



Red Hat Reference Architecture Series

Integrating OpenShift Enterprise with Identity Management (IdM) in Red Hat Enterprise Linux

OpenShift Enterprise 2.2

IdM in Red Hat Enterprise Linux 7

Windows Server 2012 - Active Directory Integration

Mark Heslin

Principal Systems Engineer

Version 1.1

January 2015





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, the Intel logo and Xeon 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.

© 2014 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



Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers.

Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook:

<https://www.facebook.com/rhrefarch>

Follow us on Twitter:

<https://twitter.com/RedHatRefArch>

Plus us on Google+:

<https://plus.google.com/u/0/b/114152126783830728030/>



Table of Contents

1 Executive Summary.....	1
2 Component Overview.....	2
2.1 OpenShift Enterprise.....	2
2.1.1 Broker.....	3
2.1.2 Node.....	4
2.1.3 Gears.....	5
2.1.4 Cartridges.....	6
2.1.5 MongoDB.....	6
2.1.6 ActiveMQ.....	7
2.1.6.1 Network of ActiveMQ Brokers.....	8
2.1.7 Summary.....	9
2.2 Identity Management (IdM) in Red Hat Enterprise Linux.....	10
2.2.1 Kerberos - Key Distribution Center (KDC).....	10
2.2.2 389 Directory Server (LDAP).....	11
2.2.3 Dogtag Certificate System.....	11
2.2.4 Domain Name System (DNS).....	11
2.2.5 Time Services (ntpd, chronyd).....	12
2.2.6 IdM Replication.....	12
2.2.7 Cross-realm Kerberos Trusts.....	13
2.3 Red Hat Enterprise Virtualization (RHEV).....	13
2.4 Red Hat Enterprise Linux (RHEL).....	14
2.5 Windows Server 2012 R2.....	15
2.5.1 Active Directory Domain Services (AD DS).....	16
2.6 System Security Services Daemon (SSSD).....	16
2.7 RHC Client Tools.....	17
2.8 IPA Admin Tools.....	17
3 Reference Architecture Configuration.....	18
3.1 Environment.....	18
3.2 Virtual Machines.....	20
3.2.1 Virtual Machine Sizing.....	20
3.2.2 Network Addresses.....	20
3.3 Software Configuration.....	21
3.3.1 Required Software Channels.....	21



3.3.2 Software Versions.....	22
3.3.3 Security.....	24
3.3.4 Network Ports.....	24
3.4 Hardware Configuration.....	26
3.4.1 Hypervisors.....	26
3.4.2 Storage.....	26
3.5 Component Log Files.....	27
4 Staging the Infrastructure.....	30
4.1 Red Hat Enterprise Virtualization (RHEV).....	30
4.2 Install Red Hat Enterprise Linux.....	31
4.3 Configure Network.....	31
4.4 Configure Domain Name System (DNS).....	32
4.5 Configure Time Service (NTP).....	33
4.6 Configure SELinux.....	34
4.7 Update Hosts File.....	35
4.8 Register Hosts and Run Updates.....	36
5 Deploying Identity Management in Red Hat Enterprise Linux.....	37
5.1 Deploy IdM Server.....	37
5.1.1 Configure Firewall Ports.....	37
5.1.2 Install Packages.....	39
5.1.3 Install/configure IdM Server.....	39
5.1.4 Verify IdM Server.....	41
5.1.5 Configure IdM Groups and Users.....	41
5.1.6 Verify IdM Groups and Users.....	43
5.2 Deploy IdM Admin Client.....	45
5.2.1 Configure Firewall Ports.....	45
5.2.2 Install Packages.....	47
5.2.3 Configure DNS.....	47
5.2.4 Install/Configure IdM Client.....	47
5.2.5 Verify IdM Admin Client.....	48
5.2.6 Install Admin Tools (optional).....	49
5.3 Deploy IdM Replica Server.....	50
5.3.1 Configure Firewall Ports.....	50
5.3.2 Install Packages.....	52
5.3.3 Run Replica Prepare.....	52



5.3.4 Copy Replica Information File.....	53
5.3.5 Run Replica Installation.....	53
5.3.6 Verify DNS.....	54
5.3.7 Verify Replication.....	56
5.3.8 Test Replication.....	56
6 Deploying OpenShift Enterprise.....	58
6.1 Configure Deployment Script.....	58
6.2 Deploy Brokers.....	61
6.2.1 Configure Entitlements.....	61
6.2.2 Configure Yum.....	62
6.2.3 Install OpenShift.....	64
6.2.3.1 Recovering from Installation Failures.....	66
6.2.4 Re-configure Firewall Ports (optional).....	66
6.3 Deploy Nodes.....	69
6.3.1 Configure Entitlements.....	69
6.3.2 Configure Yum.....	71
6.3.3 Install OpenShift.....	73
6.3.3.1 Recovering from Installation Failures.....	74
6.3.4 Re-configure Firewall Ports (optional).....	74
6.3.5 Verify Communications.....	76
6.4 Post Deployment Tasks.....	77
6.4.1 Run Post_Deploy.....	77
6.4.1.1 Recovering from Post Deployment Failures.....	78
6.4.2 Copy public rsync key to enable moving gears.....	78
6.4.3 Copy ssh host keys, httpd key/cert across Nodes.....	79
6.4.4 Copy Auth Keys across Brokers.....	80
6.4.5 Configure Broker DNS Request Balancing (optional).....	81
6.4.6 Update DNS Entries.....	82
6.5 Deploy RHC Client.....	84
6.5.1 Configure Entitlements.....	84
6.5.2 Install Package.....	85
6.6 Verify Openshift Enterprise Environment.....	87
6.6.1 Reboot Hosts.....	87
6.6.2 Run Diagnostics.....	87
7 Integrating OpenShift Enterprise with IdM.....	89
7.1 Overview.....	89



7.2 Integration Prerequisites.....	91
7.2.1 Configure Firewall Ports.....	91
7.2.2 Verify DNS.....	93
7.2.3 Configure IdM Clients.....	93
7.3 Integration Scenarios.....	96
7.3.1 IS1: Developer Authorization to Brokers (mandatory).....	96
7.3.2 IS2: Centralized Management of SSH Keys (optional).....	113
7.3.3 IS3: Centralized Management of UID's (optional).....	118
7.3.4 IS4: Kerberized Access to Gears (optional).....	123
7.3.5 IS5: Cross-realm Kerberos Trust (optional).....	126
8 Conclusion.....	144
Acknowledgements.....	145
Appendix A: References.....	146
Appendix B: Glossary.....	148
Appendix C: Active Directory - Deployment and Configuration.....	155
Appendix D: OpenShift Deployment Script.....	164
Appendix E: Deployment Checklist.....	168
Appendix F: Integration Checklist.....	170
Appendix G: Revision History.....	172



1 Executive Summary

OpenShift Enterprise (OSE) is built on the strength of a proven stack of Red Hat technologies. Based on the open source OpenShift Origin project, OpenShift Enterprise provides the freedom for developers and administrators to work the way they want. For developers, OpenShift Enterprise supports a variety of common programming languages and associated frameworks, database services and continuous integration. OpenShift Enterprise provides platform services through the use of "cartridges". Cartridges are extensible and allow customers to add their own custom services to OpenShift. Developers can access OpenShift Enterprise via a rich command line interface, web console, RESTful API or the Eclipse integrated development environment. For IT administrators, OpenShift Enterprise runs on the trusted Red Hat Enterprise Linux (RHEL) 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 Enterprise also provides standards-based application run-time environments such as JBoss Enterprise Application Platform (JEAP), Tomcat, and Apache. OpenShift Enterprise can be deployed on-premise in a data center, in a private cloud on a virtualization platform or through a cloud provider.

