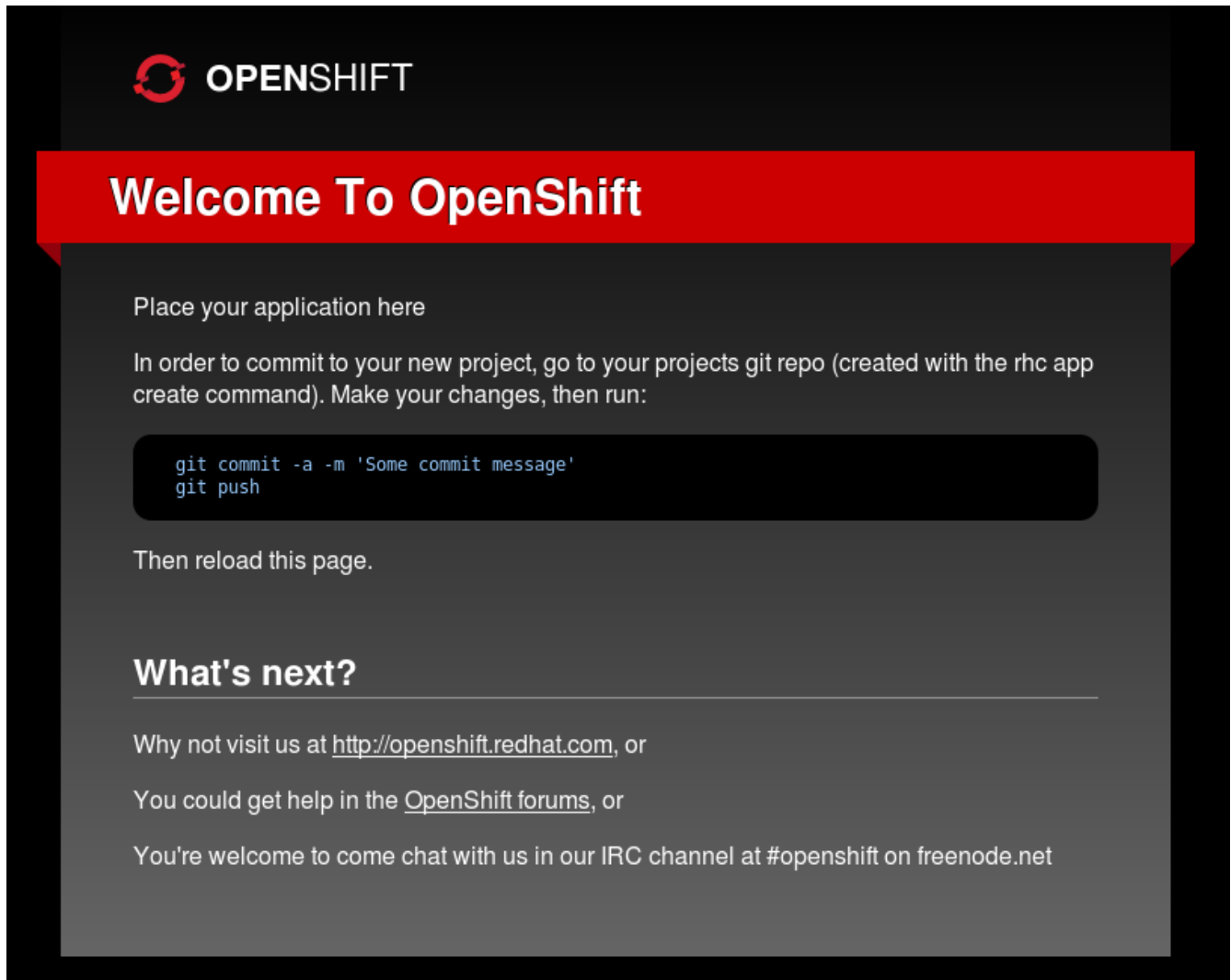


Figure 5.3.1.1: PHP App before

4. Make a change to the phpapp and push the changes with git. Change to the directory that the application was cloned to

```
$ pwd
/home/scollier0/git_projects/phpapp
```

5. Edit the `/home/scollier0/git_projects/phpapp/php/index.php` file to make a change

```
$ sed -i 's/Welcome to OpenShift/HERES MY PHP APP/' php/index.php
```

6. Check the status of git

```
$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
```



```
directory)
#
#    modified:   php/index.php
#
no changes added to commit (use "git add" and/or "git commit -a")
```

7. Commit the change and push it up to the OpenShift Enterprise server

```
$ git commit -a -m 'first phpapp commit'
[master 54eb94f] first phpapp commit
Committer: scollier0 <scollier0@client.osop.cloud.lab.eng.bos.redhat.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
```

```
    git config --global user.name "Your Name"
    git config --global user.email you@example.com
```

If the identity used for this commit is wrong, you can fix it with:

```
    git commit --amend --author='Your Name <you@example.com>'
```

```
1 files changed, 2 insertions(+), 2 deletions(-)
```

8. Check the status of git again and it should be clean

```
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
nothing to commit (working directory clean)
```

9. Push the changes with git

```
$ git push
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 413 bytes, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: restart_on_add=false
remote: Waiting for stop to finish
remote: Done
remote: restart_on_add=false
remote: ~/git/phpapp.git ~/git/phpapp.git
remote: ~/git/phpapp.git
remote: Running .openshift/action_hooks/pre_build
remote: Running .openshift/action_hooks/build
remote: Running .openshift/action_hooks/deploy
remote: hot_deploy_added=false
remote: Done
remote: Running .openshift/action_hooks/post_deploy
To ssh://abbf002e267c4a399d8a49dc82acfc3b3@phpapp-
refarch.osop.cloud.lab.eng.bos.redhat.com/~/git/phpapp.git/
3edf63b..54eb94f master -> master
```



10. Refresh the phpapp in the browser and notice the change

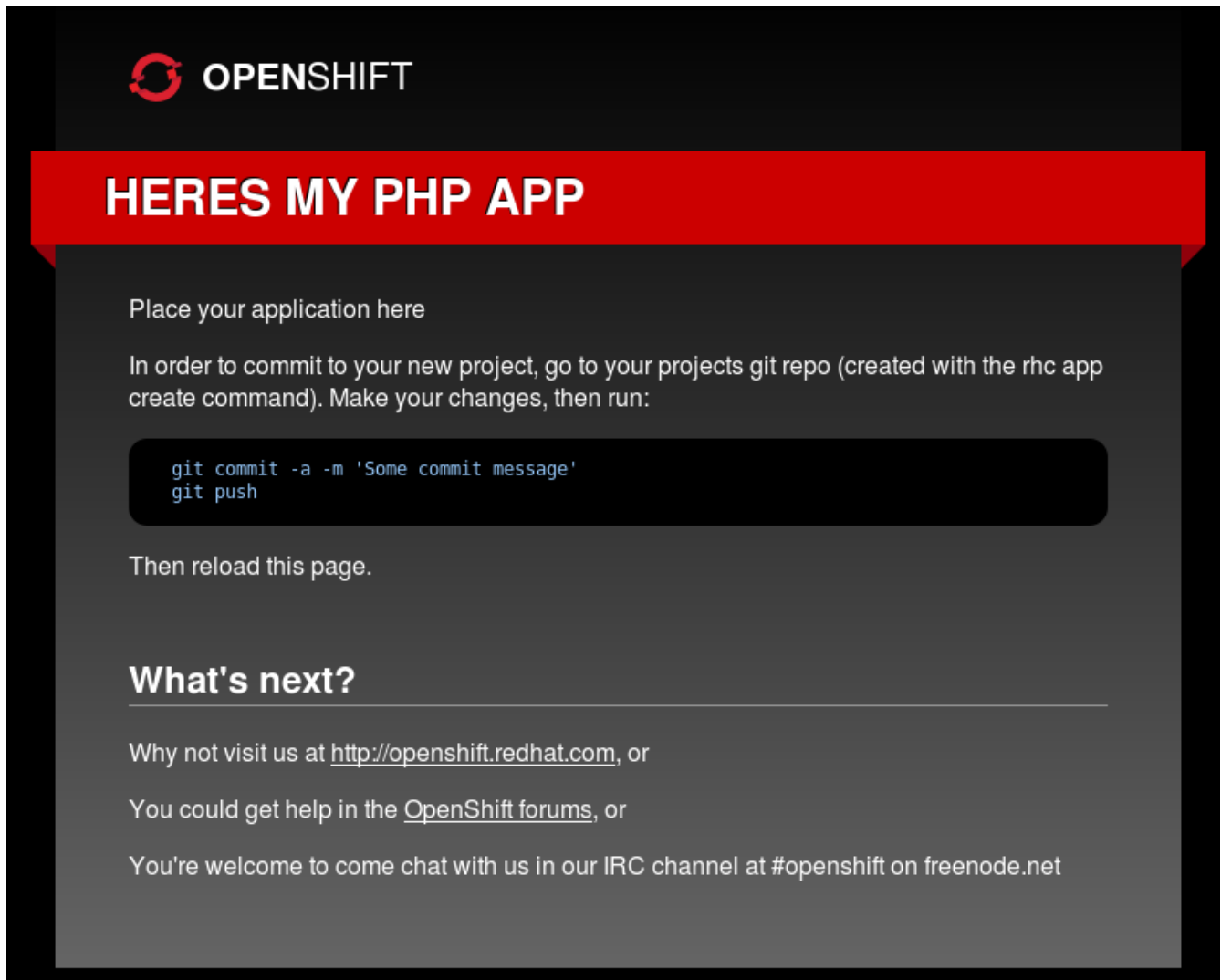


Figure 5.3.1.2: PHP App After



5.3.2 Deploy PHP Application via OpenShift Console

The OpenShift Enterprise console was deployed in **Deploy OpenShift Enterprise Console**. This section covers how to use the OpenShift enterprise console to create a simple php application. As of this writing OpenShift Enterprise is a tech preview.

1. Browse to the broker virtual IP address and open the OpenShift Enterprise console using the following URL

<https://broker.osop.cloud.lab.eng.bos.redhat.com/console>

- (a) Provide credentials of a LDAP user to access the console

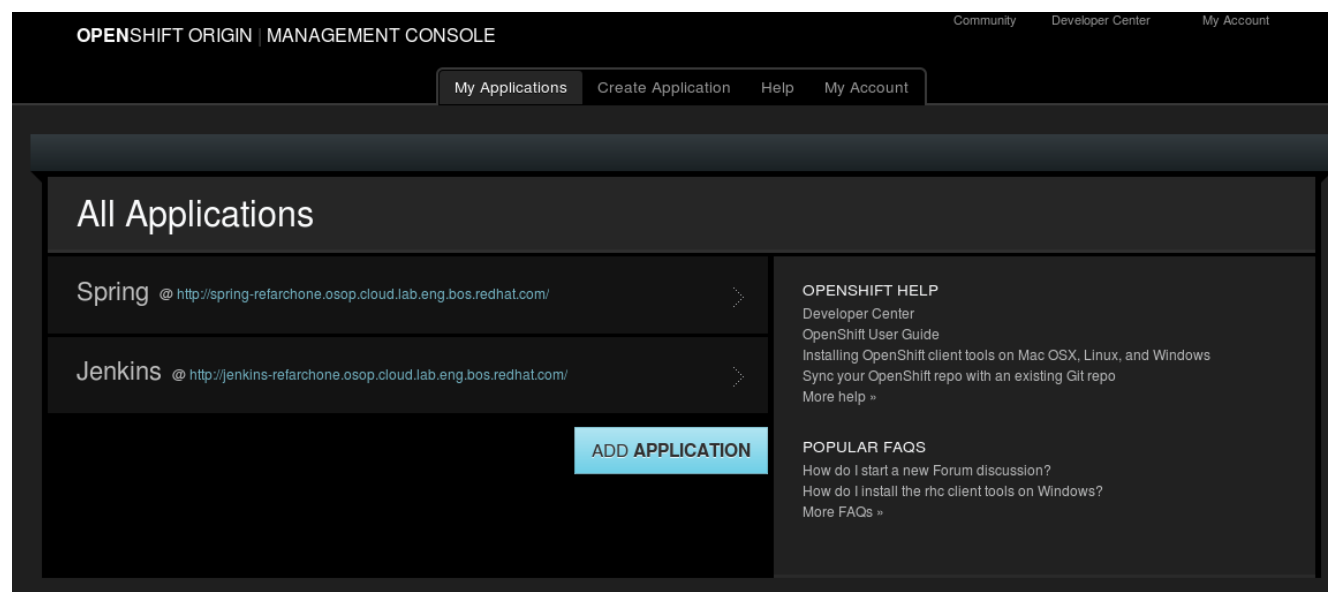


Figure 5.3.2.1: OpenShift Console

- (b) Click on **Create Application**

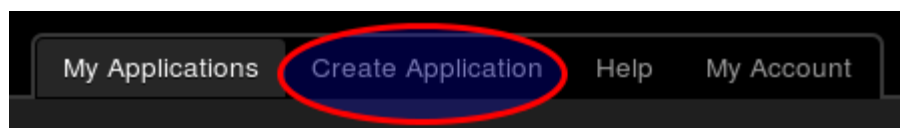


Figure 5.3.2.2: Create Application



(c) Select **PHP 5.3**

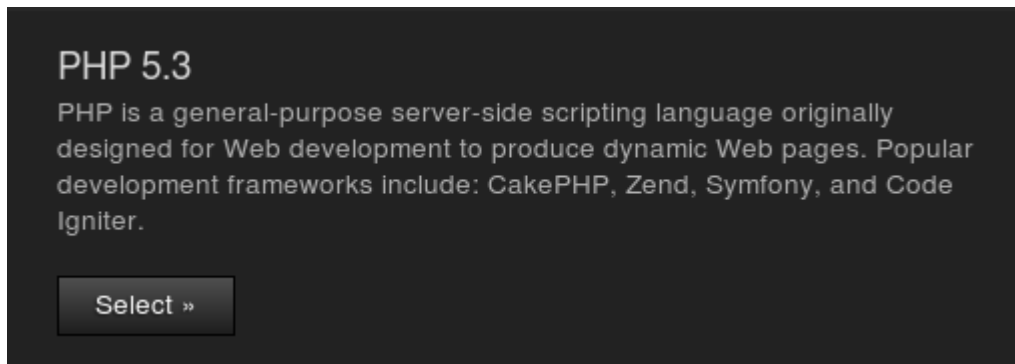


Figure 5.3.2.3: Create PHP Application

(d) Provide a **PUBLIC URL** and click **Create Application**

Figure 5.3.2.4: Provide Application Name



(e) Note the link provided to the application

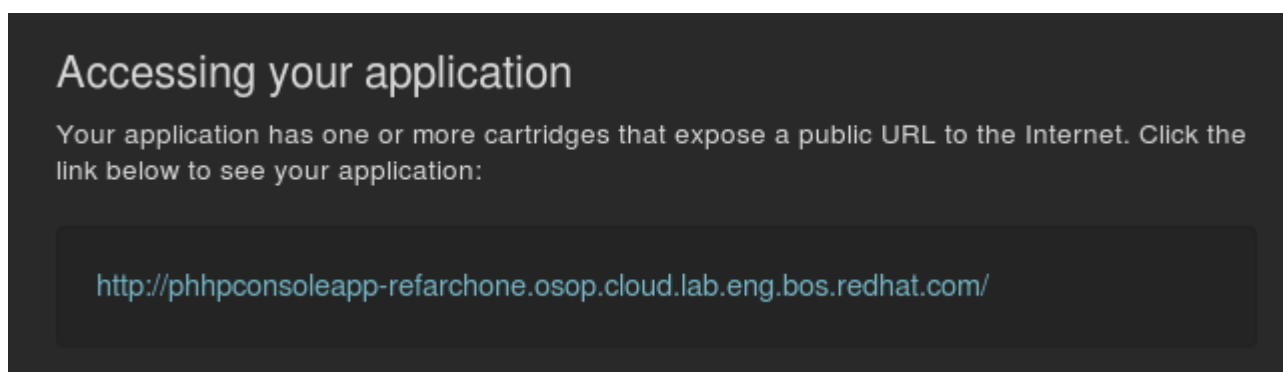


Figure 5.3.2.5: Application URL

(f) Open the link and go to the application

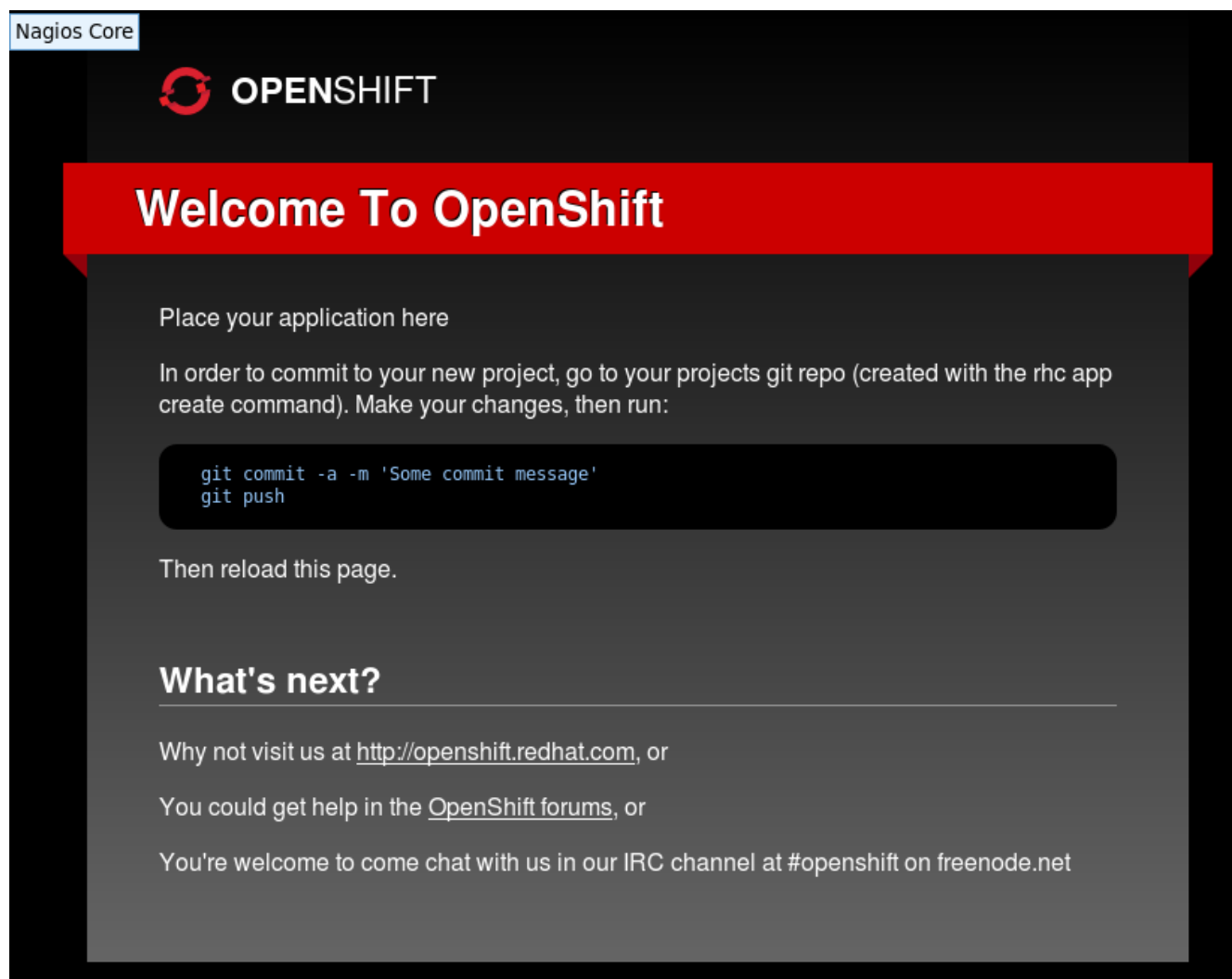


Figure 5.3.2.6: Console Application



(g) Managing the code is the same as in **Deploy PHP Application via Command Line**

5.3.3 Deploy JBoss Application with JBoss Developer Studio 5

This section demonstrates how to use JBoss Developer Studio (JBDS) 5.0.1 with OpenShift Enterprise. The following steps need to take place in order for this to work.

- Install JBDS on the client system
- Configure JBDS to point to OpenShift Enterprise
- Deploy Application

Install JBDS on the client system

1. Download the version 5 of JBDS from access.redhat.com.

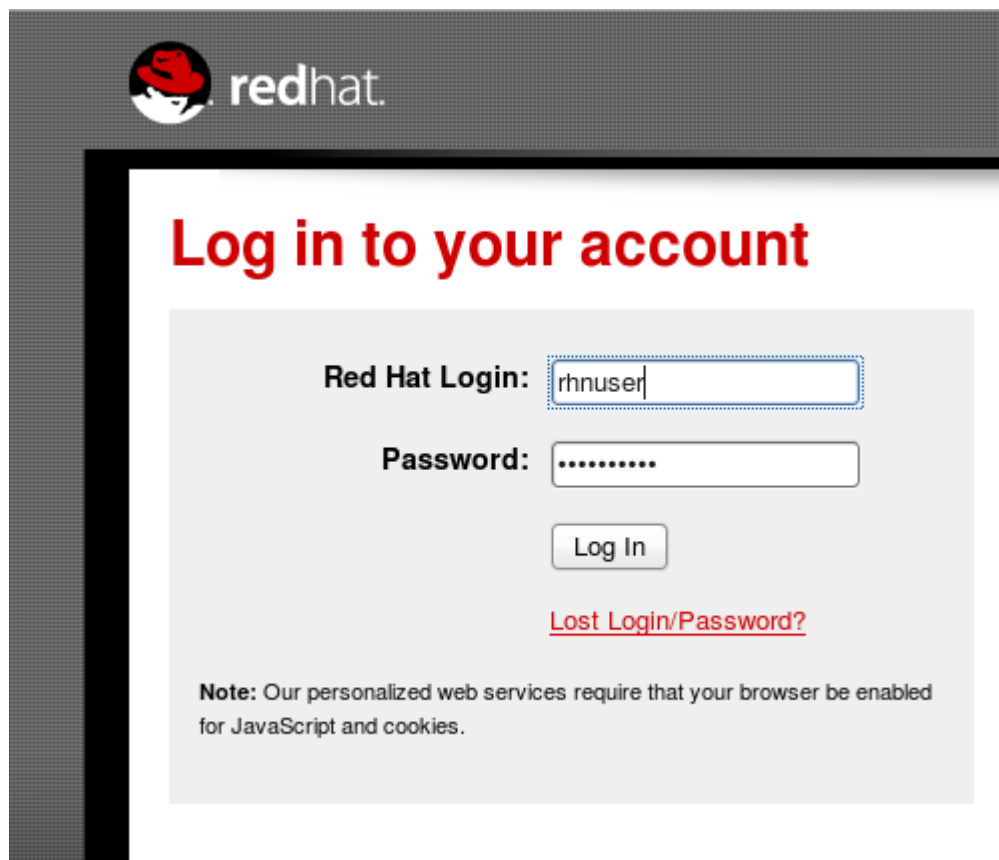


Figure 5.3.3.1: RHN Access

2. Click **Downloads** in the upper navigation menu and select **Download Software** for



JBoss Enterprise Middleware

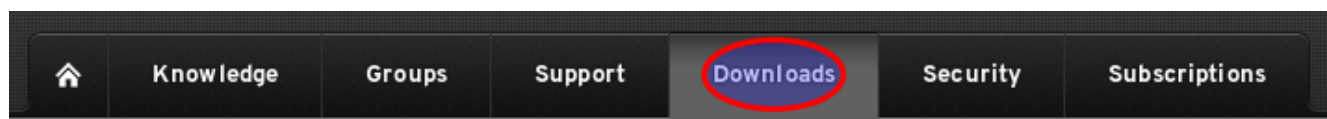


Figure 5.3.3.2:

3. For the **Product** dropdown menu, select **JBoss Developer Studio**



Figure 5.3.3.3: Download JBDS

4. Download **JBoss Developer Studio Universal Binary with EAP** Version 5

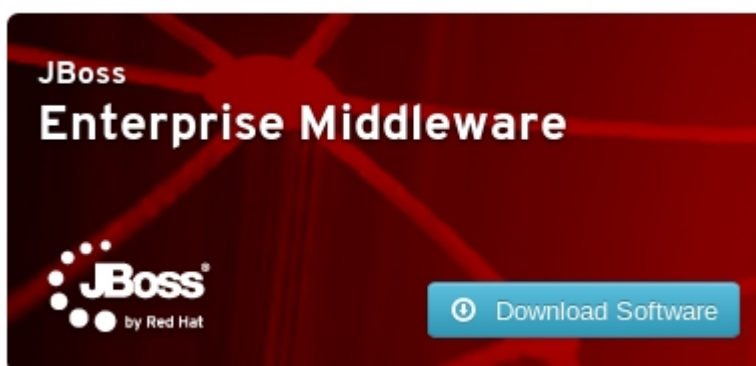


Figure 5.3.3.4: Download JBDS

5. Save the file to your home directory

6. Install JBDS

```
$ java -jar jbdevstudio-product-eap-universal-5.0.0.v20120615-1714-H213-GA.jar
```

Take all the defaults. For more details on installing JBDS, see the installation guide¹⁰. If the install is being done remotely, use a “ssh -Y” for X connection.