Identity Management (IdM) in Red Hat Enterprise Linux 7 provides a simplified, centralized and flexible solution to securely managing user identities, policies and authorizations in a native Linux-based domain. Based on the open source FreeIPA project, IdM integrates industry standard protocols and servers (Kerberos, LDAP, DNS, NTP, X509 Certificates) into a secure, reliable and scalable identity management solution. The IdM authentication and authorization framework is ideal for large scale Linux and Unix deployments in both traditional and cloud based enterprise environments.

This reference architecture demonstrates how to deploy and integrate OpenShift Enterprise 2.2 on Red Hat Enterprise Linux 6.6 with Identity Management in Red Hat Enterprise Linux 7. This configuration incorporates the latest features and capabilities of both products into a cohesive, tightly coupled solution, while also providing a solid foundation for incorporating future features and capabilities. From a sequencing perspective the IdM infrastructure is deployed here first; for customers with existing OpenShift Enterprise environments, most integration tasks can be implemented once the IdM infrastructure has been deployed.

The key to making any cross-product integration successful lies in understanding the points where the needs of one product can be aligned with the capabilities of another. During the systems engineering phase of this reference architecture, over a dozen potential integration scenarios were identified and subsequently narrowed down those presented here. Each of these scenarios has been validated, sequenced and streamlined for deployment purposes. Most of the integration scenarios are considered optional, but most environments will find these beneficial to implement as they extend the core functionality of the integration framework.

For improved availability of IdM services, an IdM replica server is configured. In turn, OpenShift Enterprise is deployed in a distributed configuration with the messaging (ActiveMQ) and database (MongoDB) services shared across multiple broker hosts. IdM Cross-realm Kerberos trusts are also presented for authentication of users within existing Active Directory domains.



2 Component Overview

2.1 OpenShift Enterprise

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.

Built on the strength of a proven stack of Red Hat technologies, OpenShift Enterprise provides the freedom for developers and administrators to work the way they want to work. For the developer, OpenShift Enterprise supports many common programming languages (Java, JavaScript/NodeJS, Ruby, PHP, Python, Perl) and associated frameworks, database services (MongoDB, MySQL, PostgreSQL) and continuous integration (Jenkins). OpenShift Enterprise provides platform services through the use of "cartridges". Cartridges are extensible and allow customers to add their own custom services to OpenShift. Developers can access OpenShift Enterprise via a rich command line interface, web console, RESTful API or the Eclipse integrated development environment (IDE). For the IT administrator, OpenShift Enterprise runs on the trusted Red Hat Enterprise Linux (RHEL) 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 Enterprise also provides standards-based application run-time environments such as JBoss Enterprise Application Platform (JEAP), Tomcat, and Apache. OpenShift Enterprise can be deployed on-premise in a data center, in a private cloud on a virtualization platform or through a cloud provider. In addition, developers can work from a variety of workstations platforms including Linux, OS X and Microsoft Windows.

OpenShift Enterprise is open source, based on the upstream OpenShift Origin project, that helps drive innovation and eliminate lock-in.



2.1.1 Broker

Brokers are the single point of contact for all application management activities. Brokers are responsible for the following tasks:

- Management of user authentication
- DNS updates
- Application orchestration
- Services and operations



Broker

Figure 2.1.1-1: Broker

Figure 2.1.1-1: Nodes represents an OpenShift broker host.

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

Figure 2.1.1-2: Broker Interfaces shows the methods available to developers for interacting with OpenShift broker hosts:

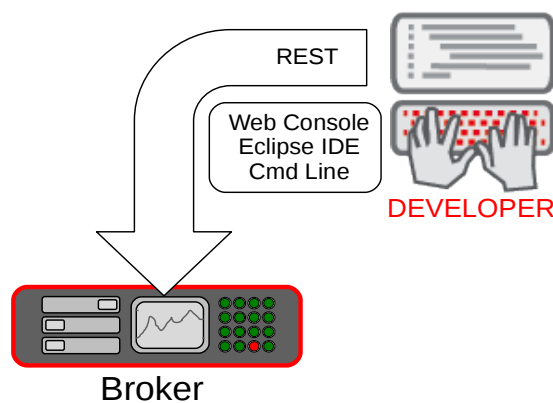


Figure 2.1.1-2: Broker Interfaces



2.1.2 Node

An OpenShift *node* is a host that runs the applications that a user has deployed.

Figure 2.1.2-1: Nodes shows how applications are deployed on OpenShift node hosts:

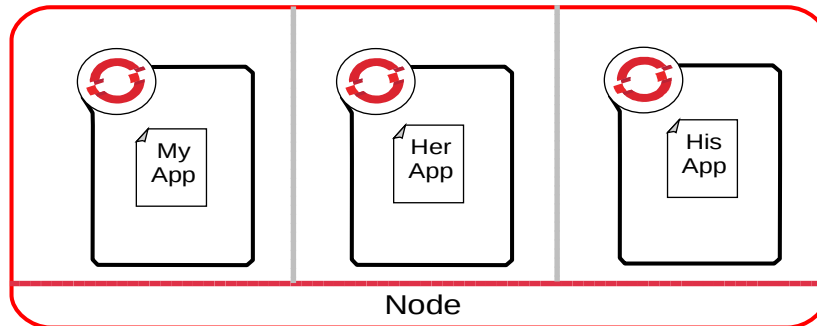


Figure 2.1.2-1: Nodes

Nodes are managed by one or more brokers hosts depending on how the infrastructure is deployed. Each node supports gears that contain the user deployed applications. Gears represent the system resources that an application is consuming.

One of the unique features of OpenShift is that it provides a true *multi-tenant* environment as depicted in **Figure 2.1.2-2: Nodes – Multi-tenancy**:

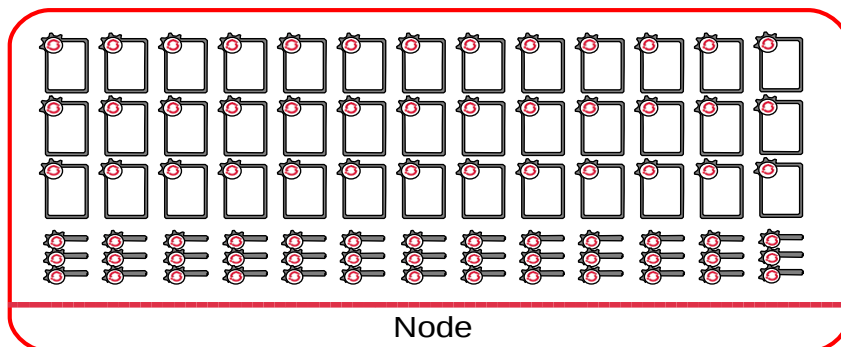


Figure 2.1.2-2: Nodes - Multi-tenancy

This is accomplished by using SELinux and Cgroup restrictions to separate the resources and data for each application, while also providing quality of service controls.

The consolidation of brokers and nodes onto one host is not supported by OpenShift Enterprise.



2.1.3 Gears

Gears are containers that run the applications that a user has deployed on a node. A gear is created for each application that is deployed within OpenShift.

Figure 2.1.3-1: Gears shows how applications deployed on OpenShift are containerized:

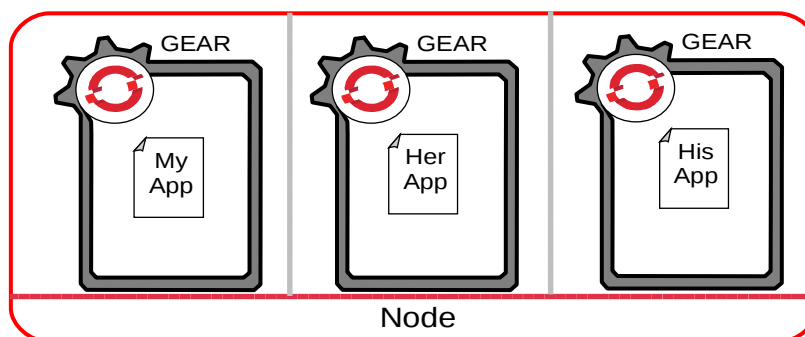


Figure 2.1.3-1: Gears

Management of the CPU, memory and disk resources required by the user application within each gear is handled through the use of *Cgroups* (control groups). *Cgroups* provide an efficient method of isolating applications and their resources from each other.

Figure 2.1.3-2: Gears, Cgroups and SELinux depicts how Gear security is implemented:

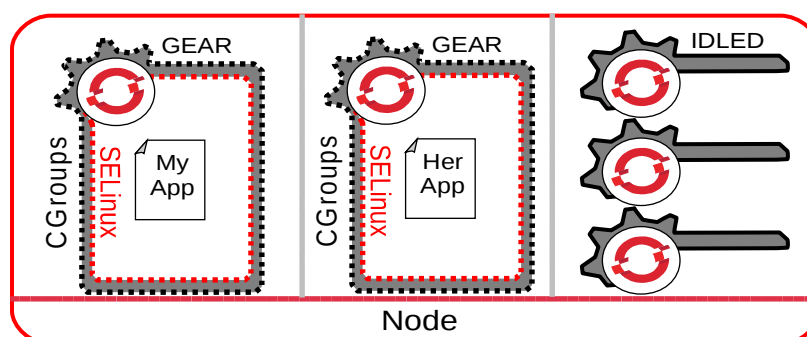


Figure 2.1.3-2: Gears, Cgroups and SELinux

Gear security is managed through the standard Red Hat Enterprise Linux SELinux security contexts and labels.



2.1.4 Cartridges

OpenShift automates the configuration of gears through the use of *cartridges*. Cartridges simplify the management and control of the underlying software that user applications within a gear consume. OpenShift Enterprise provides pre-defined cartridges for languages (PHP, Ruby, Python, Perl), middleware (JBoss EAP, JBoss EWS) and databases (PostgreSQL, MySQL, MongoDB). Custom cartridges can also be created.

Figure 2.1.4: Cartridges represents the modular structure of OpenShift cartridges:

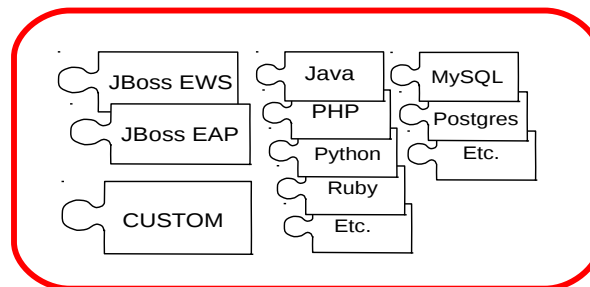


Figure 2.1.4: Cartridges

2.1.5 MongoDB

MongoDB is a NoSQL database that uses a JavaScript command syntax to store internal information about users, gears and other internal data. The data (aka “documents”) are stored in JavaScript Object Notation (JSON) format. OpenShift uses MongoDB to provide highly-available storage of user, node and application state to brokers.

In this reference architecture MongoDB runs as a service (***mongod***) on the OpenShift Enterprise *broker* hosts. For larger environments, it may be preferable to locate MongoDB on a set of hosts that are separate from broker hosts for performance purposes.



2.1.6 ActiveMQ

ActiveMQ is an open source messenger service that supports a wide range of programming languages and environments. OpenShift Enterprise uses ActiveMQ to manage communications between brokers and nodes. Brokers use the **MCollective** client to communicate to ActiveMQ which in turn communicates to the MCollective agent on the node.

In this reference architecture ActiveMQ runs as a service (*activemq*) on the *brokers*. For larger environments, it may be preferable to locate ActiveMQ on a set of hosts that are separate from broker hosts for performance purposes.