7. Create a `~/etc/eclipse.ini` file to add the parameter in bold. This file is honored by JBDS.

```
-vmargs -Dlibra_server=broker.osop.cloud.lab.eng.bos.redhat.com
```

8. Launch JBDS and take the defaults, including the workspace

```
$ /home/scollier0/jbdevstudio/jbdevstudio
```




9. Update JBDS

Click **Help** -> **Check for Update** -> Complete the wizard and install all updates.

10. Choose **OpenShift Application** in the **JBoss Central** tab

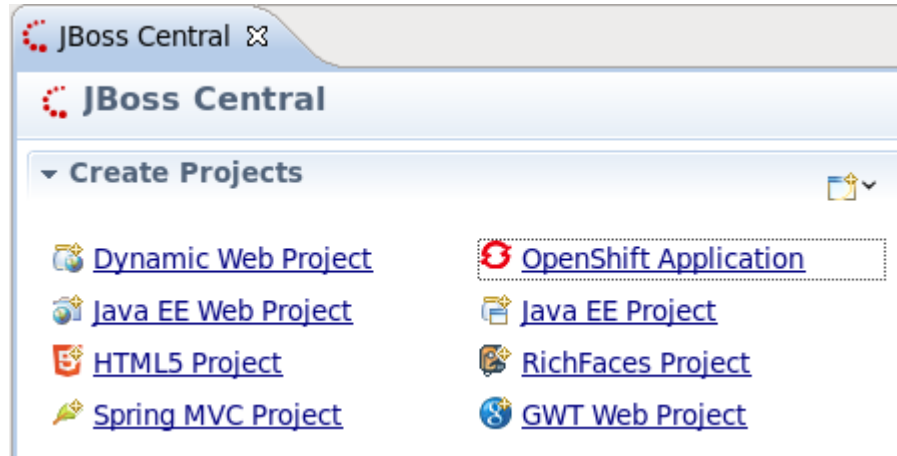


Figure 5.3.3.5: Create Application


11. Provide login credentials and click **Next**



New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.com)

Sign in to OpenShift

Please provide your OpenShift credentials.



OPENSIFT

If you do not have an account on OpenShift, please sign up [here](#).

Username

Password

☐ Save password (could trigger secure storage login)




Figure 5.3.3.6: JBDS Credentials



12. Set up OpenShift application. Provide a name and type for the new application. Click **Next**.

Figure 5.3.3.7: JBDS Application name

New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.com)

Setup OpenShift Application

Select an existing or create a new OpenShift Application.

☐ Use existing application:

New application

Name:

Type:

Gear profile: ☐ Enable scaling

Embeddable Cartridges

- ☐ cron-1.4
- ☐ haproxy-1.4
- ☐ jenkins-client-1.4
- ☐ mysql-5.1
- ☐ postgresql-8.4



13. Click **Next** to create a new project.

New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.com)

Setup Project for OpenShift Application "refarc"

Configure your project and server adapter settings, then click 'next' or 'finish'.

☒ Create a new project

Use existing project:

Server Adapter

☒ Create and setup a server for easy publishing

Figure 5.3.3.8: JBDS Existing Project



14. Change the location of your git repository and click **Finish**. If this operation times out, click finish one more time and accept the ssh key.

Figure 5.3.3.9: JBDS Git location

SSH2 Preferences.' The OpenShift logo is in the top right corner."/>



15. Click the **OpenShift Explorer** tab and then right click on the **refarchjboss jbosseap-6.0** application and choose **Web Browser**.

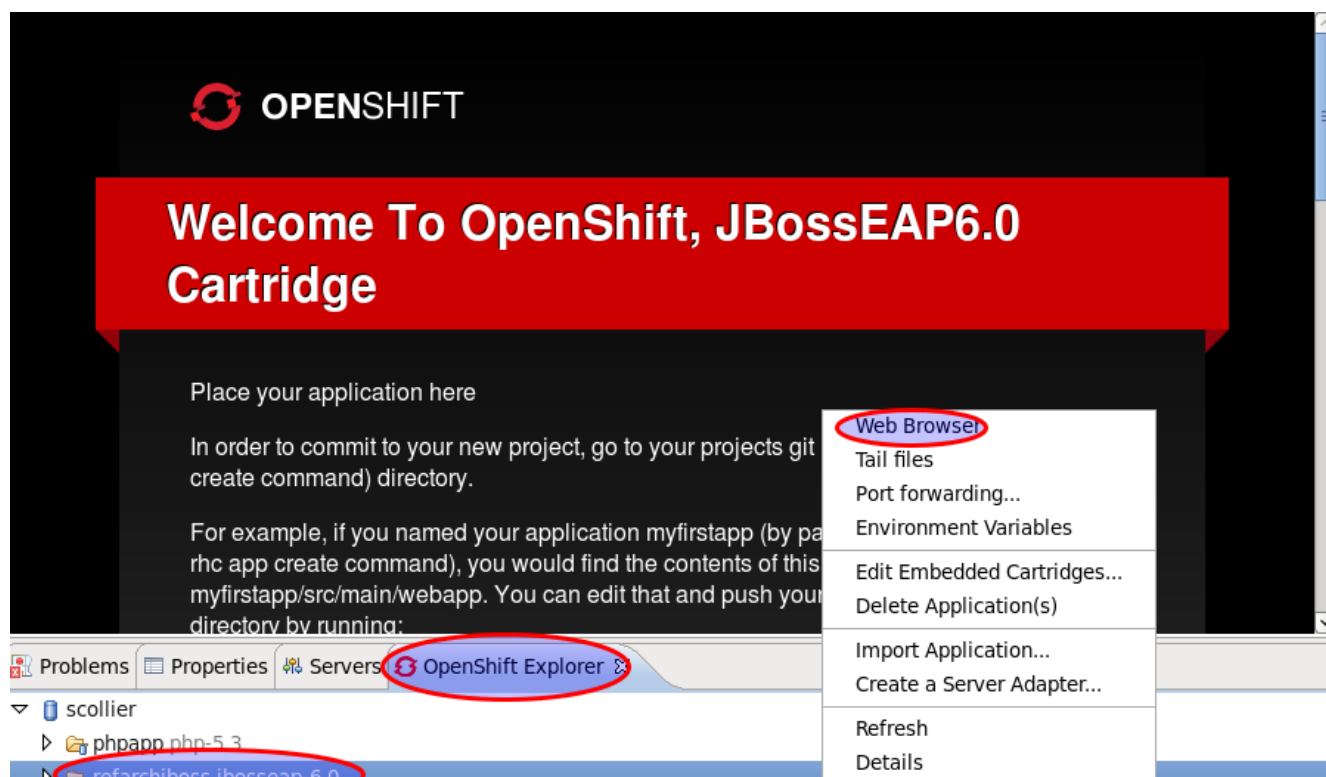
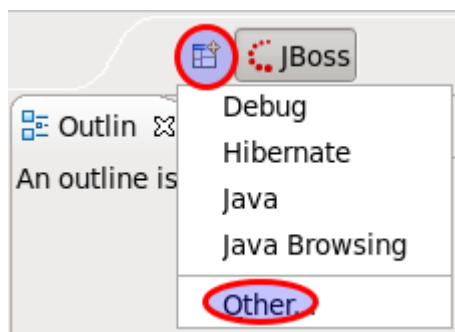


Figure 5.3.3.10: JBDS Application

Deploy Kitchen Sink application

1. Click on the **Open Perspective** button in the upper right of the screen and choose **Other**



**Figure 5.3.3.11: JBDS
Perspective**



2. Select **Remote System Explorer** and click **OK**

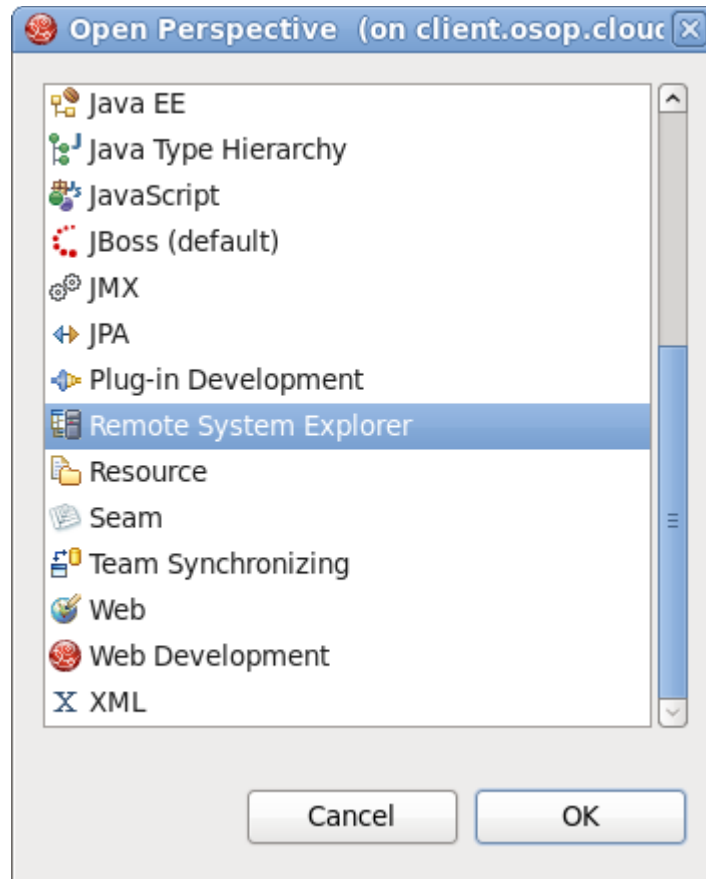


Figure 5.3.3.12: JBDS Remote System Explorer



3. In the new window, right click on **Local Shells** and select **Launch Shell**

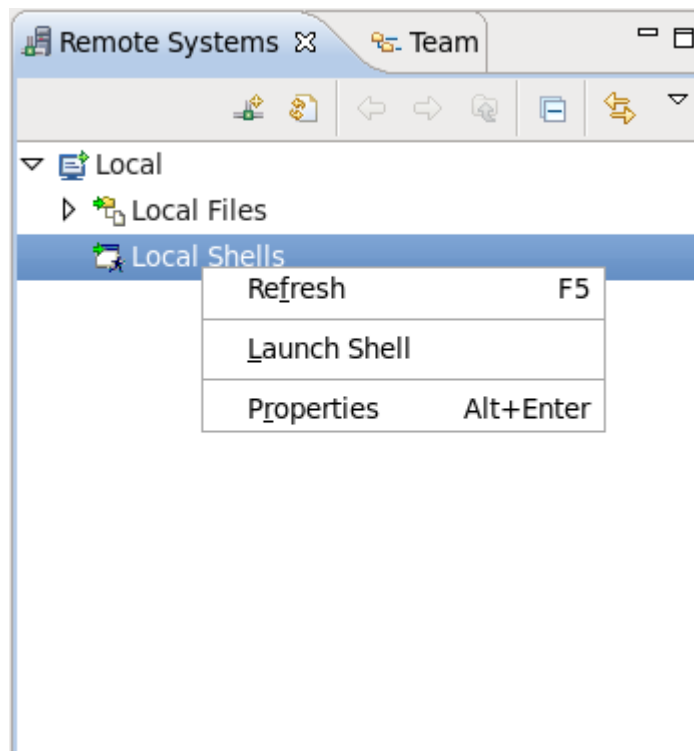


Figure 5.3.3.13: JBDS Launch Shell

4. Now there is a shell opened up in the lower navigation window.
5. Change to the refarchjboss git repository

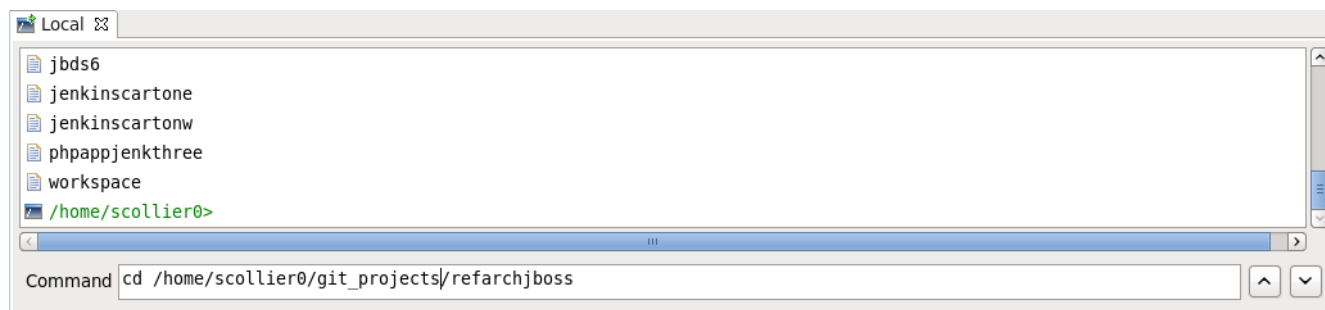


Figure 5.3.3.14: JBDS Shell



6. Add the master branch from upstream as a remote (showing command typed into the shell)

```
git remote add upstream -m master git://github.com/openshift/kitchensink-example.git
```



Figure 5.3.3.15: JBDS Shell

7. If there is a merge conflict, add the faces-config.xml file and push, then continue with the pull from upstream master. The following commands can be typed into the shell.

```
git add src/main/webapp/WEB-INF/faces-config.xml
git commit -a -m 'adding file to avoid merge conflicts'
git push
```

8. Pull the master branch

```
git pull -s recursive -X theirs upstream master
```

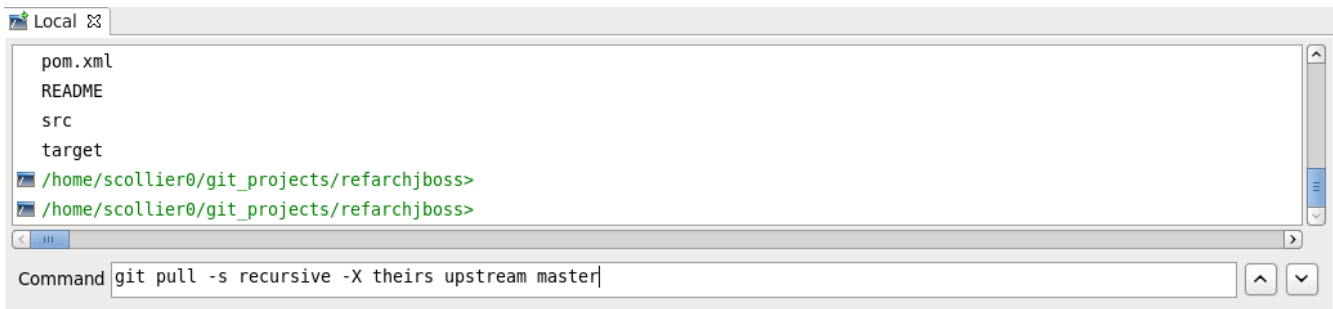


Figure 5.3.3.16: JBDS Shell

9. Push the changes. Another way to push the changes is to right click on the server and click **Publish**.

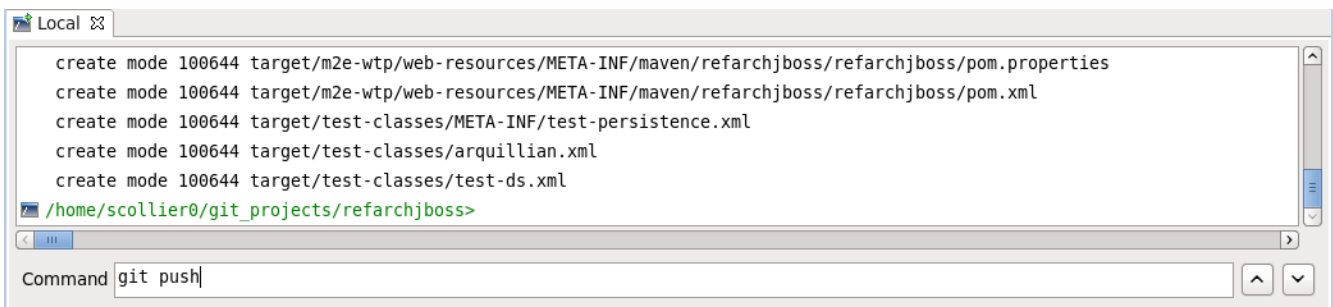


Figure 5.3.3.17: JBDS Shell



10. After the changes have been pushed, refresh the browser to see the kitchensink application

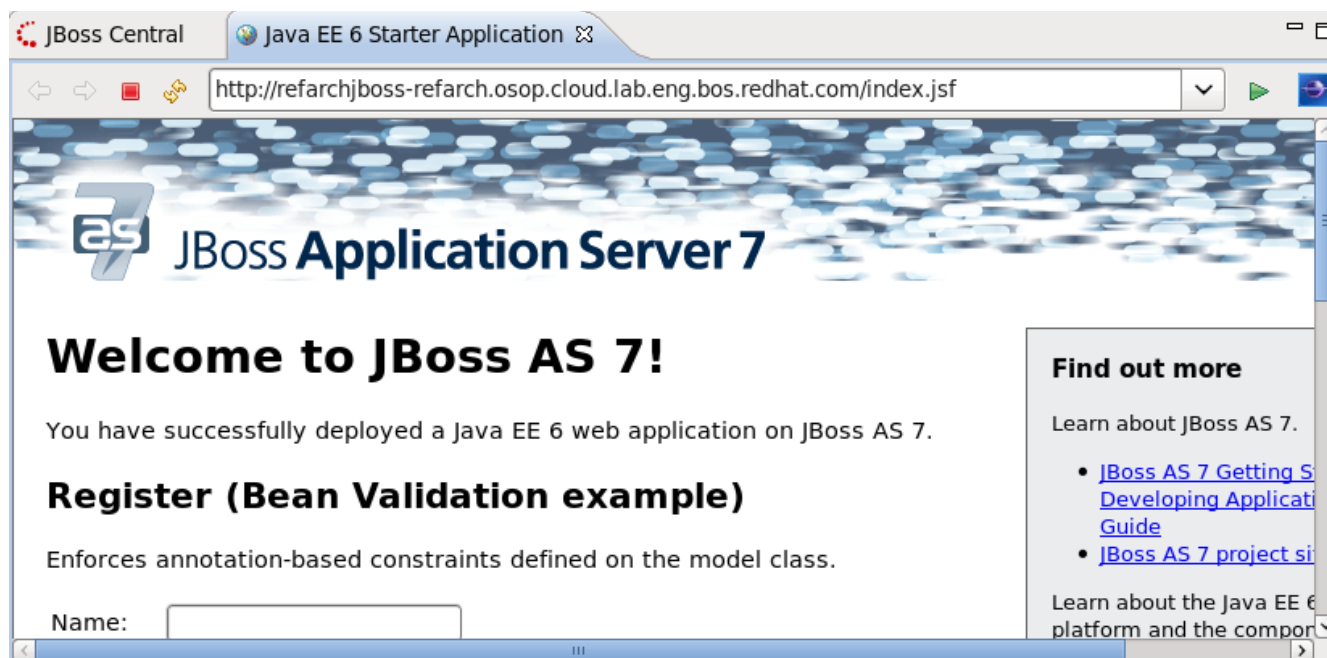


Figure 5.3.3.18: JBDS Kitchen Sink Application

5.3.4 Deploy JBoss Application with JBoss Developer Studio 6

This section demonstrates how to use JBoss Developer Studio (JBDS) 6 with OpenShift Enterprise. The following steps need to take place in order for this to work.

- Install JBDS on the client system
- Configure JBDS to point to OpenShift Enterprise
- Deploy Application

Install JBDS on the client system

1. Download the version 6 of JBDS from <http://access.redhat.com>.



The screenshot shows the Red Hat login page. At the top left is the Red Hat logo and the word "redhat.". Below this is a large red heading "Log in to your account". Underneath is a light gray box containing the login form. The form has two labels: "Red Hat Login:" and "Password:". The "Red Hat Login:" field contains the text "rhuser". The "Password:" field contains a series of dots. Below these fields is a "Log In" button. Under the button is a red link that says "Lost Login/Password?". At the bottom of the gray box is a "Note" in small text: "Note: Our personalized web services require that your browser be enabled for JavaScript and cookies."