Figure 2.1.6: ActiveMQ depicts the relationship between ActiveMQ, the MCollective *client* and *agent*:

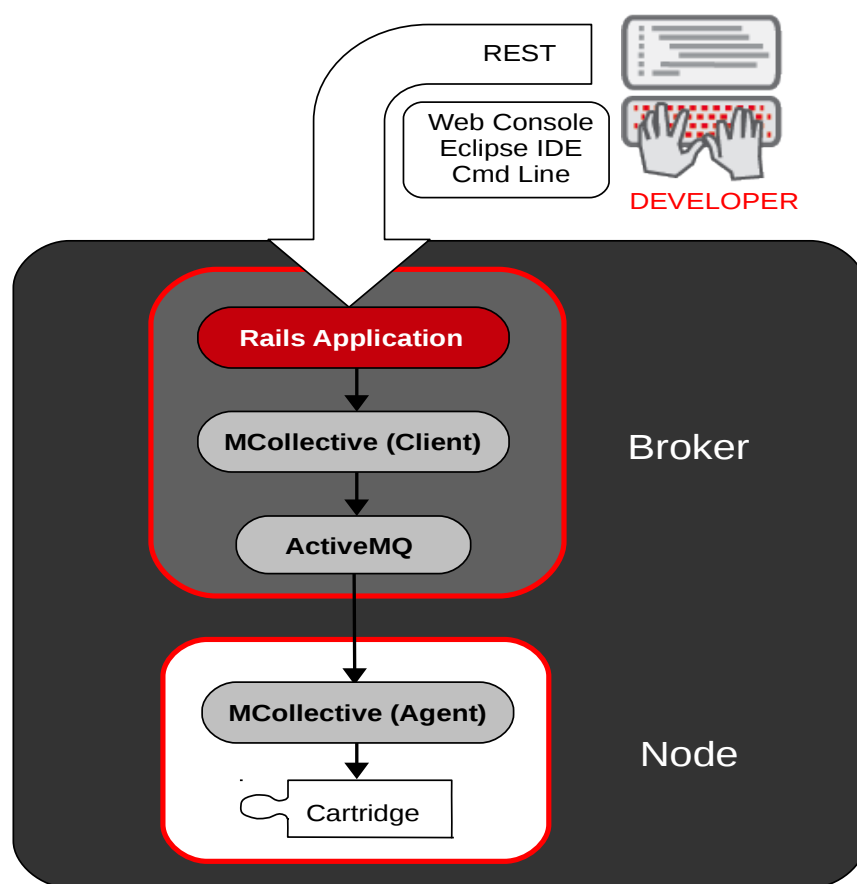


Figure 2.1.6: ActiveMQ



2.1.6.1 Network of ActiveMQ Brokers

For additional redundancy, **ActiveMQ** is configured in a **Network-of-Brokers** configuration. This provides the ability to distribute queues and topics among ActiveMQ brokers. In the event of a *broker* failure, this configuration allows client connections to fail over to any of the remaining brokers in the network. More information is provided on the [apache.org](http://activemq.apache.org/) website¹.

Figure 2.1.6.1 Network of ActiveMQ Brokers depicts the ActiveMQ network configuration:

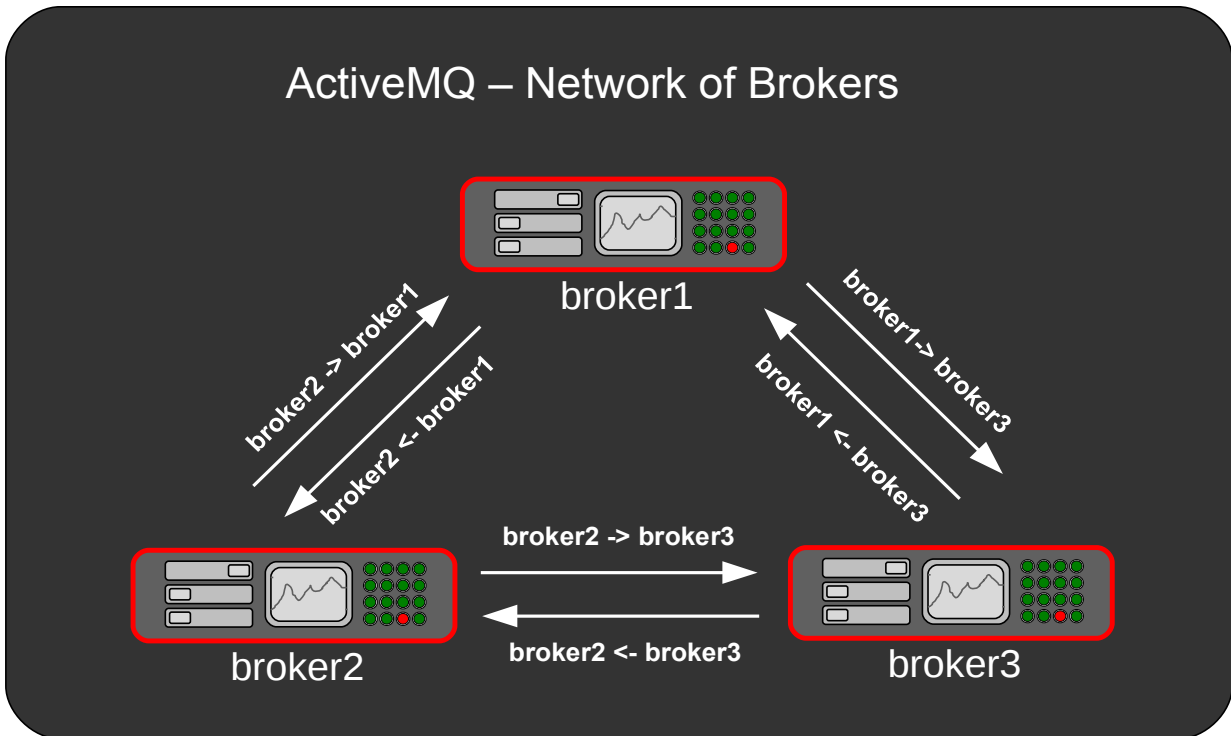


Figure 2.1.6-1: Network of ActiveMQ Brokers

In this reference architecture the ActiveMQ brokers are hosted on the OpenShift Enterprise *broker* hosts. Some environments may prefer to locate the ActiveMQ Network of Brokers on a separate set of hosts.

¹ <http://activemq.apache.org/>



2.1.7 Summary

Figure 2.1.7: OpenShift Summary View provides a complete view of OpenShift Enterprise from the developer/operator (devops) and user perspectives:

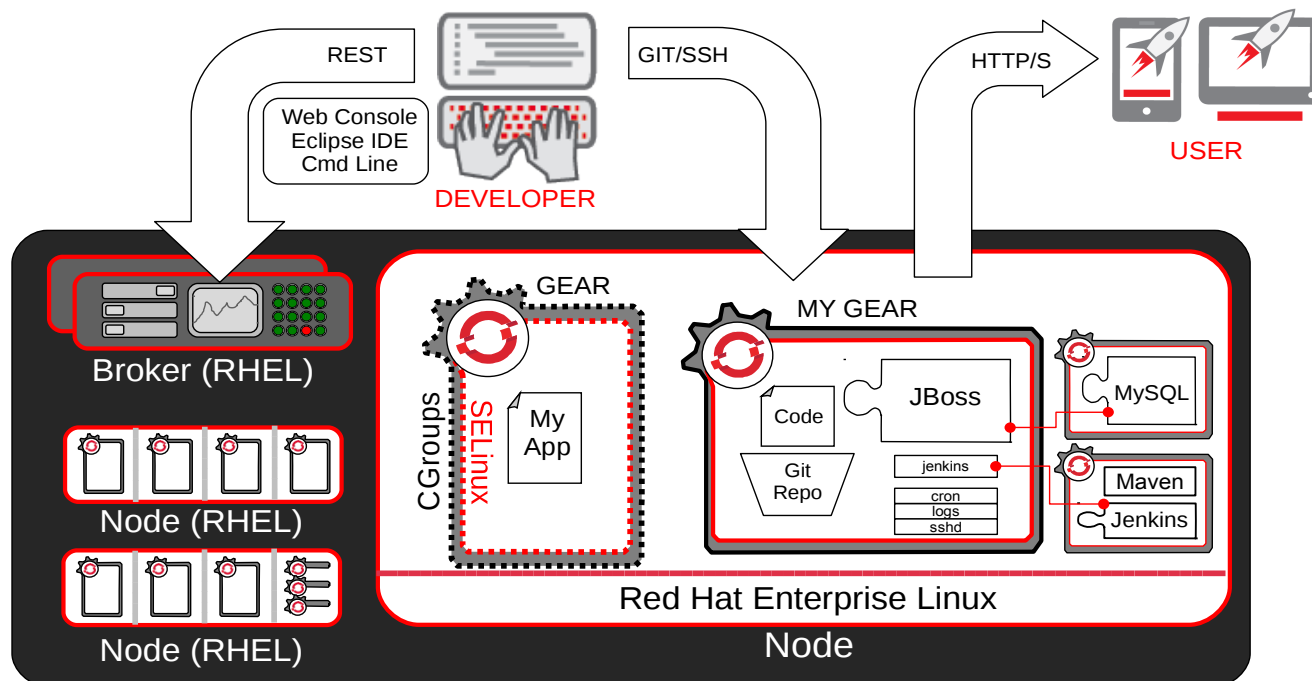


Figure 2.1.7: OpenShift Summary View



2.2 Identity Management (IdM) in Red Hat Enterprise Linux

Red Hat Identity Management (IdM) in RHEL is a domain controller for Linux and UNIX servers that uses native Linux tools. Similar to Active Directory, Identity Management provides centralized management of identity stores, authentication and authorization policies. Identity Management in RHEL is based on the upstream, open source FreeIPA project. Within this reference architecture the terms IdM and IPA are synonymous with one another.

Identity Management defines a domain, with servers and clients who share centrally-managed services, like Kerberos and DNS. Although centralized applications to manage identity, policy and authorization are not new, Identity Management is one of the only options that supports Linux/Unix domains.

Identity Management provides a unifying interface for standards-based, common network services, including PAM, LDAP, Kerberos, DNS, NTP, and certificate services, and allows Red Hat Enterprise Linux systems to serve as domain controllers.

Red Hat Identity Management in RHEL also supports Active Directory *Cross-realm Kerberos trusts*. Cross-realm trusts allow the existing user accounts within Active Directory to be used for logins on local Red Hat Enterprise hosts. Although this capability is still considered a technical preview feature under current versions of Red Hat Enterprise Linux, it's use is demonstrated here as an *optional* integration scenario.

For more information on Identity Management in Red Hat Enterprise Linux, see the [Red Hat Enterprise Linux 7 - Linux Domain, Identity, Authentication and Policy Guide](#).

The following sections describe the services included within Identity Management in Red Hat Enterprise Linux.

2.2.1 Kerberos - Key Distribution Center (KDC)

Kerberos is a network authentication protocol that uses symmetric key cryptography to provide highly secure authentication between client and server applications. Both Red Hat Enterprise Linux 6 and Windows server 2008 R2 are based on the current release of Kerberos - version 5.

Kerberos operates on the basis of “tickets” that are granted by a trusted third-party called a *key distribution center (KDC)*. The KDC maintains a secure database of secret keys that are known only to the KDC itself and the client requesting a ticket. Tickets have a configurable expiration date and must be refreshed by the client on a regular basis.

Kerberos authentication is significantly safer than normal password-based authentication because passwords are never sent over the network - even when services are accessed on other machines.

In Identity Management, the Kerberos administration server is set up on the IdM domain controller, and all Kerberos data is stored in the IdM Directory Server (LDAP). The Directory Server instance defines and enforces access controls for the Kerberos data.



2.2.2 389 Directory Server (LDAP)

Identity Management contains an internal instance of the **389 Directory Server**. All Kerberos information, user accounts, groups, services, policy information, DNS zone and host entries, and all other IdM data is stored within the 389 Directory Server.

The 389 Directory Server also supports multi-master replication for resiliency and increased availability.

2.2.3 Dogtag Certificate System

Kerberos can use certificates or keytabs for authentication and secure communications. Identity Management includes a certificate authority (CA) called **Dogtag Certificate System**. Dogtag Certificate System is responsible for issuing certificates to IdM servers, replicas, hosts and services within the IdM domain.

The IdM CA can be either a root CA, or a subordinate CA by having its policies defined from another CA external to IdM. The CA type is determined when the IdM server is configured.

2.2.4 Domain Name System (DNS)

Domain Name System is a hierarchical, distributed naming system for managing the mappings between human-friendly domain, host and service names to IP addresses. DNS also defines the protocol for DNS communication exchanges as part of the Internet Protocol (IP) suite. On Red Hat Enterprise Linux 6, DNS host resolution is configured in the file */etc/resolv.conf*.

Identity Management defines a domain — multiple machines with different users and services, each accessing shared resources and using shared identity, authentication, and policy configuration. The clients need to be able to contact each other, as IPA servers. Additionally, services like Kerberos depend on hostnames to identify their principal identities.

Hostnames are associated with IP address using the Domain Name Service (DNS). DNS maps hostnames to IP addresses and IP addresses to hostnames, providing a resource that clients can use when they need to look up a host. From the time a client is enrolled in the IPA domain, it uses DNS to locate the IPA server and then all of the services and clients within the domain.

Multiple DNS servers are usually configured, each one working as an authoritative resource for machines within a specific domain. Having the IPA server also be a DNS server is optional, but it is strongly recommended. When the IPA server also manages DNS, there is tight integration between the DNS zones and the IPA clients and the DNS configuration can be managed using native IPA tools. Even if an IPA server is a DNS server, other external DNS servers can still be used.



2.2.5 Time Services (ntpd, chronyd)

Network Time Protocol (NTP) is an Internet protocol used to synchronize computer system clocks to a reference time source. On Red Hat Enterprise Linux 6, the **ntpd** daemon handles synchronization. NTP parameters are configured in the file `/etc/ntp.conf`.