Figure 5.3.4.1: RHN Access

2. Click **Downloads** in the upper navigation menu and select **Download Software** for **JBOSS ENTERPRISE MIDDLEWARE**

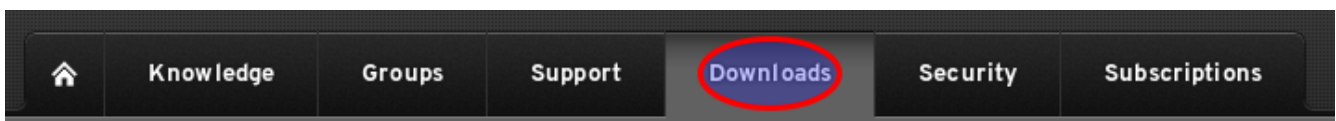


Figure 5.3.4.2:

3. For the **Product** dropdown menu, select **JBoss Developer Studio**



Product:

Figure 5.3.4.3: Download JBDS

4. Download JBoss Developer Studio Universal Binary with EAP Version 6

Software Downloads

Product:

Version:

[Releases \(3\)](#) [Patches \(0\)](#) [Security Advisories \(0\)](#)

Download File	Release Date	
JBoss Developer Studio 6.0.0 Source Code	12/07/2012 03:16 PM EDT	Download
JBoss Developer Studio 6.0.0 Universal Binary with EAP	12/07/2012 03:16 PM EDT	Download
JBoss Developer Studio 6.0.0 Stand Alone Universal Binary	12/07/2012 03:16 PM EDT	Download

Figure 5.3.4.4: JBDS v6

5. Save the file to your home directory

6. Install JBDS

```
$ # java -jar jbdevstudio-product-eap-universal-6.0.0.GA-v20121206-1855-B186.jar
```

Take all the defaults. For more details on installing JBDS, see the installation guide¹¹. If the install is being done remotely, use a “ssh -Y” for X connection.

7. Create a `~/etc/eclipse.ini` file to add the parameter in bold. This file is honored by JBDS.

```
-vmargs -Dlibra_server=broker.osop.cloud.lab.eng.bos.redhat.com
```

8. Launch JBDS and take the defaults, including the workspace

```
$ /home/scollier0/jbdevstudio/jbdevstudio
```

9. Update JBDS

Click **Help** -> **Check for Update** -> Complete the wizard and install all updates.

10. Choose **OpenShift Application** in the **JBoss Central** tab

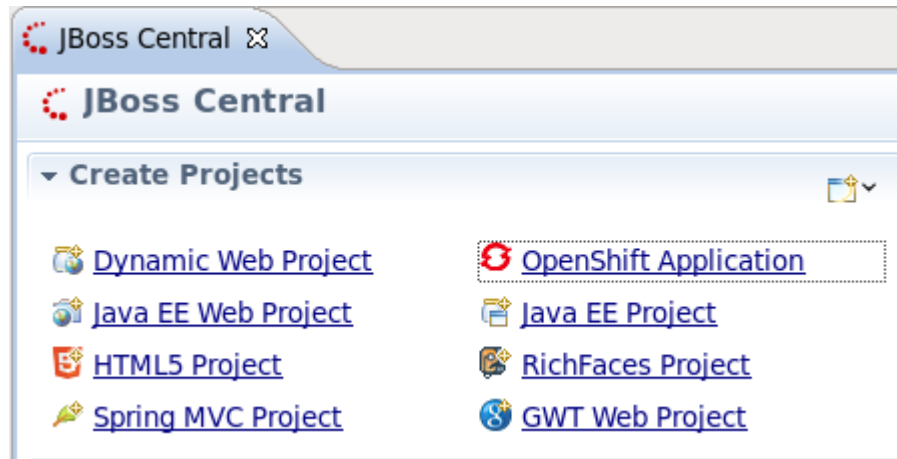



Figure 5.3.4.5: Create Application

11. Remove the check in the **Use Default Server** box and provide a **Server**, **Username** and **Password** and click **Next**.



New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.)

Sign in to OpenShift
Please provide your OpenShift credentials.


OPENSIFT

If you do not have an account on OpenShift, please sign up [here](#).

Connection: <New Connection> ▼

☐ Use default server

Server: https://broker.osop.cloud.lab.eng.bos.redhat.com ▼

Username: scollier

Password: ●●●●●●

☐ Save password (could trigger secure storage login)

ⓘ < Back Next > Cancel Finish

Figure 5.3.4.6:

12. Set up OpenShift application. Provide a name and type for the new application. Click **Next**.



Figure 5.3.4.7: JBDS Application name

New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.com)

Setup OpenShift Application

Select an existing or create a new OpenShift Application.

☐ Use existing application:

New application

Name:

Type:

Gear profile: ☐ Enable scaling

Embeddable Cartridges

- ☐ cron-1.4
- ☐ haproxy-1.4
- ☐ jenkins-client-1.4
- ☐ mysql-5.1
- ☐ postgresql-8.4


13. Click Next to create a new project.



New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.com)

Setup Project for OpenShift Application "refarc"

Configure your project and server adapter settings, then click 'next' or 'finish'.



OPENSIFT

☒ Create a new project

Use existing project:

Server Adapter

☒ Create and setup a server for easy publishing

Figure 5.3.4.8: JBDS Existing Project



14. Change the location of your git repository and click **Finish**. If this operation times out, click finish one more time and accept the ssh key.

New OpenShift Application (on client.osop.cloud.lab.eng.bos.redhat.)

Import an existing OpenShift application

Configure the cloning settings by specifying the clone destination if you create a new project, and the git remote name if you're using an existing project.

Cloning settings

☐ Use default location

Location:

☒ Use default remote name

Remote name:

Make sure that you have SSH keys added to your OpenShift account scollier via [SSH Keys wizard](#) and that the private keys are listed in [SSH2 Preferences](#)

Figure 5.3.4.9: JBDS Git Repositories



15. Click the **OpenShift Explorer** tab and then right click on the **refarchjboss jbosseap-6.0** application and choose **Web Browser**.

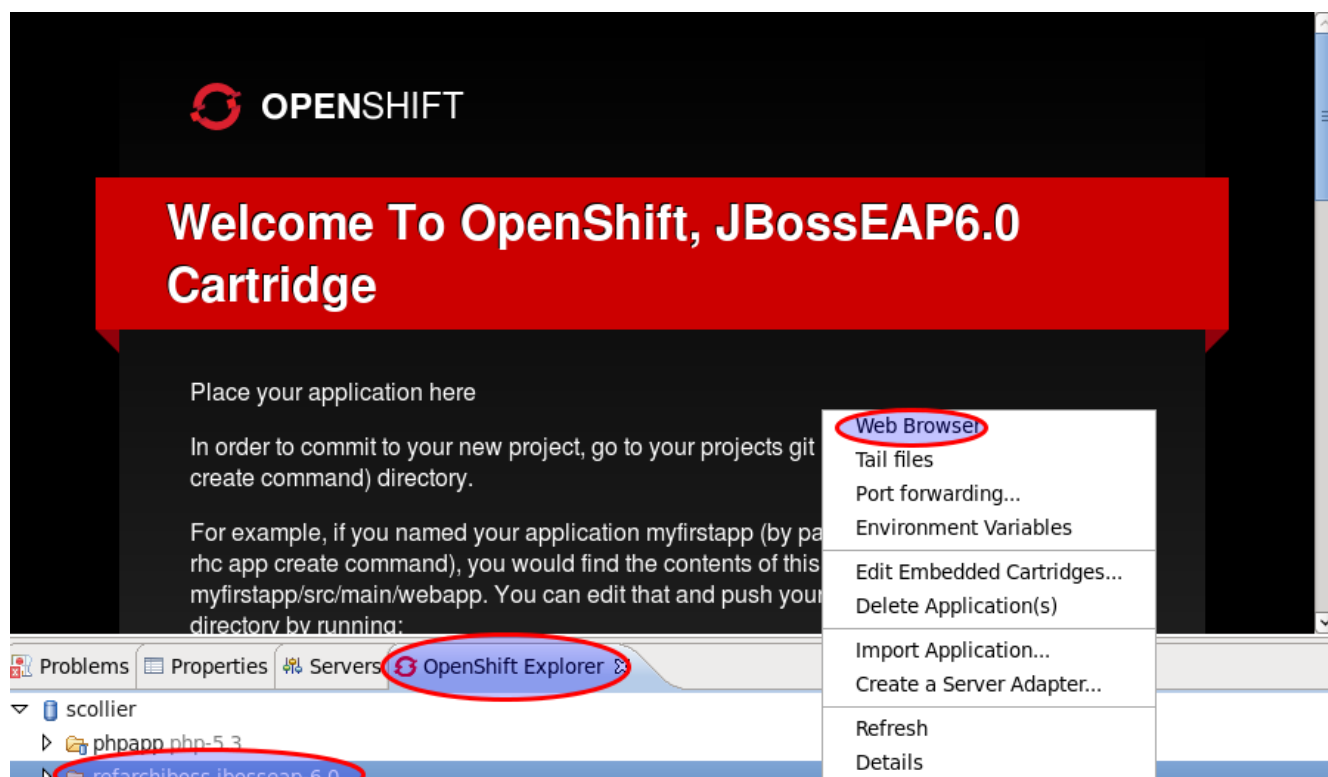


Figure 5.3.4.10: JBDS Application

Deploy Kitchen Sink application:

In the previous section the application was deployed using the local shell within the JBoss Developer Studio console. This section shows how to deploy the application via the command line.

1. Change to the refarchjboss git repository

```
$ cd /home/scollier0/git_projects/refarchjboss1/
```

2. Add the master branch from upstream as a remote (showing command typed into the shell)

```
$ git remote add upstream -m master git://github.com/openshift/kitchensink-example.git
```

3. If there is a merge conflict, add the faces-config.xml file and push, then continue with the pull from upstream master. The following commands can be typed into the shell.

```
$ git add src/main/webapp/WEB-INF/faces-config.xml
```

```
$ git commit -a -m 'adding file to avoid merge conflicts'
```



```
$ git push
```

4. If needed, configure git email and name properties

```
$ git config --global user.email "scollier@redhat.com"
```

```
$ git config --global user.name "Scott Collier"
```

5. Pull the master branch

```
$ git pull -s recursive -X theirs upstream master
```

6. Push the changes. Another way to push the changes is to right click on the server and click **Publish**.

```
$ git push
```

7. After the changes have been pushed, refresh the browser to see the kitchensink application



5.3.5 Collecting Information from the Application (user)

The developer has now deployed a few applications, with the rhc tools, console and, the JBoss Developer studio. To check the status of the application, use the command line tools. The following steps demonstrate how to get the URL for the application, status of the application as well as how to ssh directly into a gear and get some information.

1. Set up the rhc environment so that the rhc credentials are remembered

```
$ echo 'PASSWORD' > /home/scollier0/.openshift/pass.txt
```

```
$ cat <<EOF >> ~/.bash_profile
function rhc() {
  `which rhc` "$@" -p "`cat ~/.openshift/pass.txt`"
}
EOF
```

2. Source the .bash_profile

```
$ source ~/.bash_profile
```

3. Get the URL for the refarchjboss application that was created with JBDS

```
$ rhc app show refarchjboss
```

```
refarchjboss @ http://refarchjboss-
refarch.osop.cloud.lab.eng.bos.redhat.com/
=====
Application Info
=====
SSH URL    = ssh://18c63d2426034d0c88e56ec3351202b6@refarchjboss-
refarch.osop.cloud.lab.eng.bos.redhat.com
Git URL    =
ssh://18c63d2426034d0c88e56ec3351202b6@refarchjboss-
refarch.osop.cloud.lab.eng.bos.redhat.com/~/.git/refarchjboss.g
it/
UUID       = 18c63d2426034d0c88e56ec3351202b6
Created    = 8:32 AM
Gear Size  = small
Cartridges
=====
jbosseap-6.0
```

4. Get logs for the refarchjboss application. The following command is a “tail -f” of the log files. CTRL-C to exit the reading.

```
$ rhc tail refarchjboss
==> jbosseap-6.0/logs/boot.log <==
08:57:41,918 INFO [org.jboss.remoting] JBoss Remoting version 3.2.8.GA-
redhat-1
08:57:42,344 INFO [org.jboss.as.configadmin] JBAS016200: Activating
ConfigAdmin Subsystem
08:57:42,616 INFO [org.jboss.as.clustering.infinispan] JBAS010280:
Activating Infinispan subsystem.
08:57:42,629 INFO [org.jboss.as.jacorb] JBAS016300: Activating JacORB
```



```
Subsystem
08:57:42,738 INFO [org.jboss.as.clustering.jgroups] JBAS010260: Activating
JGroups subsystem.
08:57:42,930 INFO [org.jboss.as.naming] JBAS011800: Activating Naming
Subsystem
08:57:42,938 INFO [org.jboss.as.osgi] JBAS011906: Activating OSGi Subsystem
08:57:43,043 INFO [org.jboss.as.security] JBAS013101: Activating Security
Subsystem
08:57:43,630 INFO [org.jboss.as.webservices] JBAS015537: Activating
WebServices Extension
08:57:43,823 INFO [org.jboss.as.logging] JBAS011502: Removing bootstrap log
handlers
^C
RESULT:
Terminating...
```

5. Access the gear via ssh, use `rhc app show` to get the ssh login information. The important thing to note is the UID for the gear that is used to ssh in. Access the logs.

```
$ ssh 18c63d2426034d0c88e56ec3351202b6@refarchjboss-
refarch.osop.cloud.lab.eng.bos.redhat.com
```

```
*****
```

```
You are accessing a service that is for use only by authorized users.
If you do not have authorization, discontinue use at once.
Any use of the services is subject to the applicable terms of the
agreement which can be found at:
https://openshift.redhat.com/app/legal
```

```
*****
```

```
Welcome to OpenShift shell
```

```
This shell will assist you in managing OpenShift applications.
```

```
!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
Shell access is quite powerful and it is possible for you to
accidentally damage your application. Proceed with care!
If worse comes to worst, destroy your application with 'rhc app destroy'
and recreate it
!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
```

```
Type "help" for more info.
```

```
[refarchjboss-refarch.osop.cloud.lab.eng.bos.redhat.com ~]\>
```

```
[refarchjboss-refarch.osop.cloud.lab.eng.bos.redhat.com ~]\> cd jbosseap-
6.0/logs/; ls
boot.log  server.log
```

5.3.6 Scaling (user)

1. Create a php application and enable scaling

```
$ rhc app create -a scollierscale -t php-5.3 -s
```



Password: *****

Creating application 'scollierscale'

```
=====
Gear Size: default
Namespace: dec10
Scaling:   yes
Cartridge: php-5.3
```

Your application's domain name is being propagated worldwide (this might take a minute)...

The authenticity of host 'scollierscale-refarch.osop.cloud.lab.eng.bos.redhat.com (10.16.138.5)' can't be established.

RSA key fingerprint is 18:e8:43:60:8d:5c:47:87:87:0f:8a:36:f3:20:79:c1.

Are you sure you want to continue connecting (yes/no)? **yes**

Initialized empty Git repository in /home/scollier0/scollierscale/.git/
done

scollierscale @ http://scollierscale-refarch.osop.cloud.lab.eng.bos.redhat.com/

=====

Application Info

=====

Git URL =

ssh://e13f2b2911724d22848acd6739425de1@scollierscale-refarch.osop.cloud.lab.eng.bos.redhat.com/~/.git/scollierscale.git/

Created = 10:50 AM

UUID = e13f2b2911724d22848acd6739425de1

Gear Size = small

SSH URL = ssh://e13f2b2911724d22848acd6739425de1@scollierscale-refarch.osop.cloud.lab.eng.bos.redhat.com

Cartridges

=====

haproxy-1.4

php-5.3

Scaling Info

=====

Scaled x2 (minimum: 2, maximum: available gears) with haproxy-1.4 on small gears

RESULT:

Application scollierscale was created.

Note here how it is noted as being scaled at x2 on creation. One gear is a application gear for php and the other gear is for haproxy.

2. Log into the gear and perform a manual scale

\$ ssh e13f2b2911724d22848acd6739425de1@scollierscale-refarch.osop.cloud.lab.eng.bos.redhat.com



You are accessing a service that is for use only by authorized users.
If you do not have authorization, discontinue use at once.
Any use of the services is subject to the applicable terms of the
agreement which can be found at:
<https://openshift.redhat.com/app/legal>

Welcome to OpenShift shell

This shell will assist you in managing OpenShift applications.

!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
Shell access is quite powerful and it is possible for you to
accidentally damage your application. Proceed with care!
If worse comes to worst, destroy your application with 'rhc app destroy'
and recreate it
!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!

Type "help" for more info.

> haproxy_ctlld -u

There is no output here after the manual scale.

3. Go back to the client and confirm the scale took place

\$ rhc app show scollierscale

Password: *****

scollierscale @ http://scollierscale-refarch
.osop.cloud.lab.eng.bos.redhat.com/

```
=====
Application Info
=====
Git URL      =
ssh://e13f2b2911724d22848acd6739425de1@scollierscale-refarch
.osop.cloud.lab.eng.bos.redhat.com/~/.git/scollierscale.g
it/
SSH URL      = ssh://e13f2b2911724d22848acd6739425de1@scollierscale-refarch
.osop.cloud.lab.eng.bos.redhat.com
Created      = 10:50 AM
UUID         = e13f2b2911724d22848acd6739425de1
Gear Size    = small
Cartridges
=====
php-5.3
haproxy-1.4
Scaling Info
=====
Scaled x3 (minimum: 2, maximum: available gears) with haproxy-1.4 on
small gears
```

Note here how it is now scaled to x3



4. The number of apps can be shown by connecting to the gear and displaying applications

```
> pwd
/var/lib/openshift/9d344bbb2628422482845c5a6678c075/haproxy-1.4/conf

> cat gear-registry.db
b2a88aec117c4e938f261798f146307c@10.16.138.30:php-5.3;b2a88aec11-
refarch.osop.cloud.lab.eng.bos.redhat.com
189fa316f1524dc3897a8cd6c9c37b00@10.16.138.29:php-5.3;189fa316f1-
refarch.osop.cloud.lab.eng.bos.redhat.com
```

Here it is shown that haproxy is load balancing to 2 application gears.

5. Scale up another application

```
> haproxy_ctlld -u
```

6. Check the number of gears now both when logged into the gear and with the rhc command line tools.

```
> cat gear-registry.db
b2a88aec117c4e938f261798f146307c@10.16.138.30:php-5.3;b2a88aec11-
refarch.osop.cloud.lab.eng.bos.redhat.com
189fa316f1524dc3897a8cd6c9c37b00@10.16.138.29:php-5.3;189fa316f1-
refarch.osop.cloud.lab.eng.bos.redhat.com
bd7618406e9541109f476344f63ed62d@10.16.138.29:php-5.3;bd7618406e-
refarchone.osop.cloud.lab.eng.bos.redhat.com
```

```
$ rhc app show scollierscale
```

```
scollierscale @ http://scollierscale-
refarch.osop.cloud.lab.eng.bos.redhat.com/
```

```
=====
=====
```

```
Application Info
```

```
=====
```

```
Git URL =
```

```
ssh://9d344bbb2628422482845c5a6678c075@scollierscale-
refarch.osop.cloud.lab.eng.bos.redhat.com/~/.git/scolliersc
ale.git/
```

```
SSH URL = ssh://9d344bbb2628422482845c5a6678c075@scollierscale-
refarch.osop.cloud.lab.eng.bos.redhat.com
```

```
UUID = 9d344bbb2628422482845c5a6678c075
```

```
Created = 10:44 AM
```

```
Gear Size = small
```

```
Cartridges
```

```
=====
```

```
haproxy-1.4
```

```
php-5.3
```

```
Scaling Info
```

```
=====
```

```
Scaled x4 (minimum: 2, maximum: available gears) with haproxy-1.4 on
small gears
```

Here it is scaled to 4 gears, 1 haproxy gear and 3 application gears.



7. Wait ~5 minutes, issue the same command and notice the automatic scale down

```
$ rhc app show scollierscale
```

```
Password: *****
```

```
scollierscale @ http://scollierscale-refarch
.osop.cloud.lab.eng.bos.redhat.com/
```

```
=====
Application Info
=====
SSH URL    = ssh://e13f2b2911724d22848acd6739425de1@scollierscale-refarch
.osop.cloud.lab.eng.bos.redhat.com
UUID       = e13f2b2911724d22848acd6739425de1
Git URL    =
ssh://e13f2b2911724d22848acd6739425de1@scollierscale-refarch
.osop.cloud.lab.eng.bos.redhat.com/~/.git/scollierscale.g
it/
Gear Size  = small
Created    = 10:50 AM
Cartridges
=====
php-5.3
haproxy-1.4
Scaling Info
=====
Scaled x2 (minimum: 2, maximum: available gears) with haproxy-1.4 on
small gears
```

Note how the application is now scaled back down to x2 gears.

5.3.7 Jenkins Enablement (user)

This section shows how to enable a application with jenkins and what to do with it once jenkins is enabled. Some of the benefits of using jenkins:

- Application stays available while the jenkins build is going on
- Enables formal environment for running unit tests on code
- Prevents an application from being deployed if it fails a test which prevents disruption of the production application
 - Must pass build
 - Must pass unit tests (if enabled)

The Spring Framework on OpenShift is a great example here. It is available as a quickstart on github¹². The steps that are followed here:

- Deploy the quickstart
- Demonstrate successful build with no unit tests
- Demonstrate successful build with unit tests
- Demonstrate failure with unit tests

Deploy the Quickstart



1. Following the quickstart noted above, create the application. Perform the following tasks as a developer. The output from the following commands is omitted.

```
$ rhc app create -a spring -t jbosseap-6.0  
  
$ cd spring  
  
$ git remote add upstream -m master git://github.com/fabianofranz/spring-  
eap6-quickstart.git  
  
$ git pull -s recursive -X theirs upstream master  
  
$ git rm src/main/webapp/index.html  
  
$ git commit -m 'Removed default index.html'  
  
$ git push  
  
$ rhc app show spring
```

2. Change the database to enable support for MySQL.

```
$ sed -i 's/ExampleDS/MysqlDS/' src/main/resources/META-INF/persistence.xml
```

3. Create a jenkins cartridge. Note the jenkins user and password – this is required to access the jenkins web based console. As of the writing of this paper, there is a bugzilla¹³ open that concerns the options with regard to creating a jenkins enabled application.

```
$ rhc app create -a jenkins -t jenkins-1.4
```

4. Associate the jenkins cartridge with the spring application

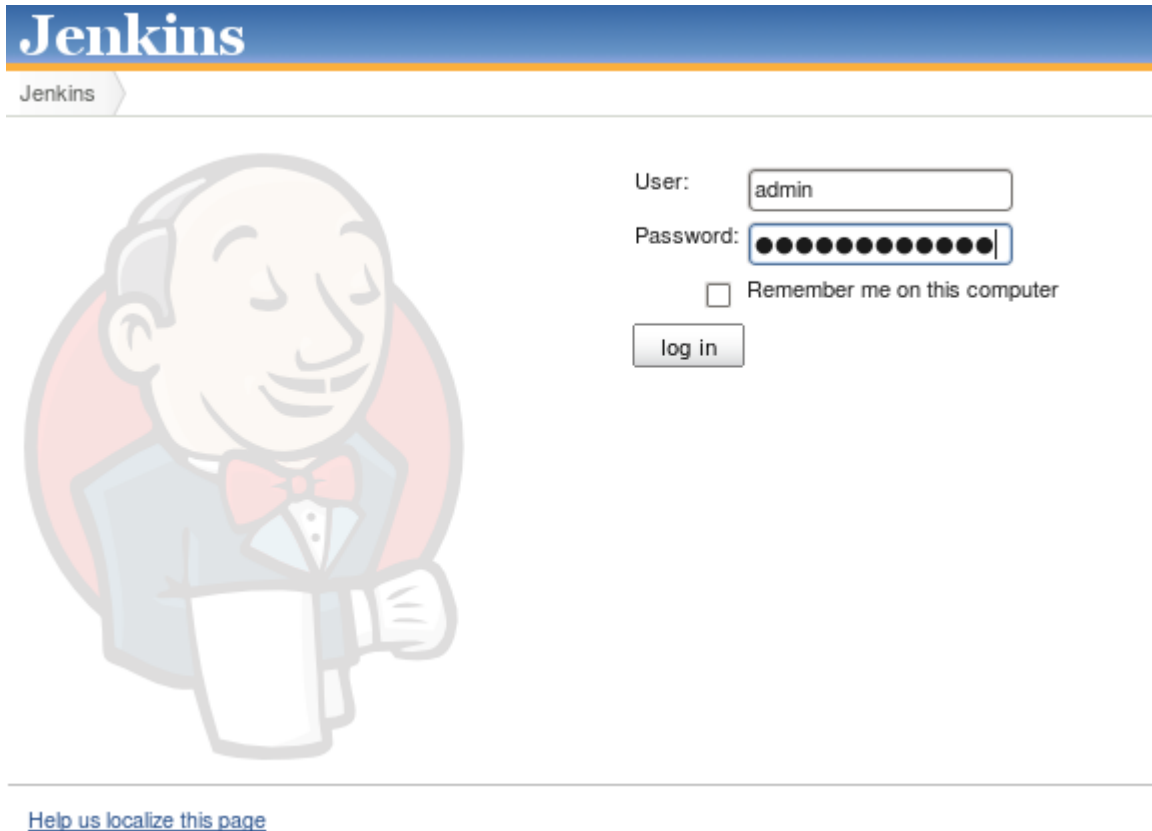
```
$ rhc cartridge add --cartridge jenkins-client-1.4 --app spring
```



Demonstrate successful build with no unit tests

1. Open a browser and access jenkins and issue a build without unit tests enabled. By default, this is the behavior for this application.

<http://jenkins-refarch.osop.cloud.lab.eng.bos.redhat.com/>



The image shows the Jenkins login interface. At the top, there is a blue header with the word "Jenkins" in white. Below the header, there is a navigation bar with the word "Jenkins" in a grey box. The main content area features a large, stylized cartoon illustration of a man in a tuxedo and bow tie on the left. To the right of the illustration, there is a login form. The form includes a "User:" label followed by a text input field containing the text "admin". Below this is a "Password:" label followed by a password input field with black dots. Under the password field, there is a checkbox labeled "Remember me on this computer". At the bottom of the form is a "log in" button. At the very bottom of the page, there is a link that says "Help us localize this page".

Figure 5.3.7.1: Jenkins Log On Screen



(a) Click on the **Build Now** link to initiate a build

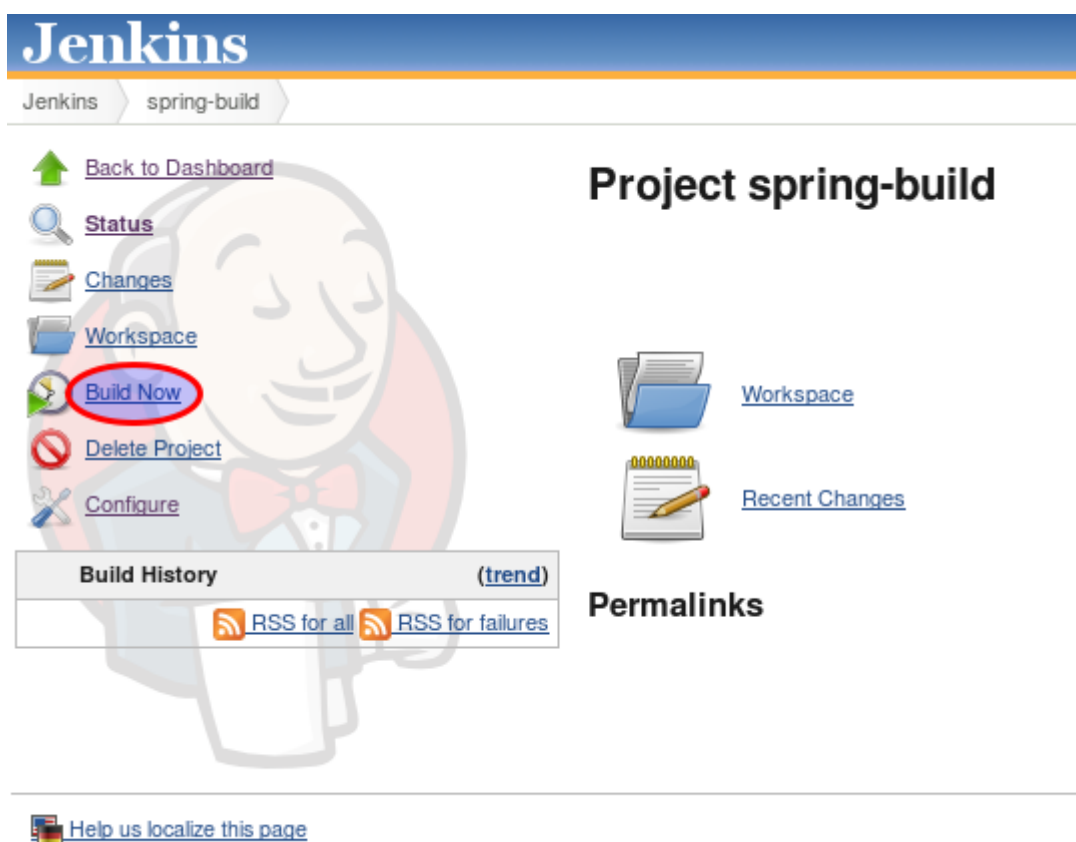


Figure 5.3.7.2: Jenkins Build Now

(b) **ENABLE AUTO REFRESH** in the upper right location on the screen



Figure 5.3.7.3: Jenkins Auto-refresh



(c) Watch the build on the left hand navigation panel

Jenkins

Jenkins spring-build

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now!](#)

[Delete Project](#)

[Configure](#)

Project spring-build

[Workspace](#)

[Recent Changes](#)

Build History (trend)

#1 (pending - Waiting for next available executor)	
--	--

[RSS for all](#) [RSS for failures](#)

Permalinks

[Help us localize this page](#)

Figure 5.3.7.4: Jenkins Build



- (d) View a successful build by checking the **Build History** status in the left hand navigation panel

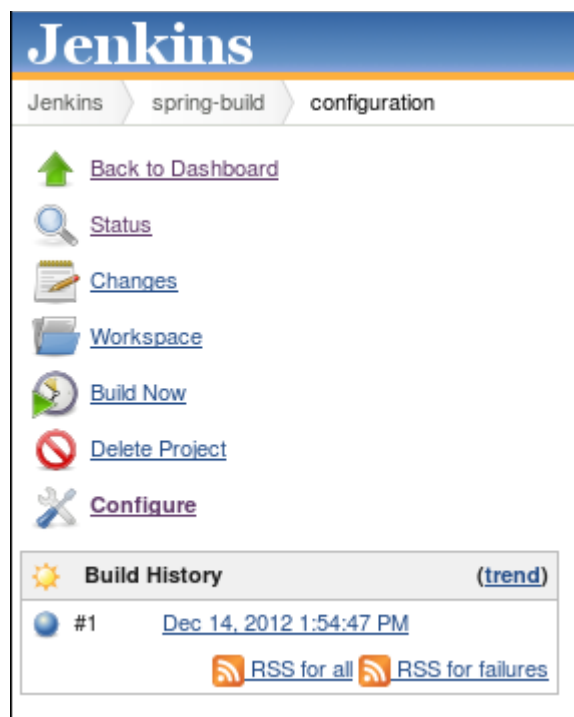


Figure 5.3.7.5: Successful Build



Demonstrate successful build with unit tests

1. Browse to the **Project spring-build** home page within jenkins
2. Enable unit tests
 - (a) Remove skip test flag from configuration
 - i. Click **Configure**
 - ii. Scroll down to the configure script and remove the `-DskipTests` flag and click save

```
mvn --global-settings $OPENSIFT_MAVEN_MIRROR clean package -Popenshift -DskipTests
```

Build

Execute shell

Command

```
source /usr/libexec/openshift/cartridges/abstract/info/lib/jenkins_util

jenkins_rsync ca3038fe01fb4e81959404249edlca9c@spring-refarchone.osop.cloud.lab.eng.bos.redhat.com:~/m2/ ~/m2/

# Build setup and run user pre_build and build
. ci_build.sh

if [ -e ${OPENSIFT_REPO_DIR}.openshift/markers/java7 ];
then
    export JAVA_HOME=/etc/alternatives/java_sdk_1.7.0
else
    export JAVA_HOME=/etc/alternatives/java_sdk_1.6.0
fi

export MAVEN_OPTS="$OPENSIFT_MAVEN_XMX"
mvn --global-settings $OPENSIFT_MAVEN_MIRROR --version
mvn --global-settings $OPENSIFT_MAVEN_MIRROR clean package -Popenshift -DskipTests

# Deploy new build

# Stop app
jenkins_stop_app ca3038fe01fb4e81959404249edlca9c@spring-refarchone.osop.cloud.lab.eng.bos.redhat.com

# Push content back to application
jenkins_sync_iboss ca3038fe01fb4e81959404249edlca9c@spring-refarchone.osop.cloud.lab.eng.bos.redhat.com

# Configure / start app
$GIT_SSH ca3038fe01fb4e81959404249edlca9c@spring-refarchone.osop.cloud.lab.eng.bos.redhat.com deploy.sh

jenkins_start_app ca3038fe01fb4e81959404249edlca9c@spring-refarchone.osop.cloud.lab.eng.bos.redhat.com

$GIT_SSH ca3038fe01fb4e81959404249edlca9c@spring-refarchone.osop.cloud.lab.eng.bos.redhat.com post_deploy.sh
```

Figure 5.3.7.6: Jenkins Configure



- iii. Clicking save returns to the **Project spring-build** application home page
- (b) Run build, check for success by reviewing logs
 - i. Click **Build Now**
 - ii. Watch the progress on the left hand navigation panel
 - iii. When the build is complete, check the logs.
 - A. Hover over the last build job with tests enabled and click **CONSOLE** on the pop up menu

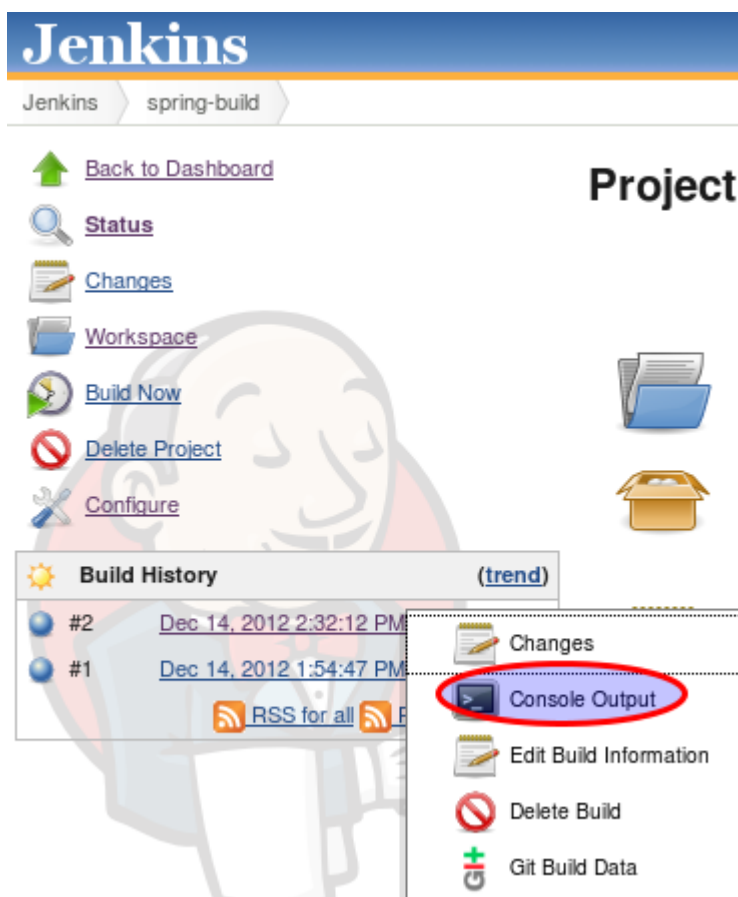


Figure 5.3.7.7: Jenkins Console

- B. In the Console Output screen, scroll down until the results of the tests are shown

```
Results :  
  
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
```

Figure 5.3.7.8: Test Results

Demonstrate successful build with unit tests failures

1. Force a failure in one of the unit tests



(a) From the client shell, navigate to the root of the git repository and open the test file and review it

```
$ vi src/test/java/org/jboss/tools/example/springmvc/test/MemberDaoTest.java
```

(b) There are 4 tests that run

- testFindByID
- testFindByEmail
- TestRegister
- TestFindAllOrderedByName

(c) The test is checking for the existence of a user named “John Smith”, change that to “John Test”

```
$ sed -i 's/Smith/Test/'  
src/test/java/org/jboss/tools/example/springmvc/test/MemberDaoTest.java
```

(d) Commit the changes

```
$ git commit -a -m 'Changed name from Smith to Test to induce failure'  
[master 7ed029b] Changed name from Smith to Test to induce failure  
1 files changed, 2 insertions(+), 2 deletions(-)
```

(e) Watch the build on both shell and console

i. Watch the shell for failure

```
$ git push  
Warning: the RSA host key for 'spring-  
refarchone.osop.cloud.lab.eng.bos.redhat.com' differs from the key for the  
IP address '10.16.138.29'  
Offending key for IP in /home/scollier0/.ssh/known_hosts:27  
Matching host key in /home/scollier0/.ssh/known_hosts:39  
Are you sure you want to continue connecting (yes/no)? yes  
Counting objects: 23, done.  
Delta compression using up to 2 threads.  
Compressing objects: 100% (5/5), done.  
Writing objects: 100% (12/12), 788 bytes, done.  
Total 12 (delta 2), reused 0 (delta 0)  
remote: restart_on_add=false  
remote: Executing Jenkins build.  
remote:  
remote: You can track your build at https://jenkins-  
refarchone.osop.cloud.lab.eng.bos.redhat.com/job/spring-build  
remote:  
remote: Waiting for build to schedule....Done  
remote: Waiting for job to  
complete.....  
.....Done  
remote: FAILED  
remote: !!!!!!!  
remote: Deployment Halted!  
remote: If the build failed before the deploy step, your previous  
remote: build is still running. Otherwise, your application may be  
remote: partially deployed or inaccessible.
```



```
remote: Fix the build and try again.  
remote: !!!!!!!!  
To ssh://ca3038fe01fb4e81959404249ed1ca9c@spring-  
refarchone.osop.cloud.lab.eng.bos.redhat.com/~git/spring.git/  
39cf630..7ed029b master -> master
```

- ii. Watch the **Project spring-build** page and notice the build on the left hand navigation screen and notice the color of the build icon by the failed build

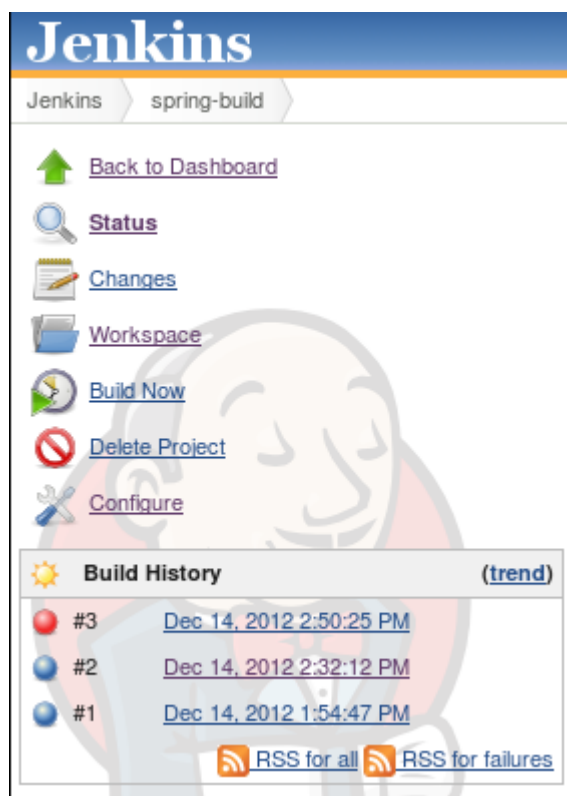


Figure 5.3.7.9: Failed Job

- iii. Browse the logs again and look at the failure notification

```
Tests run: 4, Failures: 2, Errors: 0, Skipped: 0
```

Figure 5.3.7.10: Jenkins Failure

Jenkins did not publish the application. Upon failure, the publication was canceled and the developer has a chance to review the logs, see which unit test failed and then make adjustments. No downtime was experienced.



5.3.8 Clean Up

This section shows how to remove applications from both the administrator and developer perspective. Make sure to back up any important data before proceeding.

As a developer, remove an application:

1. Backup any data that is important to you before following these steps.

2. Show the application

```
$ rhc app show phpapp
```

```
phpapp @ http://phpapp-refarchone.osop.cloud.lab.eng.bos.redhat.com/
=====
Application Info
=====
Created    = 9:45 AM
Gear Size  = small
Git URL    =
ssh://4a5c80136d0140db8b65de90dcd08e23@phpapp-
refarchone.osop.cloud.lab.eng.bos.redhat.com/~/.git/phpapp.git/
UUID       = 4a5c80136d0140db8b65de90dcd08e23
SSH URL    = ssh://4a5c80136d0140db8b65de90dcd08e23@phpapp-
refarchone.osop.cloud.lab.eng.bos.redhat.com
Cartridges
=====
php-5.3
```

3. Delete the application

```
$ rhc app delete phpapp
```

```
Are you sure you wish to delete the 'phpapp' application? (yes/no)
yes
```

```
Deleting application 'phpapp'
```

```
RESULT:
```

```
Application 'phpapp' successfully deleted
```

4. This does not remove any local git repository directories. If desired, clean those up manually.

5. Show the application

```
$ rhc app show phpapp
```

```
Application phpapp does not exist
```

As an administrator, remove an application:

1. On the broker host, list the applications for the user

```
# oo-admin-ctl-domain -l scollier | egrep -i '^name|^Namespace:'
Namespace: refarch
name: refarchjboss
name: scollierscale
```



```
name: phpapp  
name: phpapptmp
```

2. Get the status of the application

```
# oo-admin-ctl-app -l scollier -a phpapptmp -c status
```

```
Total Accesses: 0  
Total kBytes: 0  
Uptime: 686  
ReqPerSec: 0  
BytesPerSec: 0  
BusyWorkers: 1  
IdleWorkers: 0  
Scoreboard: W...
```

3. Stop the application

```
# oo-admin-ctl-app -l scollier -a phpapptmp -c stop  
Success
```

4. Get the status of the application

```
# oo-admin-ctl-app -l scollier -a phpapptmp -c status  
Application 'phpapptmp' is either stopped or inaccessible
```

5. Destroy the application

```
# oo-admin-ctl-app -l scollier -a phpapptmp -c destroy  
!!!! WARNING !!!! WARNING !!!! WARNING !!!!  
You are about to destroy the phpapptmp application.
```

This is NOT reversible, all remote data for this application will be removed.

```
Do you want to destroy this application (y/n): y  
Successfully destroyed application: phpapptmp
```

6. From the developers view, check the app with the rhc tools

```
$ rhc app show phpapptmp  
Application phpapptmp does not exist
```



5.4 Node Gear & District Management

An OpenShift district defines a set of node hosts and the resource definitions they must share in order to enable transparent migration of gears between hosts. While districts are not strictly required, they are simple to use and any production installation can benefit from them. OpenShift administration guides generally assume that districts are in use.

The main purpose of districts is to allocate gear resources identically across all node hosts in the district, allowing a gear to keep its same UID (and related IP addresses, ports, etc.) when moved between any node hosts within the district. Thus, users' apps will not be disrupted by changes when gears are migrated between node hosts within the district, even if they have hard-coded specific values into their apps (as inevitably they do) rather than sourcing environment variables.

Gears are containers with a set of resources that allow users to run applications. This section shows how to migrate from a configuration that doesn't contain a district to a configuration that contains two districts as shown in **Figure 5.4.1: Districts**.

- District one – Two Nodes – Small Gears
- District two – One Node – Medium Gears

There are different ways to configure districts withing OpenShift. One way is to know what you need before you deploy the environment and configure everything upfront. Fortunately OpenShift is flexible enough to allow for changes later – which is the method this reference architecture is using. At the end of this section there will be two districts that can run either small or medium gears.

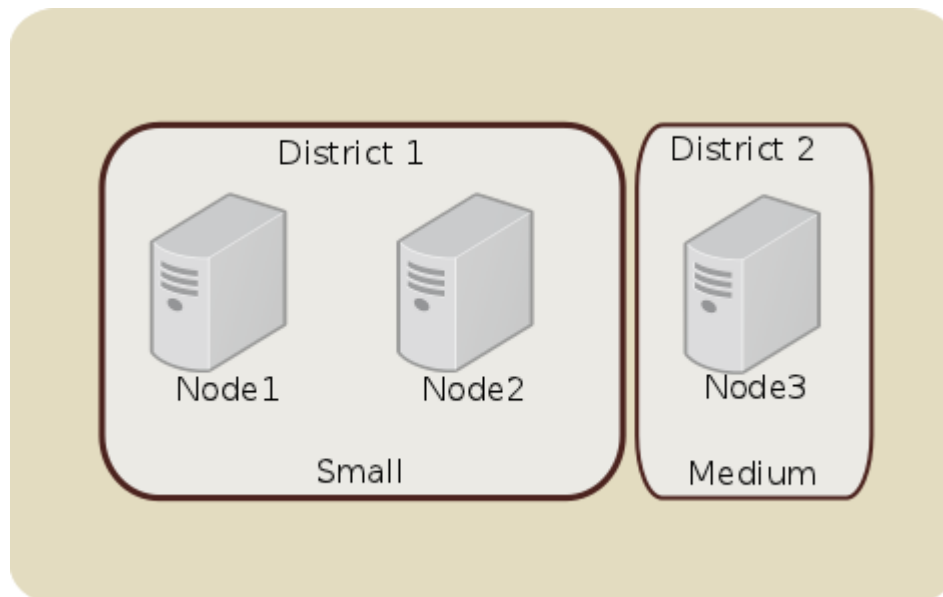


Figure 5.4.1: Districts



5.4.1 Add a Node to the Infrastructure

In order to meet the requirements of multiple districts for this reference architecture another node needs to be installed. Follow the steps in **4.5 Deploy Nodes**.

- Confirm that the new node shows up in the messaging infrastructure. Log into the broker to show this.

```
# mco ping
node2.osop.cloud.lab.eng.bos.redhat.com  time=51.46 ms
node1.osop.cloud.lab.eng.bos.redhat.com  time=91.91 ms
node3.osop.cloud.lab.eng.bos.redhat.com  time=93.50 ms

---- ping statistics ----
3 replies max: 93.50 min: 51.46 avg: 78.95
```

5.4.2 Enabling Node Profiles

Each node can only run one type of gear. This reference architecture has three nodes, two of them host applications that run in small gears and the third node hosts applications that run in medium gears. By default, the brokers support small and medium gears with a default set to small.