By default, Identity Management installs and configures an NTP server which is used by the domain to synchronize clocks for other servers, replicas, clients and services within the IdM domain. Time synchronization is essential for these services to function properly.

IdM servers are not required to host an NTP server but it is the default configuration during installation. For IdM servers deployed on a virtual machine, NTP should **not** be configured. To prevent NTP from being installed during IdM server installation, use the `--no-ntp` option.

On Red Hat Enterprise Linux 7, the default daemon to handle time services is **chronyd**. Similar in functionality to **ntpd**, **chronyd** is tailored to the roaming nature of laptops and mobile devices.

In this reference architecture, **ntpd** is used for all Red Hat Enterprise Linux 6 and 7 hosts.

2.2.6 IdM Replication

Identity Management in Red Hat Enterprise Linux supports the use of replica servers to provide improved availability and balance client requests across the IdM domain. IdM *replica* servers are based on the configuration of an existing IdM server. Once the replication agreement is established, the replica server is functionally identical to the IdM server.

Figure 2.2.6: IdM Replication depicts the relationship between the IdM server and replica:

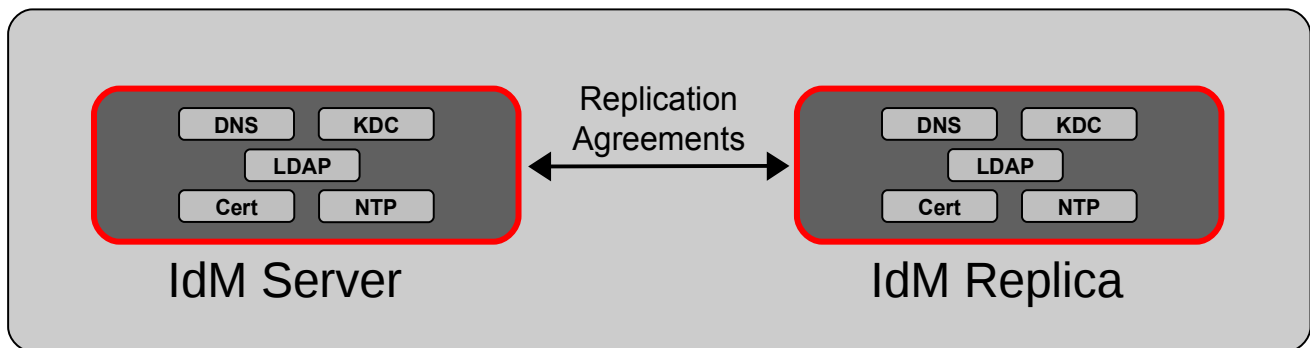


Figure 2.2.6: IdM Replication

IdM servers and replicas both use underlying LDAP directories to store user and host entries, configuration data, policy configuration, keytabs, certificates and keys. Servers and replicas propagate data among each other through multi-master replication agreements. Both servers and replicas are masters (peers) in the replication topology.

Since all servers within an IdM domain are LDAP peers, replication must conform to the same topology limits of a 389 Directory Server domain. The maximum number of peer servers in the IdM domain is 20.



2.2.7 Cross-realm Kerberos Trusts

One of the newer features of Identity Management in Red Hat Enterprise Linux is **Cross-realm Kerberos trusts**. Trusts are a peer based relationship created between the IdM domain and the Windows Active Directory domain. Users within a cross-realm Kerberos trust are permitted to obtain tickets and connect to the hosts and services in the trusted realm as if they were a member of the peer realm.

Figure 2.2.7: Cross-realm Kerberos Trusts depicts the relationship between IdM and Active Directory domains when trusts are in place:

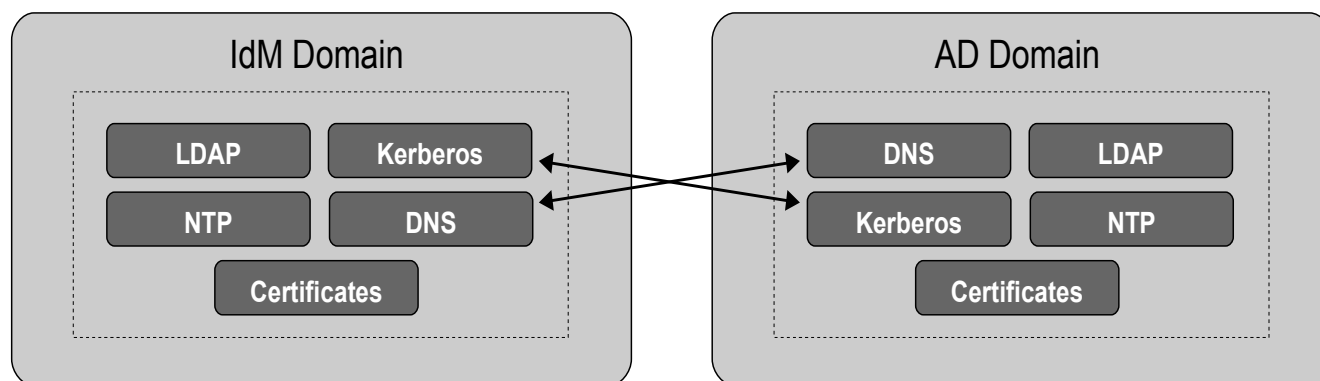


Figure 2.2.7: Cross-realm Kerberos Trusts

Trusts created through IdM centralize and simplify the complexities of trust relationships between Windows and Linux domains. The common entry points are through Kerberos (for authentication) and DNS (for host and service) lookups.

Red Hat Enterprise Linux clients utilize both the System Security Services Daemon (SSSD) and Samba when accessing Active Directory resources through IdM.

This reference architecture demonstrates the configuration of a cross-realm Kerberos trust using two IdM servers (one server, one replica), a configuration known as a *multi-master* trust. The use of cross-realm Kerberos trusts is optional but described in detail in **IS5: Cross-realm Kerberos Trust (optional)**.

For more information on cross-realm Kerberos trusts and associated areas within Identity Management, see the [Red Hat Enterprise Linux 7 - Windows Integration Guide](#).

2.3 Red Hat Enterprise Virtualization (RHEV)

Red Hat Enterprise Virtualization (RHEV) combines the KVM hypervisor (powered by the Red Hat Enterprise Linux kernel) with an enterprise grade, multi-hypervisor management platform that provides key virtualization features such as live migration, high availability, power management, and virtual machine life cycle management. 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 consists of the following two components:

- **RHEV Manager (RHEV-M)**: A feature-rich virtualization management system that provides advanced capabilities for hosts and guests.
- **RHEV Hypervisor**: A modern, scalable, high performance hypervisor based on RHEL KVM. It can be deployed as *RHEV-H*, a small footprint secure hypervisor image included with the RHEV subscription, or as a RHEL server (*RHEL-H* purchased separately) managed by **RHEV-M**.