1. Confirm that node1 and node2 are configured to run small gears

```
# grep small /etc/openshift/resource_limits.conf
node_profile=small
```

2. Configure node3 with a medium profile

```
# sed -i 's/small/medium/' /etc/openshift/resource_limits.conf
```

- (a) Change the *MEMORY_LIMIT_IN_BYTES* restriction for this profile from 512MB to 1GB.

```
# echo "(1024^3)" | bc
1073741824

# grep memory_limit /etc/openshift/resource_limits.conf
# For no over commitment, should be memory_limit_in_bytes / (Total System
Memory - 1G)
memory_limit_in_bytes=1073741824
```

- (b) Add another 100MB for swapping to the *MEMORY_MEMSW_LIMIT_IN_BYTES* option.

```
# echo "(1024^3) + (1024^2 * 100)" | bc
1178599424

# grep memory_mem /etc/openshift/resource_limits.conf
memory_memsw_limit_in_bytes=1178599424 # 1024M + 100M (100M swap)
```

- (c) Reboot node3 for these changes to take effect.

5.4.3 Districts

Enable two districts. One for hosts that run small gears (node1 and node2) and one for hosts that run medium gears (node3).



5.4.3.1 Create Districts

1. Confirm current district configuration, perform this from a broker

```
# oo-admin-ctl-district
```

No districts created yet. Use 'oo-admin-ctl-district -c create' to create one.

2. Create the small district

```
# oo-admin-ctl-district --command create --name small --node_profile small  
Successfully created district: 20e6c23321974a138d978c2bfe52a9f6
```

```
{"externally_reserved_uids_size"=>0,  
"name"=>"small",  
"max_capacity"=>6000,  
"creation_time"=>"2012-12-13T14:03:09-05:00",  
"available_capacity"=>6000,  
"available_uids"=>"<6000 uids hidden>",  
"node_profile"=>"small",  
"uuid"=>"20e6c23321974a138d978c2bfe52a9f6",  
"max_uid"=>6999,  
"server_identities"=>{},  
"active_server_identities_size"=>0}
```

3. Create the medium district

```
# oo-admin-ctl-district --command create --name medium --node_profile medium  
Successfully created district: 491dd23b61d346738a5b131740b2b9fa
```

```
{"max_uid"=>6999,  
"active_server_identities_size"=>0,  
"available_uids"=>"<6000 uids hidden>",  
"node_profile"=>"medium",  
"externally_reserved_uids_size"=>0,  
"max_capacity"=>6000,  
"creation_time"=>"2012-12-13T14:06:44-05:00",  
"uuid"=>"491dd23b61d346738a5b131740b2b9fa",  
"available_capacity"=>6000,  
"server_identities"=>{},  
"name"=>"medium"}
```

4. View both districts

```
# oo-admin-ctl-district
```

```
{"available_capacity"=>6000,  
"available_uids"=>"<6000 uids hidden>",  
"externally_reserved_uids_size"=>0,  
"max_capacity"=>6000,  
"active_server_identities_size"=>0,  
"server_identities"=>{},  
"name"=>"small",  
"node_profile"=>"small",  
"uuid"=>"20e6c23321974a138d978c2bfe52a9f6",  
"creation_time"=>"2012-12-13T14:03:09-05:00",
```



```
"max_uid"=>6999}

{"available_capacity"=>6000,
 "available_uids"=>"<6000 uids hidden>",
 "externally_reserved_uids_size"=>0,
 "max_capacity"=>6000,
 "active_server_identities_size"=>0,
 "server_identities"=>{},
 "name"=>"medium",
 "node_profile"=>"medium",
 "uuid"=>"491dd23b61d346738a5b131740b2b9fa",
 "creation_time"=>"2012-12-13T14:06:44-05:00",
 "max_uid"=>6999}
```

5.4.3.2 Move Applications and Add a Node to District

One of the requirements for adding nodes to a district is that the node cannot be running any applications. This section demonstrates how to move applications from node1 to node2 and add node1 to the small district and then move all the applications back to node1 and add node2 to the small district.

1. Try to add node1 to the small district

```
# oo-admin-ctl-district --command add-node --name small --server_identity
node1.osop.cloud.lab.eng.bos.redhat.com
ERROR OUTPUT:
Node with server identity: node1.osop.cloud.lab.eng.bos.redhat.com already
has apps on it
```

2. From node1, confirm node1 has a gear running

```
# oo-admin-ctl-gears list
e8714e9fcdc043568a8130db09fc65e9
```

3. From node2, confirm node2 has a gear running

```
# oo-admin-ctl-gears list
16a44b8dc6b34e6e9cbe1d3bf0fc37d8
```

4. From a broker, move the gear from node1 to node2

```
# oo-admin-move --target_server_identity
node2.osop.cloud.lab.eng.bos.redhat.com --gear_uuid
e8714e9fcdc043568a8130db09fc65e9
URL: http://moveapp-refarch.osop.cloud.lab.eng.bos.redhat.com
Login: scollier
App UUID: e8714e9fcdc043568a8130db09fc65e9
Gear UUID: e8714e9fcdc043568a8130db09fc65e9
DEBUG: Source district uuid: NONE
DEBUG: Destination district uuid: NONE
DEBUG: Getting existing app 'moveapp' status before moving
DEBUG: Gear component 'php-5.3' was running
DEBUG: Stopping existing app cartridge 'php-5.3' before moving
DEBUG: Force stopping existing app cartridge 'php-5.3' before moving
DEBUG: Reserved uid ' ' on district: 'NONE'
DEBUG: Creating new account for gear 'moveapp' on
```




```
node2.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Moving content for app 'moveapp', gear 'moveapp' to
node2.osop.cloud.lab.eng.bos.redhat.com
Identity added: /etc/openshift/rsync_id_rsa (/etc/openshift/rsync_id_rsa)
Warning: Permanently added '10.16.138.29' (RSA) to the list of known hosts.
Warning: Permanently added '10.16.138.30' (RSA) to the list of known hosts.
Agent pid 12063
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 12063 killed;
DEBUG: Performing cartridge level move for 'php-5.3' on
node2.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Starting cartridge 'php-5.3' in 'moveapp' after move on
node2.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Fixing DNS and mongo for gear 'moveapp' after move
DEBUG: Changing server identity of 'moveapp' from
'node1.osop.cloud.lab.eng.bos.redhat.com' to
'node2.osop.cloud.lab.eng.bos.redhat.com'
DEBUG: Deconfiguring old app 'moveapp' on
node1.osop.cloud.lab.eng.bos.redhat.com after move
Successfully moved 'moveapp' with gear uuid
'e8714e9fcdc043568a8130db09fc65e9' from
'node1.osop.cloud.lab.eng.bos.redhat.com' to
'node2.osop.cloud.lab.eng.bos.redhat.com'
```

5. From node2, confirm node2 has two gears running

```
# oo-admin-ctl-gears list
99a41724319741bba03a3784aa8efa04
e8714e9fcdc043568a8130db09fc65e9
```

6. From node1, confirm node1 does not have any gears running.

```
# oo-admin-ctl-gears list
```

7. Add node1 to the small district

```
# oo-admin-ctl-district --command add-node --name small --server_identity
node1.osop.cloud.lab.eng.bos.redhat.com --allow_change_district
Success!
```

```
{"name"=>"small",
"available_uids"=>"<6000 uids hidden>",
"externally_reserved_uids_size"=>0,
"server_identities"=>
{"node1.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true}},
"max_capacity"=>6000,
"available_capacity"=>6000,
"node_profile"=>"small",
"creation_time"=>"2012-12-13T14:03:09-05:00",
"uuid"=>"20e6c23321974a138d978c2bfe52a9f6",
"active_server_identities_size"=>1,
"max_uid"=>6999}
```

8. Move the gears from node2 to node1

```
# oo-admin-move --target_server_identity
```



```
node1.osop.cloud.lab.eng.bos.redhat.com --gear_uuid
e8714e9fcdc043568a8130db09fc65e9
URL: http://moveapp-refarchone.osop.cloud.lab.eng.bos.redhat.com
Login: scollier
App UUID: e8714e9fcdc043568a8130db09fc65e9
Gear UUID: e8714e9fcdc043568a8130db09fc65e9
DEBUG: Source district uuid: NONE
DEBUG: Destination district uuid: NONE
DEBUG: Getting existing app 'moveapp' status before moving
DEBUG: Gear component 'php-5.3' was running
DEBUG: Stopping existing app cartridge 'php-5.3' before moving
DEBUG: Force stopping existing app cartridge 'php-5.3' before moving
DEBUG: Reserved uid '' on district: 'NONE'
DEBUG: Creating new account for gear 'moveapp' on
node1.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Moving content for app 'moveapp', gear 'moveapp' to
node1.osop.cloud.lab.eng.bos.redhat.com
Identity added: /etc/openshift/rsync_id_rsa (/etc/openshift/rsync_id_rsa)
Warning: Permanently added '10.16.138.30' (RSA) to the list of known hosts.
Warning: Permanently added '10.16.138.29' (RSA) to the list of known hosts.
Agent pid 12565
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 12565 killed;
DEBUG: Performing cartridge level move for 'php-5.3' on
node1.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Starting cartridge 'php-5.3' in 'moveapp' after move on
node1.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Fixing DNS and mongo for gear 'moveapp' after move
DEBUG: Changing server identity of 'moveapp' from
'node2.osop.cloud.lab.eng.bos.redhat.com' to
'node1.osop.cloud.lab.eng.bos.redhat.com'
DEBUG: Deconfiguring old app 'moveapp' on
node2.osop.cloud.lab.eng.bos.redhat.com after move
Successfully moved 'moveapp' with gear uuid
'e8714e9fcdc043568a8130db09fc65e9' from
'node2.osop.cloud.lab.eng.bos.redhat.com' to
'node1.osop.cloud.lab.eng.bos.redhat.com'

# oo-admin-move --target_server_identity
node1.osop.cloud.lab.eng.bos.redhat.com --gear_uuid
16a44b8dc6b34e6e9cbe1d3bf0fc37d8 --allow_change_district
URL: http://phpmove2-refarch.osop.cloud.lab.eng.bos.redhat.com
Login: scollier
App UUID: 16a44b8dc6b34e6e9cbe1d3bf0fc37d8
Gear UUID: 16a44b8dc6b34e6e9cbe1d3bf0fc37d8
DEBUG: Source district uuid: NONE
DEBUG: Destination district uuid: 20e6c23321974a138d978c2bfe52a9f6
DEBUG: Getting existing app 'phpmove2' status before moving
DEBUG: Gear component 'php-5.3' was running
DEBUG: Stopping existing app cartridge 'php-5.3' before moving
DEBUG: Force stopping existing app cartridge 'php-5.3' before moving
DEBUG: Reserved uid '1002' on district: '20e6c23321974a138d978c2bfe52a9f6'
DEBUG: Creating new account for gear 'phpmove2' on
```



```
node1.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Moving content for app 'phpmove2', gear 'phpmove2' to
node1.osop.cloud.lab.eng.bos.redhat.com
Identity added: /etc/openshift/rsync_id_rsa (/etc/openshift/rsync_id_rsa)
Warning: Permanently added '10.16.138.30' (RSA) to the list of known hosts.
Warning: Permanently added '10.16.138.29' (RSA) to the list of known hosts.
Agent pid 13575
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 13575 killed;
DEBUG: Performing cartridge level move for 'php-5.3' on
node1.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Starting cartridge 'php-5.3' in 'phpmove2' after move on
node1.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Fixing DNS and mongo for gear 'phpmove2' after move
DEBUG: Changing server identity of 'phpmove2' from
'node2.osop.cloud.lab.eng.bos.redhat.com' to
'node1.osop.cloud.lab.eng.bos.redhat.com'
DEBUG: Deconfiguring old app 'phpmove2' on
node2.osop.cloud.lab.eng.bos.redhat.com after move
Successfully moved 'phpmove2' with gear uuid
'16a44b8dc6b34e6e9cbe1d3bf0fc37d8' from
'node2.osop.cloud.lab.eng.bos.redhat.com' to
'node1.osop.cloud.lab.eng.bos.redhat.com'
```

9. Confirm there are no gears are running on node2

```
# oo-admin-ctl-gears list
```

10. Add node2 to the small district

```
# oo-admin-ctl-district --command add-node --name small --server_identity
node2.osop.cloud.lab.eng.bos.redhat.com
Success!
```

```
{"name"=>"small",
 "active_server_identities_size"=>2,
 "server_identities"=>
  {"node1.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true},
   "node2.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true}},
 "uuid"=>"20e6c23321974a138d978c2bfe52a9f6",
 "available_capacity"=>5999,
 "max_uid"=>6999,
 "node_profile"=>"small",
 "creation_time"=>"2012-12-13T14:03:09-05:00",
 "max_capacity"=>6000,
 "externally_reserved_uids_size"=>0,
 "available_uids"=>"<5999 uids hidden>"}
```

11. Re-balance the applications and move a gear back to node2 and confirm both nodes have one gear each.

```
# oo-admin-move --target_server_identity
node2.osop.cloud.lab.eng.bos.redhat.com --gear_uuid
16a44b8dc6b34e6e9cbe1d3bf0fc37d8
URL: http://phpmove2-refarch.osop.cloud.lab.eng.bos.redhat.com
```



```
Login: scollier
App UUID: 16a44b8dc6b34e6e9cbe1d3bf0fc37d8
Gear UUID: 16a44b8dc6b34e6e9cbe1d3bf0fc37d8
DEBUG: Source district uuid: 20e6c23321974a138d978c2bfe52a9f6
DEBUG: Destination district uuid: 20e6c23321974a138d978c2bfe52a9f6
DEBUG: District unchanged keeping uid
DEBUG: Getting existing app 'phpmove2' status before moving
DEBUG: Gear component 'php-5.3' was running
DEBUG: Stopping existing app cartridge 'php-5.3' before moving
DEBUG: Force stopping existing app cartridge 'php-5.3' before moving
DEBUG: Creating new account for gear 'phpmove2' on
node2.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Moving content for app 'phpmove2', gear 'phpmove2' to
node2.osop.cloud.lab.eng.bos.redhat.com
Identity added: /etc/openshift/rsync_id_rsa (/etc/openshift/rsync_id_rsa)
Warning: Permanently added '10.16.138.29' (RSA) to the list of known hosts.
Warning: Permanently added '10.16.138.30' (RSA) to the list of known hosts.
Agent pid 14088
unset SSH_AUTH_SOCK;
unset SSH_AGENT_PID;
echo Agent pid 14088 killed;
DEBUG: Performing cartridge level move for 'php-5.3' on
node2.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Starting cartridge 'php-5.3' in 'phpmove2' after move on
node2.osop.cloud.lab.eng.bos.redhat.com
DEBUG: Fixing DNS and mongo for gear 'phpmove2' after move
DEBUG: Changing server identity of 'phpmove2' from
'node1.osop.cloud.lab.eng.bos.redhat.com' to
'node2.osop.cloud.lab.eng.bos.redhat.com'
DEBUG: Deconfiguring old app 'phpmove2' on
node1.osop.cloud.lab.eng.bos.redhat.com after move
Successfully moved 'phpmove2' with gear uuid
'16a44b8dc6b34e6e9cbe1d3bf0fc37d8' from
'node1.osop.cloud.lab.eng.bos.redhat.com' to
'node2.osop.cloud.lab.eng.bos.redhat.com'
```

12. Confirm node1 has one gear

```
# oo-admin-ctl-gears list
e8714e9fcdc043568a8130db09fc65e9
```

13. Confirm node2 has one gear

```
# oo-admin-ctl-gears list
16a44b8dc6b34e6e9cbe1d3bf0fc37d8
```

14. Confirm node3 does not have any gears

```
# oo-admin-ctl-gears list
```

15. Add node3 to the medium district

```
# oo-admin-ctl-district --command add-node --name medium --server_identity
node3.osop.cloud.lab.eng.bos.redhat.com
Success!
```



```
{
  "name"=>"medium",
  "active_server_identities_size"=>1,
  "uuid"=>"491dd23b61d346738a5b131740b2b9fa",
  "server_identities"=>
    {
      "node3.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true}},
  "max_capacity"=>6000,
  "creation_time"=>"2012-12-13T14:06:44-05:00",
  "available_capacity"=>6000,
  "externally_reserved_uids_size"=>0,
  "max_uid"=>6999,
  "node_profile"=>"medium",
  "available_uids"=>"<6000 uids hidden>"
}
```

16. View the districts

oo-admin-ctl-district

```
{
  "externally_reserved_uids_size"=>0,
  "max_uid"=>6999,
  "server_identities"=>
    {
      "node2.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true},
      "node1.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true}},
  "available_capacity"=>5999,
  "max_capacity"=>6000,
  "active_server_identities_size"=>2,
  "uuid"=>"20e6c23321974a138d978c2bfe52a9f6",
  "available_uids"=>"<5999 uids hidden>",
  "node_profile"=>"small",
  "name"=>"small",
  "creation_time"=>"2012-12-13T14:03:09-05:00"}

```

```
{
  "externally_reserved_uids_size"=>0,
  "max_uid"=>6999,
  "server_identities"=>
    {
      "node3.osop.cloud.lab.eng.bos.redhat.com"=>{"active"=>true}},
  "available_capacity"=>6000,
  "max_capacity"=>6000,
  "active_server_identities_size"=>1,
  "uuid"=>"491dd23b61d346738a5b131740b2b9fa",
  "available_uids"=>"<6000 uids hidden>",
  "node_profile"=>"medium",
  "name"=>"medium",
  "creation_time"=>"2012-12-13T14:06:44-05:00"}

```

Note which nodes are in which districts.



5.5 Users

Now that the environment is fully configured, this section shows how to deploy an application to the district that supports medium gears. Recall, by default all applications are created as small gears.

1. From the broker, display current gear information for user

```
# oo-admin-ctl-user -l scollier
```

```
User scollier:
  consumed gears: 2
    max gears: 100
  gear sizes: small
```

2. Enable medium sized gears for the user

```
# oo-admin-ctl-user -l scollier --addgearsizes medium
```

```
Adding gear size medium for user scollier... Done.
```

```
User scollier:
  consumed gears: 2
    max gears: 100
  gear sizes: small, medium
```

3. From the client, deploy a medium sized application

```
$ rhc app create phpmedium php-5.3 --gear-size medium
```

```
Creating application 'phpmedium'
```

```
=====
```

```
Cartridge: php-5.3
Gear Size: medium
Namespace: refarch
Scaling: no
```

```
Your application's domain name is being propagated worldwide (this might take a minute)...
```

```
The authenticity of host 'phpmedium-refarch.osop.cloud.lab.eng.bos.redhat.com (10.16.138.57)' can't be established.
```

```
RSA key fingerprint is c0:f0:1a:3c:93:6c:08:68:f3:f5:f5:4e:52:50:c2:ea.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Initialized empty Git repository in /home/scollier0/phpmedium/.git/
done
```

```
phpmedium @ http://phpmedium-refarch.osop.cloud.lab.eng.bos.redhat.com/
```

```
=====
```

```
Application Info
=====
```



```

Created      = 3:12 PM
UUID        = e775030f960b4148923e158a411f273a
Gear Size   = medium
SSH URL     = ssh://e775030f960b4148923e158a411f273a@phpmedium-
refarchone.osop.cloud.lab.eng.bos.redhat.com
Git URL     =
ssh://e775030f960b4148923e158a411f273a@phpmedium-
refarch.osop.cloud.lab.eng.bos.redhat.com/~git/phpmedium.git/
Cartridges
=====
  php-5.3

```

RESULT:
Application phpmedium was created.

4. Confirm the gear is located on node3 , from node3:

```

# oo-admin-ctl-gears list
e775030f960b4148923e158a411f273a

```

5. Now associate that UID with a application name, from node3

```

# ll /var/lib/openshift/.httpd.d/
total 8
drwxr-xr-x. 2 root root 4096 Dec 13 15:12
e775030f960b4148923e158a411f273a_refarch_phpmedium

```

Another way to find out which user owns that particular application is by creating the following Ruby script on a broker called `/root/find_user.rb` and making it executable.

1. Create script and make executable

```

#!/usr/bin/ruby
## run this on broker machine
gear_uuid = ARGV[0]
require '/var/www/openshift/broker/config/environment'
app, gear = Application::find_by_gear_uuid(gear_uuid)
puts app.user.login
# chmod +x /root/find_user.rb

```

2. Run the script to find which user is associated with the application.

```

# ./find_user.rb e775030f960b4148923e158a411f273a
scollier

```



6 Conclusion

OpenShift Enterprise provides the flexibility that developers require to quickly deploy applications in a PaaS environment. The goal of this reference architecture is to provide guidance on how to complete the following tasks.

- Components Overview
- Reference Architecture Configuration
- Infrastructure Deployment
- Operational Management

Each of these sections went into great detail to provide enough information for administrators and developers to take advantage of the functionality that OpenShift Enterprise provides. One of the most important aspects of this reference architecture was the configuration of a distributed environment. The OpenShift Enterprise configuration detailed within can be customized even further to meet the requirements of specific environments.



Appendix A: Revision History

Revision 1.0

December 2012

Scott Collier

Initial Release



Appendix B: Contributors

Contributor	Title	Contribution
Matt Hicks	Director of Software Engineering	Review
Joe Fernandes	Senior Product Marketing Manager	Review
John Keck	Senior Manager – Software Engineering	Review
Luke Meyer	Senior Software Engineer	Review
Chrisopher Alfonso	Principal Software Engineer	Deployment Assistance
Brenton Leanhardt	Principal Software Engineer	Deployment Assistance
Hiram Chirino	Consulting Software Engineer	ActiveMQ Configuration
Eric Schabell	Senior Product Marketing Manager	JBDS Assistance



Appendix C: ActiveMQ Configuration Files

bsn1:

```
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<!-- START SNIPPET: example -->
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:amq="http://activemq.apache.org/schema/core"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://activemq.apache.org/schema/core
http://activemq.apache.org/schema/core/activemq-core.xsd">

  <!-- Allows us to use system properties as variables in this
configuration file -->
  <bean
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigure
r">
    <property name="locations">
      <value>file:${activemq.conf}/credentials.properties</value>
    </property>
  </bean>

  <broker xmlns="http://activemq.apache.org/schema/core"
brokerName="bsn2.osop.cloud.lab.eng.bos.redhat.com" useJmx="true"
dataDirectory="${activemq.base}/data">

    <destinationPolicy>
      <policyMap>
        <policyEntries>
          <!-- <policyEntry topic="" producerFlowControl="true"
memoryLimit="1mb"> -->
          <policyEntry topic="" producerFlowControl="false"/>
          <policyEntry queue="*.reply.>" gcInactiveDestinations="true"
inactiveTimeoutBeforeGC="300000" />
        </policyEntries>
```



```
</policyMap>
</destinationPolicy>

<managementContext>
  <managementContext createConnector="false"/>
</managementContext>

<networkConnectors>
  <networkConnector name="bsn1-bsn2" duplex="true" uri="static:
(tcp://10.16.138.27:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>

  <networkConnector name="bsn1-bsn3" duplex="true" uri="static:
(tcp://10.16.138.31:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>
</networkConnectors>

<plugins>
  <statisticsBrokerPlugin/>
  <simpleAuthenticationPlugin>
    <users>
      <authenticationUser username="mcollective"
password="marionette" groups="mcollective,everyone"/>
      <authenticationUser username="amq" password="password"
groups="admins,everyone"/>
      <authenticationUser username="admin" password="secret"
groups="mcollective,admin,everyone"/>
    </users>
  </simpleAuthenticationPlugin>
  <authorizationPlugin>
    <map>
      <authorizationMap>
        <authorizationEntries>
          <authorizationEntry queue=">" write="admins" read="admins"
admin="admins" />
          <authorizationEntry topic=">" write="admins" read="admins"
admin="admins" />
          <authorizationEntry topic="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
          <authorizationEntry queue="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
          <authorizationEntry topic="ActiveMQ.Advisory.>"
read="everyone,all" write="everyone,all" admin="everyone,all"/>
        </authorizationEntries>
      </authorizationMap>
    </map>
  </authorizationPlugin>
</plugins>
```



```

        </authorizationPlugin>
    </plugins>

    <systemUsage>
        <systemUsage>
            <memoryUsage>
                <memoryUsage limit="64 mb"/>
            </memoryUsage>
            <storeUsage>
                <storeUsage limit="100 gb"/>
            </storeUsage>
            <tempUsage>
                <tempUsage limit="50 gb"/>
            </tempUsage>
        </systemUsage>
    </systemUsage>

    <transportConnectors>
        <transportConnector name="openwire" uri="tcp://0.0.0.0:61616"/>
        <transportConnector name="stomp" uri="stomp://0.0.0.0:61613"/>
    </transportConnectors>

</broker>

    <import resource="jetty.xml"/>

</beans>
<!-- END SNIPPET: example -->

```

bsn2:

```

<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<!-- START SNIPPET: example -->
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:amq="http://activemq.apache.org/schema/core"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://activemq.apache.org/schema/core

```



```
http://activemq.apache.org/schema/core/activemq-core.xsd">

    <!-- Allows us to use system properties as variables in this
configuration file -->
    <bean
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigure
r">
        <property name="locations">
            <value>file:${activemq.conf}/credentials.properties</value>
        </property>
    </bean>

<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="bsn2.osop.cloud.lab.eng.bos.redhat.com" useJmx="true"
dataDirectory="${activemq.base}/data">

    <destinationPolicy>
        <policyMap>
            <policyEntries>
                <!-- <policyEntry topic="" producerFlowControl="true"
memoryLimit="1mb"> -->
                <policyEntry topic="" producerFlowControl="false"/>
                <policyEntry queue="*.reply.>" gcInactiveDestinations="true"
inactiveTimeoutBeforeGC="300000" />
            </policyEntries>
        </policyMap>
    </destinationPolicy>

    <managementContext>
        <managementContext createConnector="false"/>
    </managementContext>

    <networkConnectors>
        <networkConnector name="bsn2-bsn1" duplex="true" uri="static:
(tcp://10.16.138.28:61616)" userName="amq" password="password">
            <staticallyIncludedDestinations>
                <topic physicalName="mcollective.discovery.reply"/>
                <topic physicalName="mcollective.openshift.reply"/>
            </staticallyIncludedDestinations>
        </networkConnector>

        <networkConnector name="bsn2-bsn3" duplex="true" uri="static:
(tcp://10.16.138.31:61616)" userName="amq" password="password">
            <staticallyIncludedDestinations>
                <topic physicalName="mcollective.discovery.reply"/>
                <topic physicalName="mcollective.openshift.reply"/>
            </staticallyIncludedDestinations>
        </networkConnector>
    </networkConnectors>

    <plugins>
        <statisticsBrokerPlugin/>
        <simpleAuthenticationPlugin>
            <users>
```



```
        <authenticationUser username="mcollective"
password="marionette" groups="mcollective,everyone"/>
        <authenticationUser username="amq" password="password"
groups="admins,everyone"/>
        <authenticationUser username="admin" password="secret"
groups="mcollective,admin,everyone"/>
    </users>
</simpleAuthenticationPlugin>
<authorizationPlugin>
    <map>
        <authorizationMap>
            <authorizationEntries>
                <authorizationEntry queue=">" write="admins" read="admins"
admin="admins" />
                <authorizationEntry topic=">" write="admins" read="admins"
admin="admins" />
                <authorizationEntry topic="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
                <authorizationEntry queue="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
                <authorizationEntry topic="ActiveMQ.Advisory.>"
read="everyone,all" write="everyone,all" admin="everyone,all"/>
            </authorizationEntries>
        </authorizationMap>
    </map>
</authorizationPlugin>
</plugins>

    <systemUsage>
        <systemUsage>
            <memoryUsage>
                <memoryUsage limit="64 mb"/>
            </memoryUsage>
            <storeUsage>
                <storeUsage limit="100 gb"/>
            </storeUsage>
            <tempUsage>
                <tempUsage limit="50 gb"/>
            </tempUsage>
        </systemUsage>
    </systemUsage>

    <transportConnectors>
        <transportConnector name="openwire" uri="tcp://0.0.0.0:61616"/>
        <transportConnector name="stomp" uri="stomp://0.0.0.0:61613"/>
    </transportConnectors>

</broker>

    <import resource="jetty.xml"/>

</beans>
<!-- END SNIPPET: example -->
```



bsn3:

```
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<!-- START SNIPPET: example -->
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:amq="http://activemq.apache.org/schema/core"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://activemq.apache.org/schema/core
http://activemq.apache.org/schema/core/activemq-core.xsd">

  <!-- Allows us to use system properties as variables in this
configuration file -->
  <bean
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigure
r">
    <property name="locations">
      <value>file:${activemq.conf}/credentials.properties</value>
    </property>
  </bean>

<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="bsn2.osop.cloud.lab.eng.bos.redhat.com" useJmx="true"
dataDirectory="${activemq.base}/data">

  <destinationPolicy>
    <policyMap>
      <policyEntries>
        <!-- <policyEntry topic="" producerFlowControl="true"
memoryLimit="1mb"> -->
        <policyEntry topic="" producerFlowControl="false"/>
        <policyEntry queue="*.reply.>" gcInactiveDestinations="true"
inactiveTimeoutBeforeGC="300000" />
      </policyEntries>
    </policyMap>
  </destinationPolicy>

  <managementContext>
```




```
<managementContext createConnector="false"/>
</managementContext>

<networkConnectors>
  <networkConnector name="bsn3-bsn1" duplex="true" uri="static:
(tcp://10.16.138.28:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>

  <networkConnector name="bsn3-bsn2" duplex="true" uri="static:
(tcp://10.16.138.27:61616)" userName="amq" password="password">
    <staticallyIncludedDestinations>
      <topic physicalName="mcollective.discovery.reply"/>
      <topic physicalName="mcollective.openshift.reply"/>
    </staticallyIncludedDestinations>
  </networkConnector>
</networkConnectors>

<plugins>
  <statisticsBrokerPlugin/>
  <simpleAuthenticationPlugin>
    <users>
      <authenticationUser username="mcollective"
password="marionette" groups="mcollective,everyone"/>
      <authenticationUser username="amq" password="password"
groups="admins,everyone"/>
      <authenticationUser username="admin" password="secret"
groups="mcollective,admin,everyone"/>
    </users>
  </simpleAuthenticationPlugin>
  <authorizationPlugin>
    <map>
      <authorizationMap>
        <authorizationEntries>
          <authorizationEntry queue=">" write="admins" read="admins"
admin="admins" />
          <authorizationEntry topic=">" write="admins" read="admins"
admin="admins" />
          <authorizationEntry topic="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
          <authorizationEntry queue="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
          <authorizationEntry topic="ActiveMQ.Advisory.>"
read="everyone,all" write="everyone,all" admin="everyone,all"/>
        </authorizationEntries>
      </authorizationMap>
    </map>
  </authorizationPlugin>
</plugins>

<systemUsage>
```



```
<systemUsage>
  <memoryUsage>
    <memoryUsage limit="64 mb"/>
  </memoryUsage>
  <storeUsage>
    <storeUsage limit="100 gb"/>
  </storeUsage>
  <tempUsage>
    <tempUsage limit="50 gb"/>
  </tempUsage>
</systemUsage>
</systemUsage>

<transportConnectors>
  <transportConnector name="openwire" uri="tcp://0.0.0.0:61616"/>
  <transportConnector name="stomp" uri="stomp://0.0.0.0:61613"/>
</transportConnectors>

</broker>

<import resource="jetty.xml"/>
</beans>
<!-- END SNIPPET: example -->
```



Appendix D: Configuration Files

The following files are provided in the tarball with the reference architecture on the Red Hat customer portal. Each directory represents a host.

```
# ls -laRth final_config_files_jan_2013/ | grep -v total | grep -v 4.0K
final_config_files_jan_2013/:

final_config_files_jan_2013/bsn2:
-rw-r--r--. 1 root root 4.8K Jan 4 2013 activemq.xml
-rw-r--r--. 1 root root 217 Jan 4 2013 dhclient-eth0.conf
-rw-r--r--. 1 root root 1.1K Jan 4 2013 jetty-realm.properties
-rw-r--r--. 1 root root 5.6K Jan 4 2013 jetty.xml
-rw-----. 1 root root 982 Jan 4 2013 iptables
-rw-r--r--. 1 root root 2.0K Jan 4 2013 mongodb.conf

final_config_files_jan_2013/bsn3:
-rw-r--r--. 1 root root 4.9K Jan 4 2013 activemq.xml
-rw-r--r--. 1 root root 217 Jan 4 2013 dhclient-eth0.conf
-rw-r--r--. 1 root root 1.1K Jan 4 2013 jetty-realm.properties
-rw-r--r--. 1 root root 5.6K Jan 4 2013 jetty.xml
-rw-----. 1 root root 980 Jan 4 2013 iptables
-rw-r--r--. 1 root root 2.0K Jan 4 2013 mongodb.conf

final_config_files_jan_2013/broker2:
-rw-r-----. 1 root root 4.3K Jan 4 2013 production.rb
-rw-r--r--. 1 root root 1.1K Jan 4 2013 openshift-origin-auth-remote-
user.conf-console
-rw-r--r--. 1 root root 1.8K Jan 4 2013 openshift-origin-auth-remote-
user.conf-broker
-rw-r--r--. 1 root root 1.1K Jan 4 2013 openshift-origin-auth-remote-
user.conf
-rw-----. 1 root root 673 Jan 4 2013 iptables
-rw-r--r--. 1 root root 110 Jan 4 2013 ifcfg-eth0:0
-rw-r--r--. 1 root root 87 Jan 4 2013 dhclient-eth0.conf
-rw-r--r--. 1 root root 1.3K Jan 4 2013 client.cfg
-rw-r-----. 1 root root 1.3K Jan 4 2013 broker.conf
-rw-----. 1 root root 285 Jan 4 2013 arptables

final_config_files_jan_2013/broker1:
-rw-r-----. 1 root root 4.3K Jan 4 2013 production.rb
-rw-r--r--. 1 root root 1.1K Jan 4 2013 openshift-origin-auth-remote-
user.conf-console
-rw-r--r--. 1 root root 1.8K Jan 4 2013 openshift-origin-auth-remote-
user.conf-broker
-rw-----. 1 root root 673 Jan 4 2013 iptables
-rw-r--r--. 1 root root 110 Jan 4 2013 ifcfg-eth0:0
-rw-r--r--. 1 root root 220 Jan 4 2013 dhclient-eth0.conf
-rw-r--r--. 1 root root 1.3K Jan 4 2013 client.cfg
-rw-r-----. 1 root root 1.3K Jan 4 2013 broker.conf
-rw-----. 1 root root 286 Jan 4 2013 arptables

final_config_files_jan_2013/lvs1:
```



```
-rw-r--r--. 1 root root 823 Jan 4 2013 lvs.cf
-rwxr-xr-x. 1 root root 154 Jan 4 2013 check_web_https.sh
-rw-----. 1 root root 675 Jan 4 2013 iptables

final_config_files_jan_2013/lvs2:
-rw-r--r--. 1 root root 823 Jan 4 2013 lvs.cf
-rwxr-xr-x. 1 root root 154 Jan 4 2013 check_web_https.sh
-rw-----. 1 root root 675 Jan 4 2013 iptables

final_config_files_jan_2013/bsn1:
-rw-r--r--. 1 root root 4.9K Jan 4 2013 activemq.xml
-rw-r--r--. 1 root root 217 Jan 4 2013 dhclient-eth0.conf
-rw-r--r--. 1 root root 1.1K Jan 4 2013 jetty-realm.properties
-rw-r--r--. 1 root root 5.6K Jan 4 2013 jetty.xml
-rw-----. 1 root root 1.1K Jan 4 2013 iptables
-rw-r--r--. 1 root root 2.0K Jan 4 2013 mongodb.conf

final_config_files_jan_2013/client:
-rw-r--r--. 1 root root 120 Jan 4 2013 express.conf
-rw-r--r--. 1 root root 64 Jan 4 2013 eclipse.ini

final_config_files_jan_2013/node3:
-rw-----. 1 root root 747 Jan 4 2013 iptables
-rw-r--r--. 1 root root 2.5K Jan 4 2013 node.conf
-rw-r--r--. 1 root root 218 Jan 4 2013 dhclient-eth0.conf
-rw-r-----. 1 root root 1.3K Jan 4 2013 server.cfg

final_config_files_jan_2013/node2:
-rw-----. 1 root root 747 Jan 4 2013 iptables
-rw-r--r--. 1 root root 2.5K Jan 4 2013 node.conf
-rw-r--r--. 1 root root 218 Jan 4 2013 dhclient-eth0.conf
-rw-r-----. 1 root root 1.3K Jan 4 2013 server.cfg

final_config_files_jan_2013/node1:
-rw-----. 1 root root 747 Jan 4 2013 iptables
-rw-r--r--. 1 root root 2.5K Jan 4 2013 node.conf
-rw-r--r--. 1 root root 218 Jan 4 2013 dhclient-eth0.conf
-rw-r-----. 1 root root 1.3K Jan 4 2013 server.cfg
```



Appendix E: Configure Microsoft Windows 2008 R2 Active Directory Server

Deploy the Microsoft Windows Server 2008 R2 Standard Active Directory Domain controller. Using the Red Hat Enterprise Virtualization Manager, deploy the Windows server.

On the Red Hat Enterprise Virtualization Manager:

1. Follow the instructions in the **Create a Windows Virtual Machine** section of the Red Hat Enterprise Virtualization 3.1 guide¹⁴.
2. Once the Windows virtual machine is installed, perform the following steps to configure Active Directory
3. Make sure the Windows server has been updated with the latest patches from Microsoft

1. Open a console to the Windows virtual machine and either continue using the console or enable remote desktop with the following instructions

1. On the first screen **Initial Configuration Tasks**, select **Enable Remote Desktop** in **Step 3 Customize this Server**
2. In the **System Properties** screen, select the radio button to **Allow Connections from computers running any version of Remote Desktop (less secure)**, or the more secure option.

3. Open a remote desktop connection with the following command

```
$ rdesktop -g 1024x768 -uadministrator -pPASSWORD 10.16.138.38 &
```

4. Configure Active Directory

1. Open Server Manager and add a role
 1. **Start -> Administrative Tools -> Server Manager**
 2. Click on **Roles** in the left hand navigation pane
 3. Click **Add Roles** in the right hand navigation pane
 4. Click **Next**
 5. Select **Active Directory Domain Services**
 6. Add the **.NET Framework 3.5.1 Features**
 1. Click **Add Required Features**
 7. Click **Next**
 8. Click **Next**
 9. Click **Install**
 10. Click **Close**



2. Promote the server to a domain controller for a new forest

1. Click **Start** and type in **dcpromo** in the dialog box

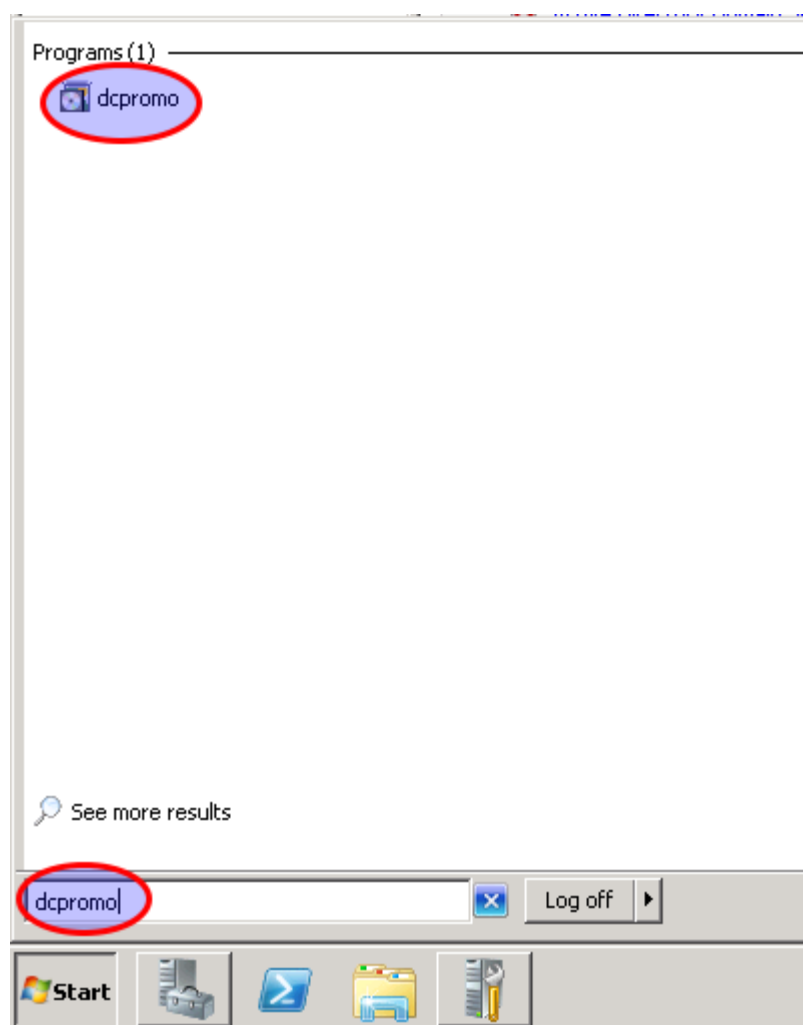


Figure 6.1: dcpromo



2. Click Next

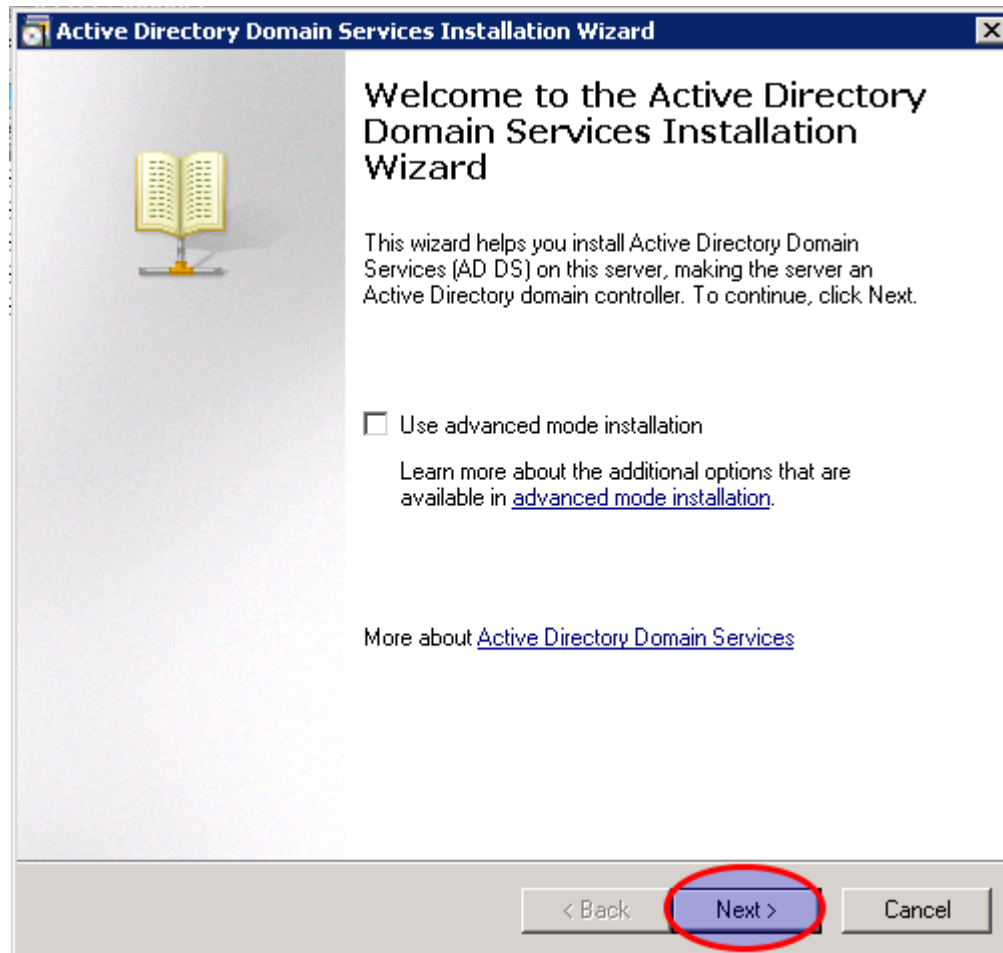


Figure 6.2: AD Wizard



3. Click Next

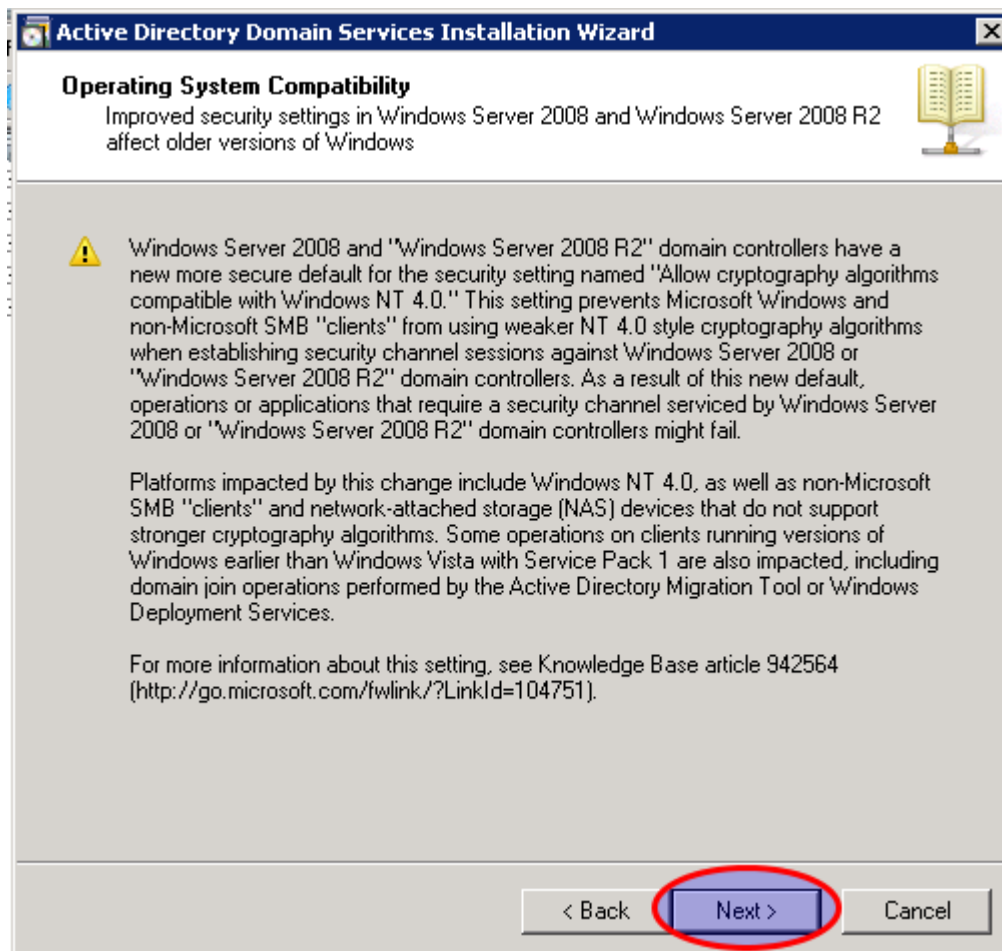


Figure 6.3: OS Compatibility



4. Select **Create a new domain in a new forest**, click **Next**

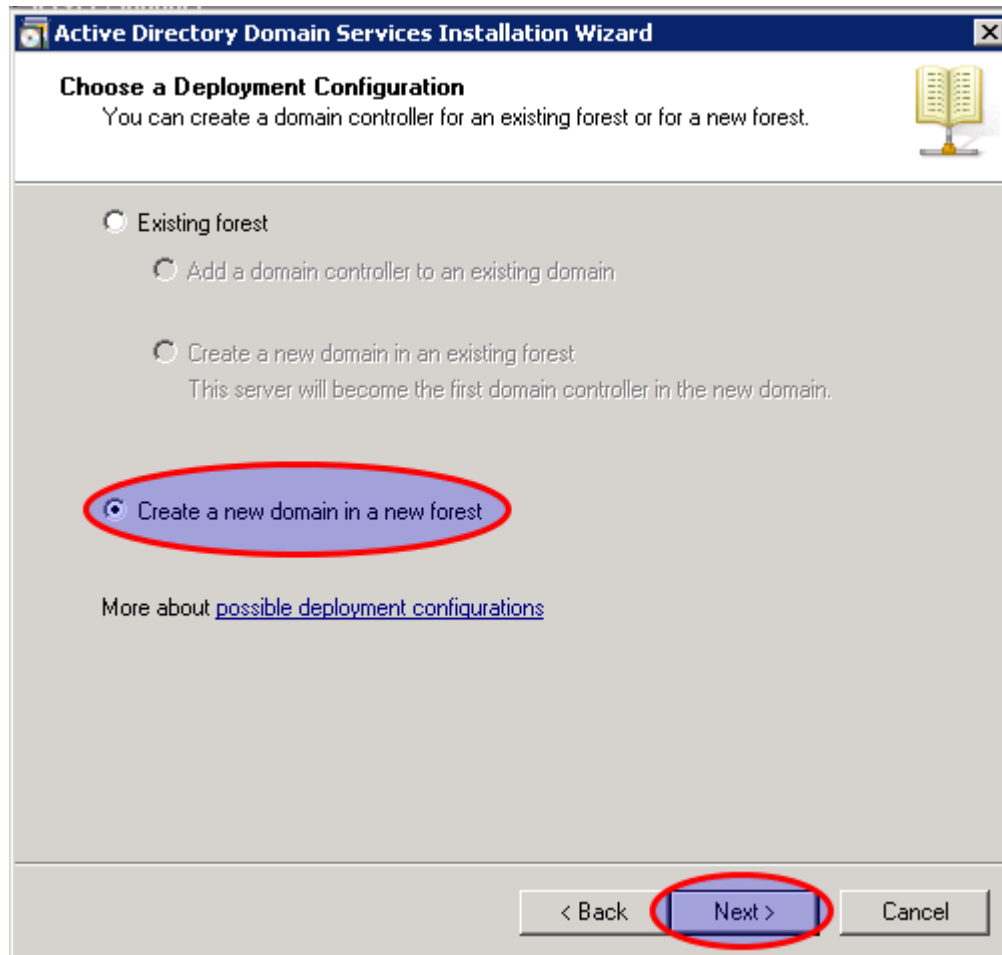


Figure 6.4: Deployment Configuration



5. Provide a FQDN of the forest root domain

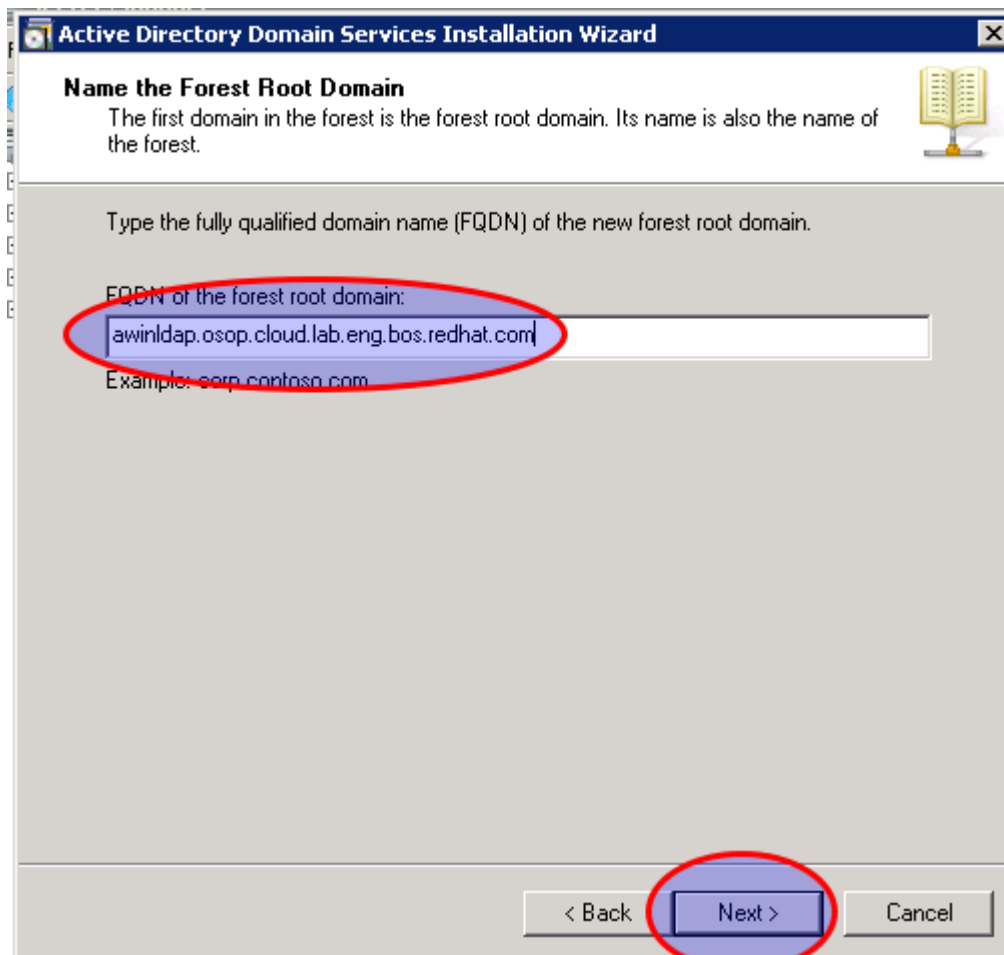


Figure 6.5: Root Domain



6. Set the Forest Functional Level

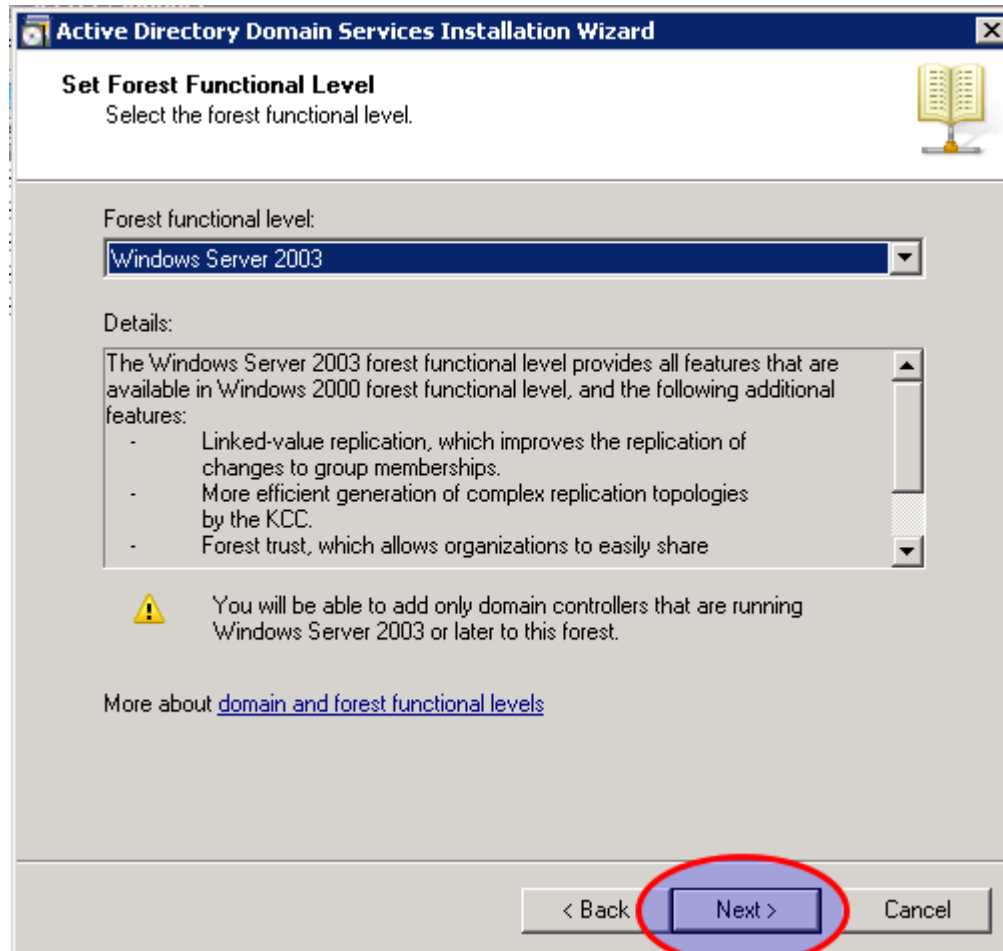


Figure 6.6: Functional Level



7. Include DNS

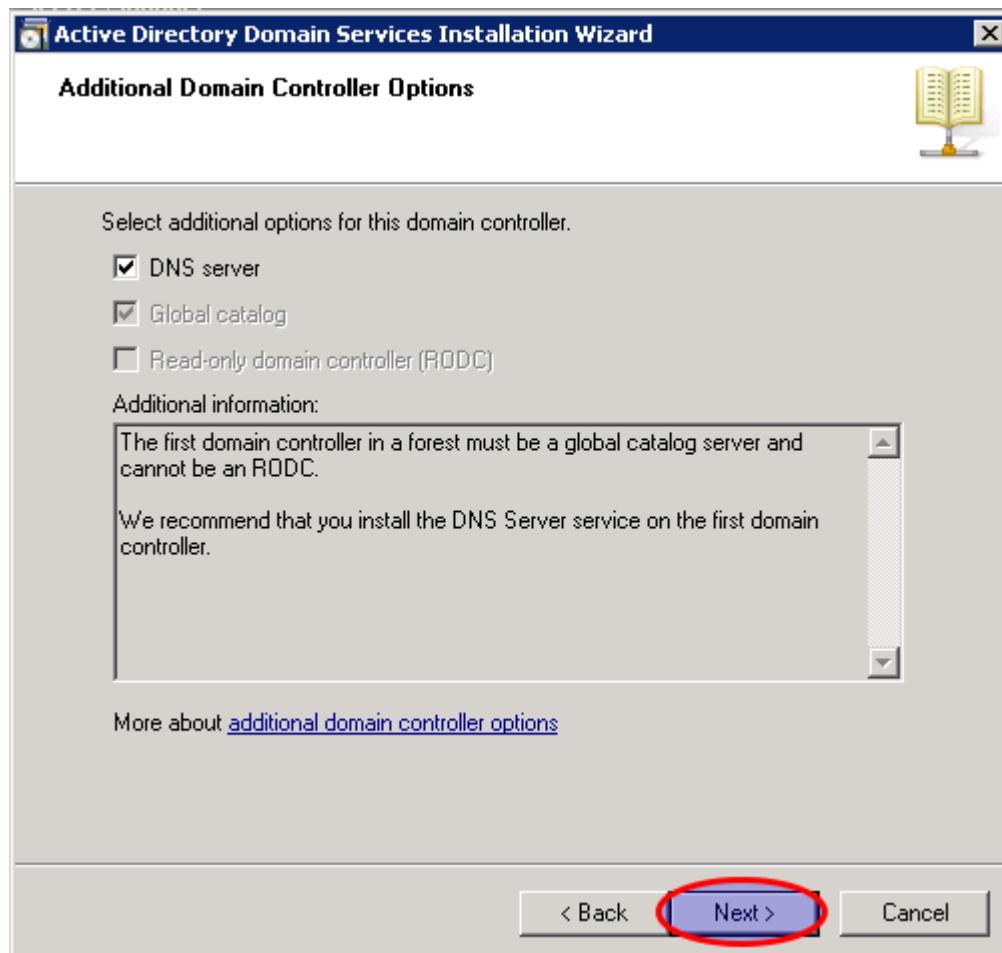


Figure 6.7: DC Options

8. Allow DNS server delegation

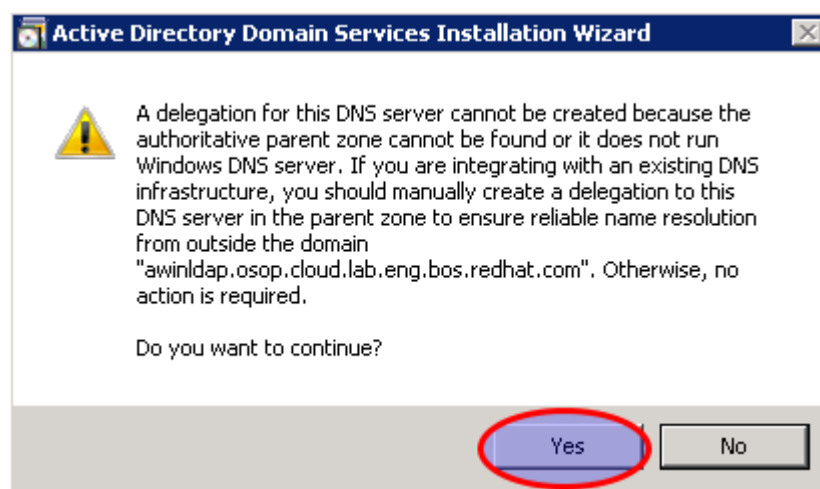


Figure 6.8: DNS Delegation



9. Click **Next** to continue the installation

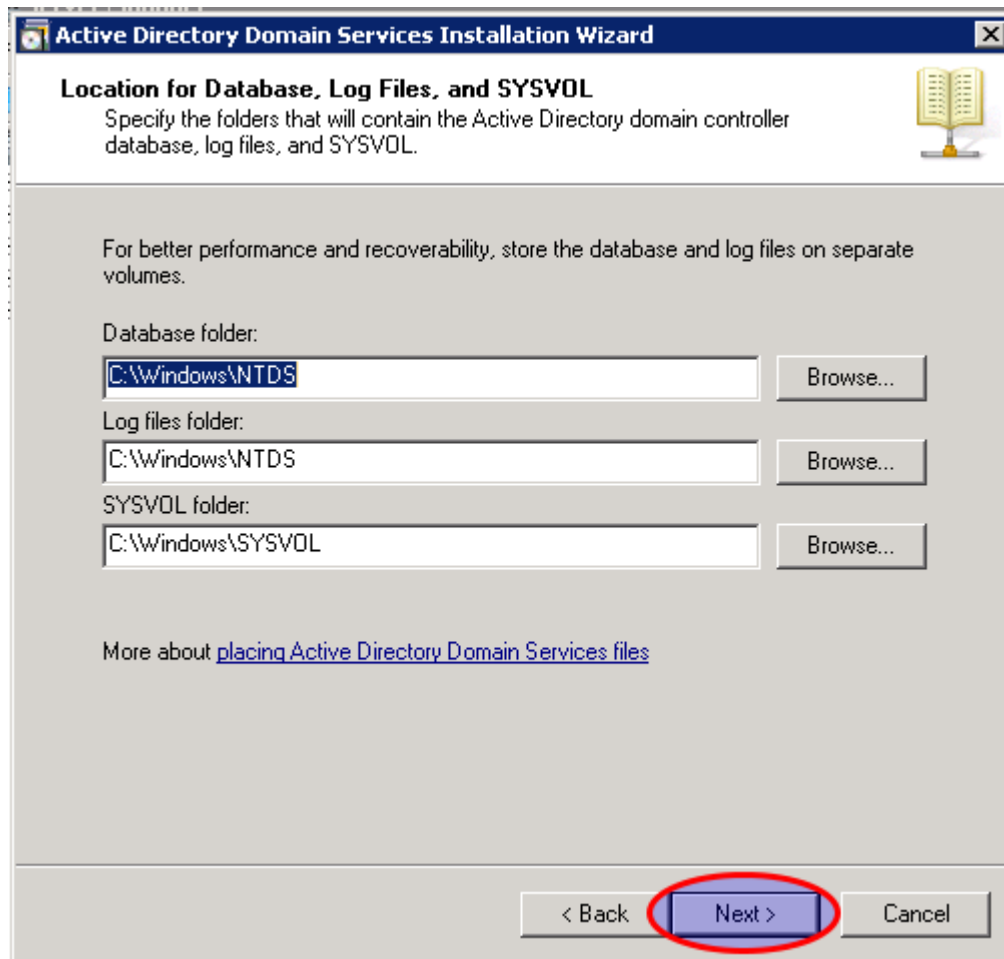


Figure 6.9: Database Location



10. Provide a Restore mode password

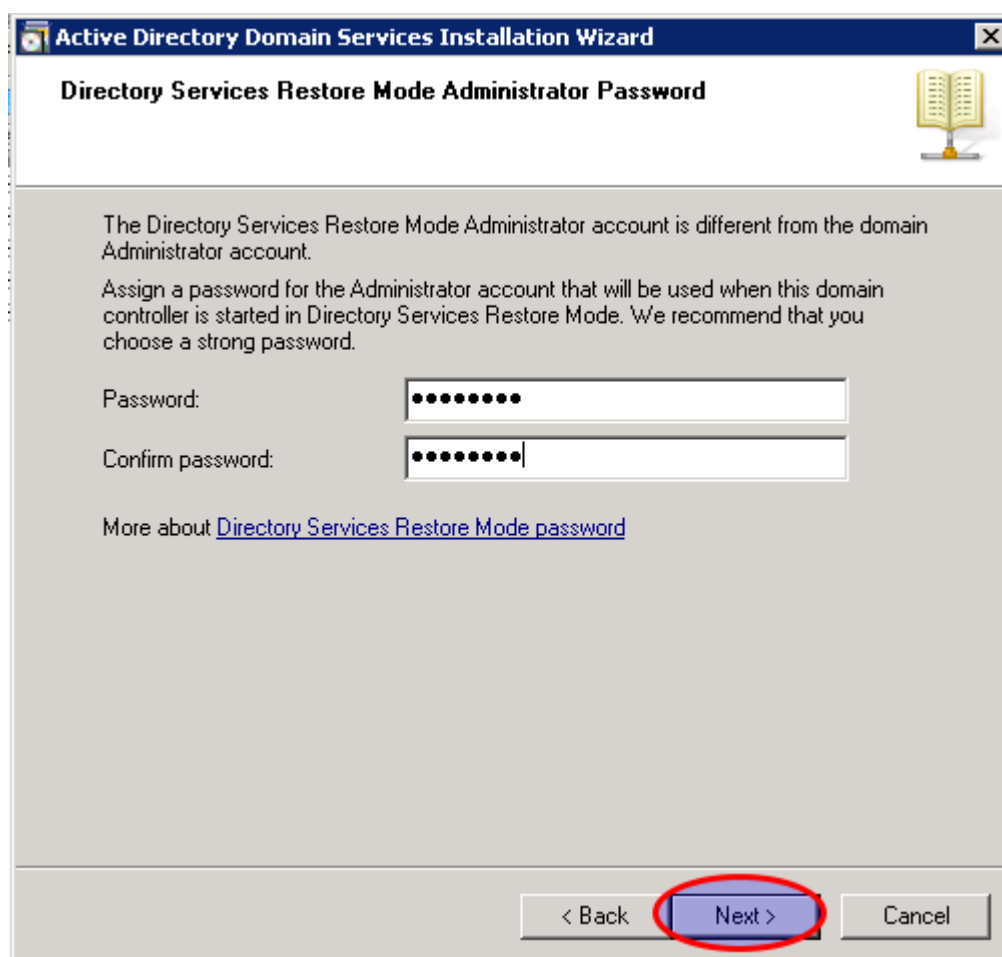


Figure 6.10: Restore Password



11. Click **Next** at the summary

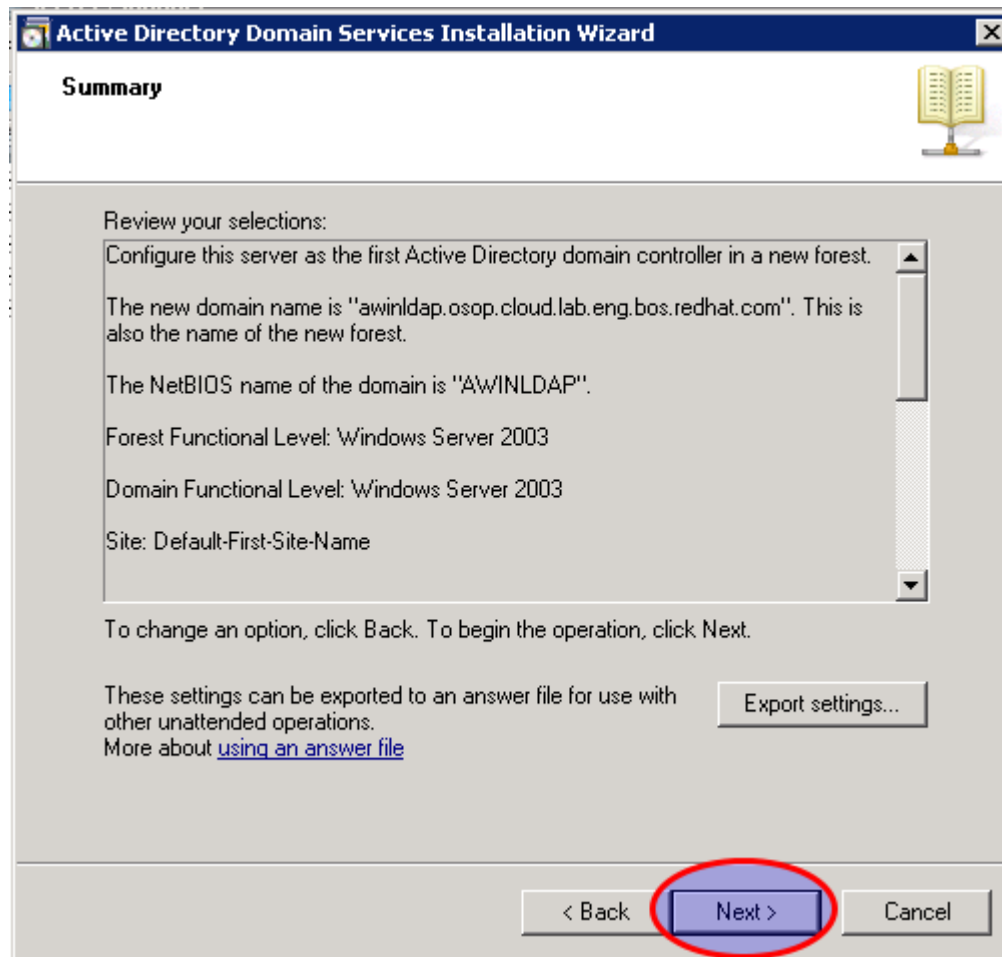


Figure 6.11: DC Install Wizard



Figure 6.12: Install Progress



12. Click **Finish**

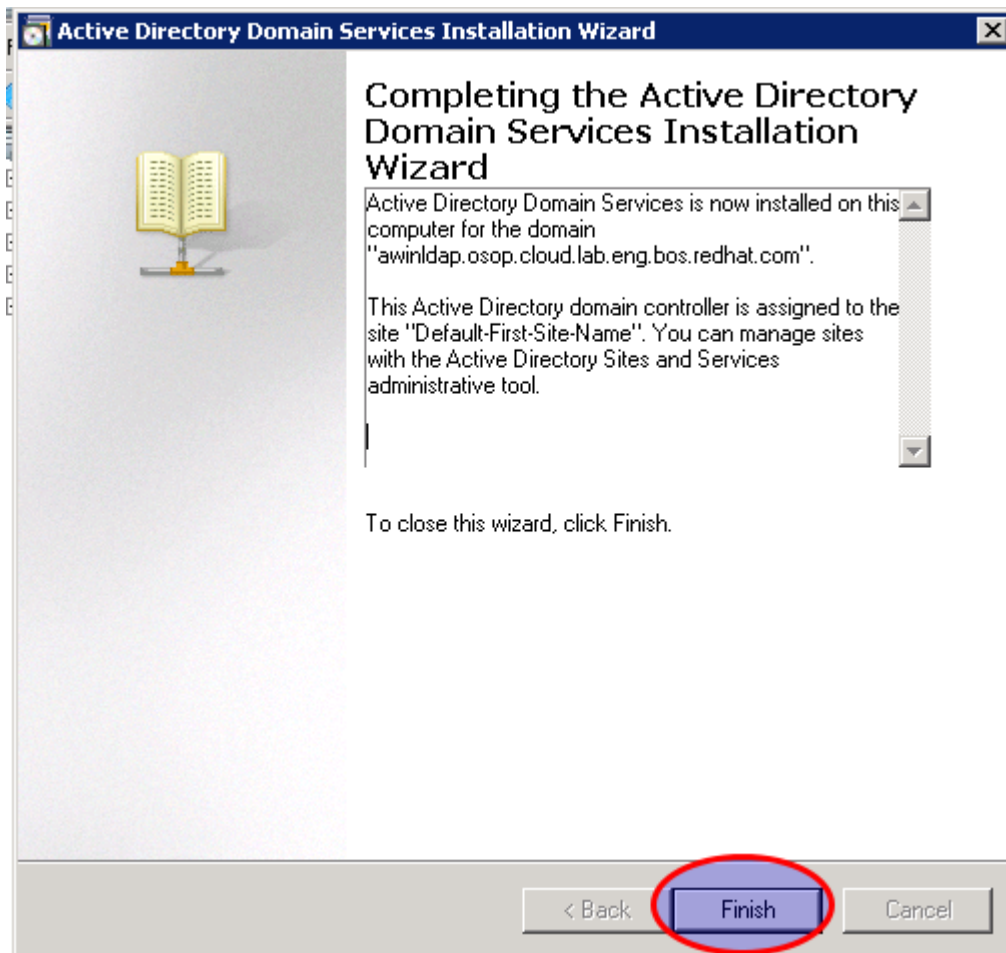


Figure 6.13: Finish DC Installation

13. Click **Restart Now**

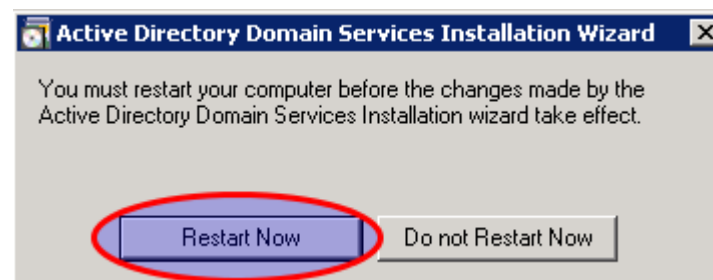


Figure 6.14: DC Reboot



5. Add a few users to Active Directory

1. Click **Start -> Administrative Tools -> Active Directory Users and Computers**

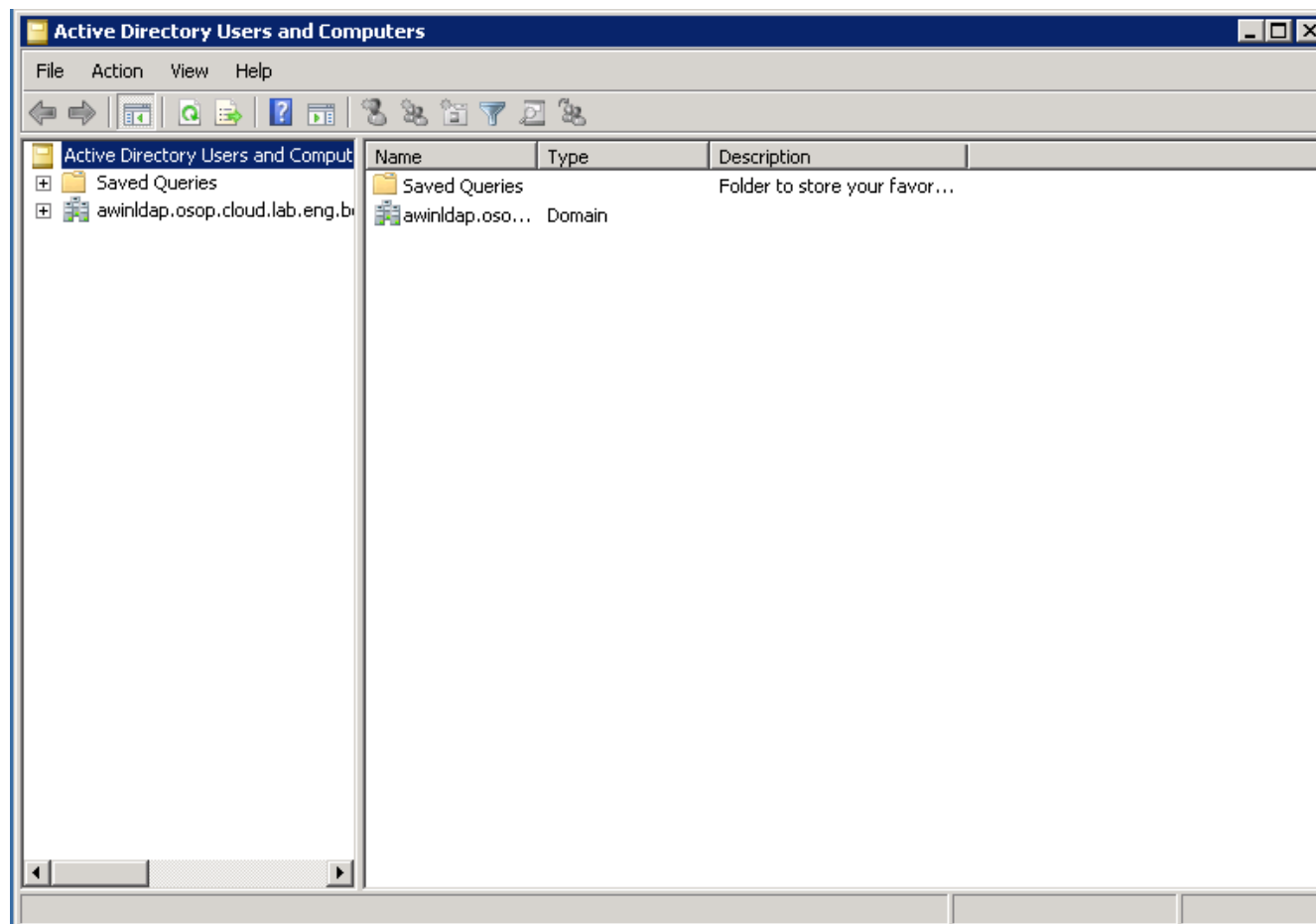


Figure 6.15: AD Users and Computers



2. On the left hand navigation pane, expand the organizational unit (OU) and create a new one.

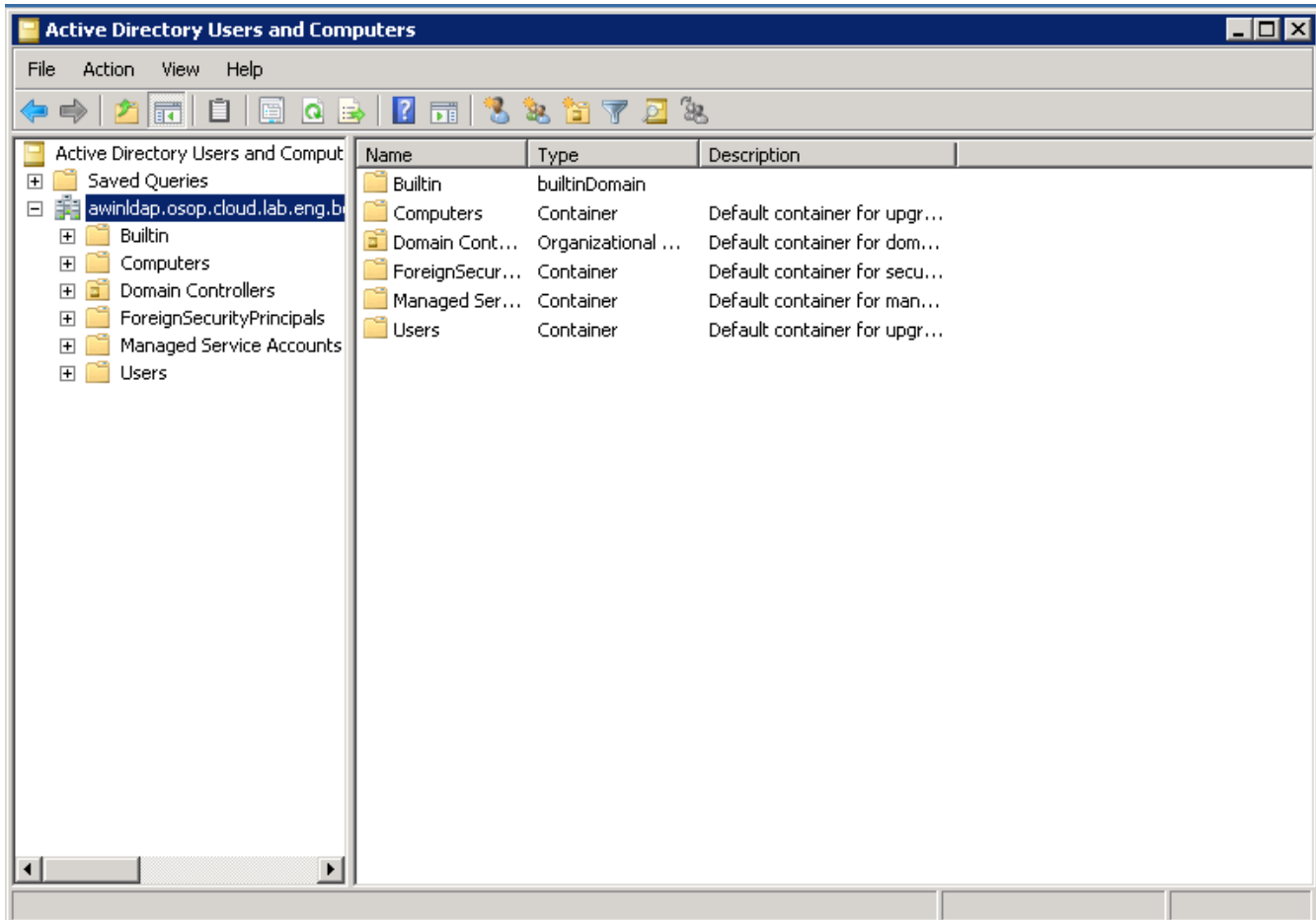


Figure 6.16: AD Users and Computers Groups



3. Provide a **Name** and click OK

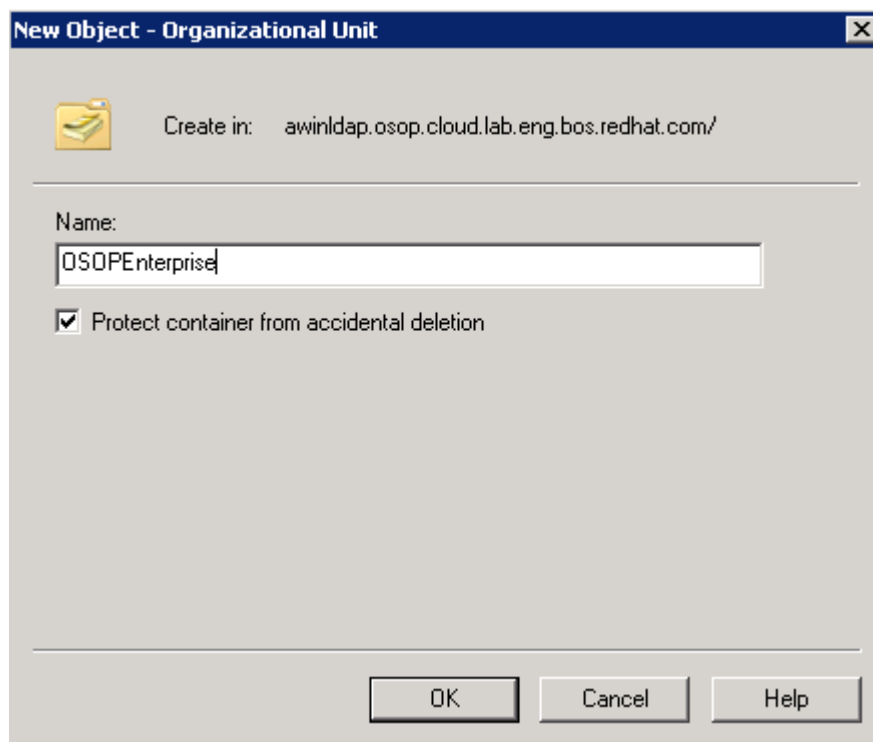


Figure 6.17: New OU



4. Right click on the new OU and create a new user, provide **First name**, **Last name**, **User logon name** and click **Next**

New Object - User

Create in: sop.cloud.lab.eng.bos.redhat.com/DSOPEnterprise

First name: Initials:

Last name:

Full name:

User logon name:

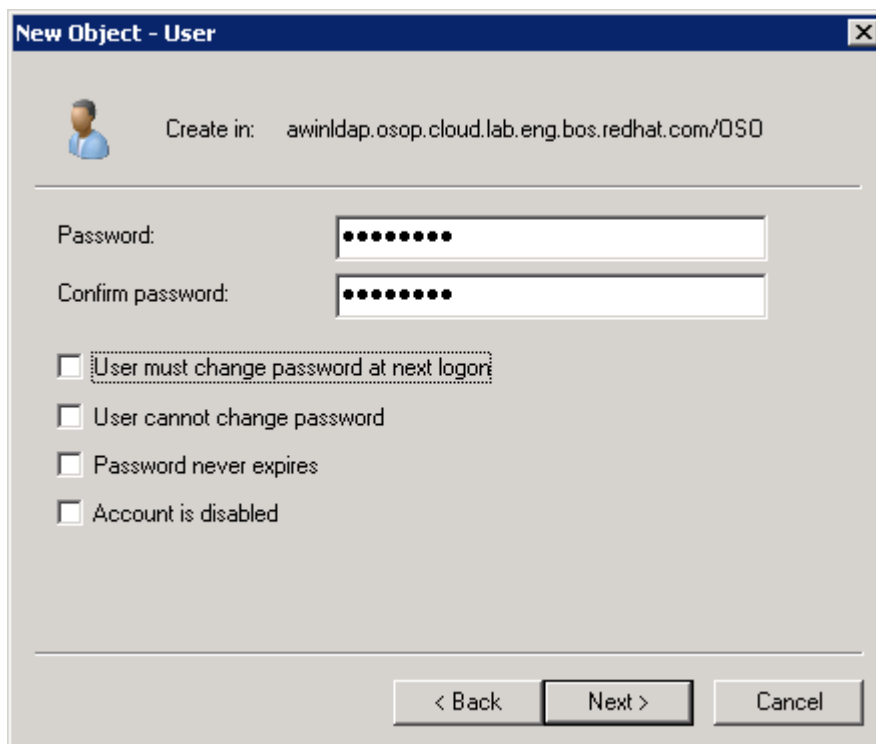
User logon name (pre-Windows 2000):

< Back Next > Cancel

Figure 6.18: OU Info



5. Provide a **Password** and uncheck **User must change password at next logon** and click **Next**



The image shows a 'New Object - User' dialog box. At the top, it says 'Create in: awindap.osop.cloud.lab.eng.bos.redhat.com/OSO'. Below this, there are two password fields: 'Password:' and 'Confirm password:', both containing masked characters. Underneath the password fields are four checkboxes, all of which are unchecked: 'User must change password at next logon', 'User cannot change password', 'Password never expires', and 'Account is disabled'. At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'.

Figure 6.19: User Password



6. Click **Finish**

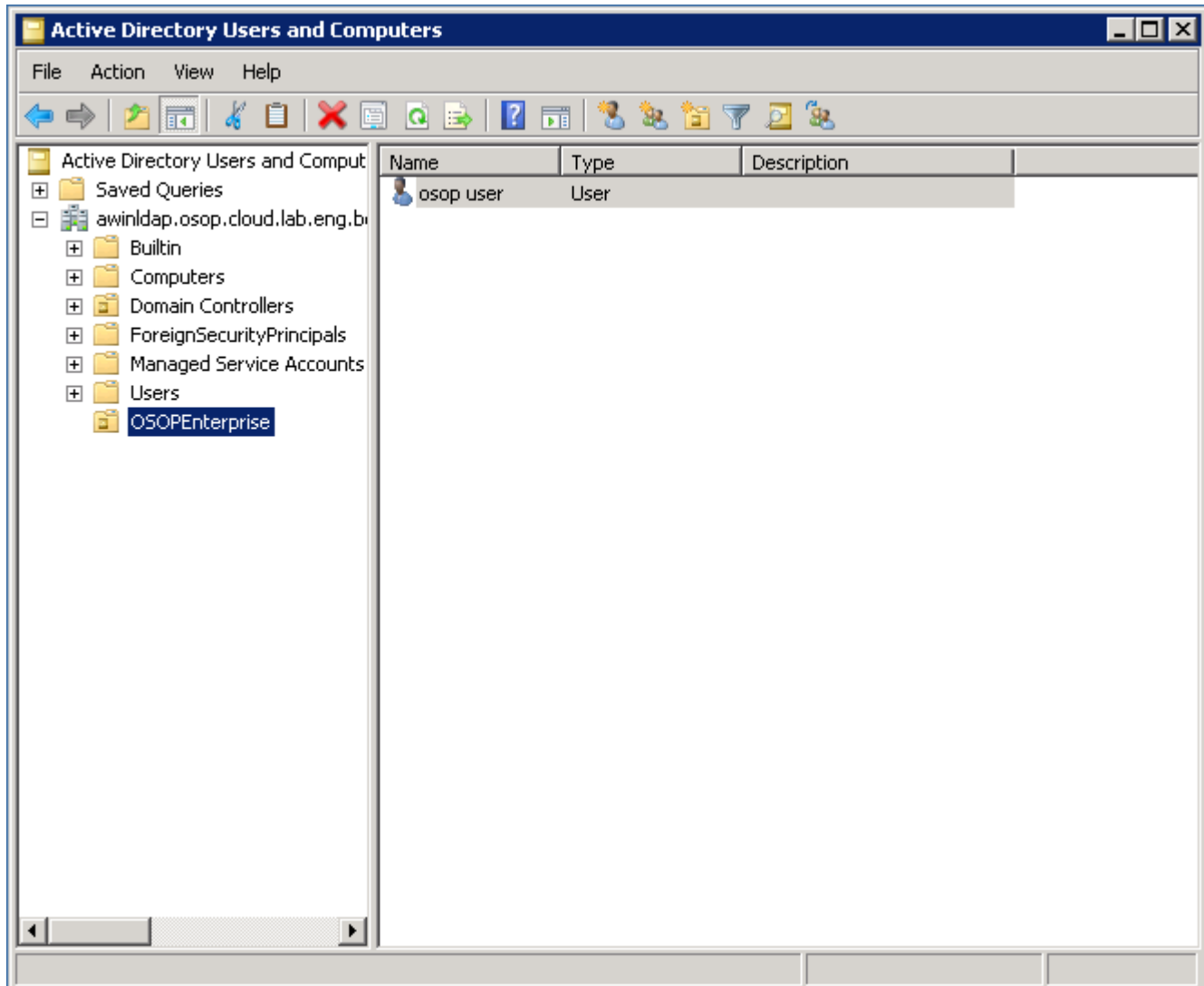


Figure 6.20: AD User

7. Repeat steps 4-6 and create a few more users



- 1 https://access.redhat.com/knowledge/docs/OpenShift_Enterprise/
 - 2 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Load_Balancer_Administration/index.html#s2-lvs-directrouting-VSA
 - 3 https://access.redhat.com/knowledge/sites/default/files/attachments/ose_final_config_files_jan_2013.tar
 - 4 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Installation_Guide/index.html
 - 5 <http://activemq.apache.org/networks-of-brokers.html>
 - 6 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Installation_Guide/index.html
 - 7 https://bugzilla.redhat.com/show_bug.cgi?id=888043
 - 8 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Installation_Guide/index.html
 - 9 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Load_Balancer_Administration/index.html#ch-initial-setup-VSA
 - 10 https://access.redhat.com/knowledge/docs/JBoss_Developer_Studio/
 - 11 https://access.redhat.com/knowledge/docs/JBoss_Developer_Studio/
 - 12 <https://github.com/fabianofranz/spring-eap6-quickstart>
 - 13 https://bugzilla.redhat.com/show_bug.cgi?id=885587
 - 14 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Virtualization/3.1-Beta/html-single/Quick_Start_Guide/index.html#Quick_Start_Guide-Manage_Virtual_Machine
-
-