All OpenShift Enterprise and Identity Management hosts, and clients within this reference architecture were deployed using RHEV. The underlying RHEV infrastructure consists of two, *RHEL-H* hypervisors deployed as a datacenter consisting of two HP ProLiant BL460c G6 blades and a Dell/Equallogic PS4110e iSCSI storage array. The data center is managed through a RHEV-M virtual machine hosted on an independent KVM server.

2.4 Red Hat Enterprise Linux (RHEL)

Red Hat Enterprise Linux 6 is Red Hat's trusted datacenter platform and 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 6.6 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



Red Hat Enterprise Linux 7 is Red Hat's latest release of its flagship platform and delivers dramatic improvements in reliability, performance and scalability. A wealth of new features provide architects, system administrators and developers with the resources necessary to innovate and manage more efficiently:

Linux Containers:

- Combine lightweight application isolation with flexible of image-based deployment
- Provide secure multi-tenancy through the use of control groups (Cgroups) for resource management, namespaces for process isolation and SELinux for security

Identity Management:

- *Cross-realm Kerberos trusts* permit users with Active Directory credentials to access Linux resources without additional identity authentication
- *Realmd* automatically discovers information about Microsoft Active Directory domains and simplifies the configuration needed to join a domain

Management

- **systemd** a system and service manager that combines compatibility for most SysV and LSB init scripts, offers on-demand starting of daemons and fine-grained service control
- *OpenLMI* provides a common infrastructure for the remote management and monitoring of hardware, operating systems and system services

File Systems

- *XFS* is now included as the default file system, providing scalability and supporting file system sizes up to 500TB
- *Ext4* supports file system sizes of 50TB
- *Btrfs* introduced as a Technology Preview, Btrfs includes volume management, snapshot support, data and metadata integrity checking for large-scale use cases

There are many other feature enhancements to Red Hat Enterprise Linux 7. Please see the Red Hat website for more information².

2.5 Windows Server 2012 R2

Windows Server 2012 R2 is Microsoft's enterprise operating system for businesses and provides features for virtualization, power savings, manageability and mobile access.

Windows Server 2012 R2 is available in four editions – Foundation, Essentials, Standard, and Datacenter. Windows Server 2012 R2 Datacenter edition is used in this reference architecture.

² <http://www.redhat.com/products/enterprise-linux/>



2.5.1 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.

In this reference architecture, user accounts within an existing Active Directory domain are authenticated through an IdM Cross-realm Kerberos trust.

2.6 System Security Services Daemon (SSSD)

The System Security Services Daemon (SSSD) provides access to different identity and authentication providers. SSSD is an intermediary between local clients and any configured data store. The local clients connect to SSSD and then SSSD contacts the external providers. This brings a number of benefits for administrators:

- Offline authentication. SSSD can optionally keep a cache of user identities and credentials that it retrieves from remote authentication/identification services. This allows users to authenticate to resources successfully, even if the remote identification server is offline or the local machine is offline.
- Reduced load on authentication/identification servers. Rather than having every client contact the identification server directly, all local clients can contact SSSD which can connect to the identification server or check its cache.
- Single user account. Remote users frequently have multiple user accounts, such as one for their local system and one for the organizational system. Since SSSD supports caching and offline authentication, remote users can connect to network resources simply by authenticating to their local machine and then SSSD maintains their network credentials.

SSSD recognizes domains, which are associated with different identity servers. Domains are a combination of an identity provider and an authentication method. SSSD works with LDAP identity providers (OpenLDAP, Red Hat Directory Server, IdM in RHEL, Microsoft Active Directory) and native LDAP authentication or Kerberos authentication.

In this reference architecture, SSSD is automatically configured as part of the IdM client installation process.



2.7 RHC Client Tools

The OpenShift Client tools are known as **rhc** and are built and packaged using the Ruby programming language. OpenShift integrates with the Git version control system to provide powerful, decentralized version control for the source code of applications deployed within OpenShift.

OpenShift **rhc** can be run on any operating system with Ruby 1.8.7 or higher. All rhc commands are run from a command line (*aka - Terminal*) window. In this reference architecture, rhc is only installed on the rhc client to verify OpenShift functionality.

The installation of the RHC client tool is detailed in section **6.5 Deploy RHC Client**.

2.8 IPA Admin Tools

IdM provides a client tool to assist with the most common IdM server tasks from a client within the IdM domain.

The installation of the IPA admin tool is described in section **5.2.6 Install Admin Tools (optional)**.



3 Reference Architecture Configuration

The sections that follow describe the systems and configurations used in the development of this reference architecture.

3.1 Environment

Figure 3.1: Environment Overview depicts the overall systems deployed in this reference architecture environment:

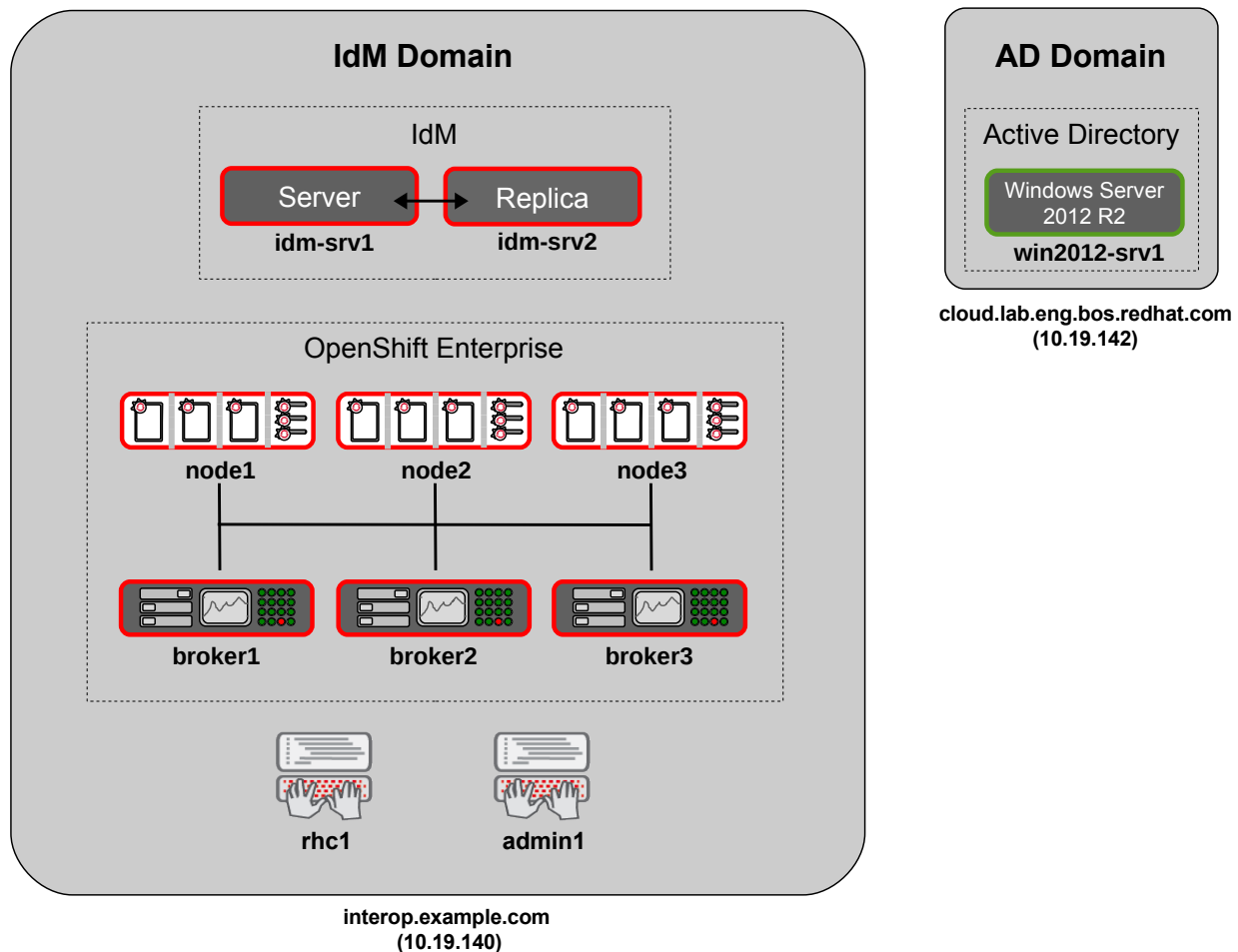


Figure 3.1: Environment Overview



The OpenShift Enterprise environment is deployed with three brokers and three nodes to create a distributed environment for increased availability. In this configuration, each OpenShift broker shares the messaging (ActiveMQ) and database (MongoDB) services.

MongoDB is a NoSQL database that stores internal information about users, gears and other internal data. MongoDB provides fast, scalable access for deployed applications. For improved availability, MongoDB is configured with replica sets where one broker functions as the primary, and the remaining brokers act as secondaries. In the event of a failure to the primary broker, one of the secondaries is elected and assumes the role of the MongoDB master.

Similarly, ActiveMQ is configured in a Network-of-Brokers configuration. This provides the ability to distribute queues and topics among ActiveMQ brokers. In the event of a broker failure, this allows client connections to fail over to any of the remaining brokers in the network.

The IdM environment is deployed with two servers - one master, one replica. Clients connect to the services provided by IdM through the use of DNS discovery. In the event of an IdM server failure, clients continue operating through the surviving IdM server. The configuration and use of DNS discovery is automatic and transparent to IdM clients.

IdM to Windows Active Directory Cross-realm Kerberos trusts are also presented for the authentication of existing Active Directory domains. The details of Cross-realm Kerberos trusts are described in section **7.3.5 IS5: Cross-realm Kerberos Trust (optional)**.

From a sequencing perspective, the IdM infrastructure is deployed first in order to establish a solid framework for providing services to all IdM clients. Next, the OpenShift Enterprise infrastructure is deployed and verified. Once these two environments have been fully configured, then the process of integrating OpenShift with Identity Management (IdM) begins.

For existing OpenShift environments, most integration tasks are still applicable or can be adjusted as required by the local environment once the IdM infrastructure has been deployed and made available.



3.2 Virtual Machines

All systems in this reference architecture are virtual machines except for the Red Hat Enterprise Virtualization (RHEV) hypervisors. The virtual machines and network addresses are listed in **Table 3.2.1: Virtual Machine Sizing** and **Table 3.2.2: Network Addresses** below:

3.2.1 Virtual Machine Sizing

Hostname	Disk	Memory	CPU
broker1.interop.example.com	20 GB	8 GB	2 (virtual cores)
broker2.interop.example.com	20 GB	8 GB	2 (virtual cores)
broker3.interop.example.com	20 GB	8 GB	2 (virtual cores)
node1.interop.example.com	20 GB	8 GB	2 (virtual cores)
node2.interop.example.com	20 GB	8 GB	2 (virtual cores)
node3.interop.example.com	20 GB	8 GB	2 (virtual cores)
ldm-srv1.interop.example.com	20 GB	8 GB	2 (virtual cores)
ldm-srv2.interop.example.com	20 GB	8 GB	2 (virtual cores)
admin1.interop.example.com	10 GB	4 GB	2 (virtual cores)
rhc1.interop.example.com	10 GB	4 GB	2 (virtual cores)
win-srv1.cloud.lab.eng.bos.redhat.com	50 GB	8 GB	4 (virtual cores)

Table 3.2.1: Virtual Machine Sizing

3.2.2 Network Addresses

Hostname	IP Address
broker1.interop.example.com	10.19.140.11
broker2.interop.example.com	10.19.140.12
broker3.interop.example.com	10.19.140.13
node1.interop.example.com	10.19.140.21
node2.interop.example.com	10.19.140.22
node3.interop.example.com	10.19.140.23
ldm-srv1.interop.example.com	10.19.140.101
ldm-srv2.interop.example.com	10.19.140.102
admin1.interop.example.com	10.19.140.100
rhc1.interop.example.com	10.19.140.10
win-srv1.cloud.lab.eng.bos.redhat.com	10.19.142.100

Table 3.2.2: Network Addresses



3.3 Software Configuration

3.3.1 Required Software Channels

The required software channels and associated subscription names are shown below in

Table: 3.3.1 Required Software Channels:

Hostname	Channel	Subscription
broker1 broker2 broker3	Red Hat Enterprise Linux Server	"Red Hat Enterprise Linux 6 Server"
	Red Hat OpenShift Enterprise 2.2 Infrastructure - x86_64	"OpenShift Enterprise 2.2 Broker Infrastructure"
	Red Hat Software Collections 1	"OpenShift Enterprise 2.2 Broker Infrastructure"
node1 node2 node3	Red Hat Enterprise Linux Server	"Red Hat Enterprise Linux 6 Server"
	Red Hat OpenShift Enterprise Application Node	"OpenShift Enterprise"
	Red Hat OpenShift Enterprise 2.2 JBoss EAP Add-On	"JBoss Enterprise Application Platform for OpenShift Enterprise"
	JBoss Enterprise Application Platform	"JBoss Enterprise Application Platform for OpenShift Enterprise"
	JBoss Enterprise Web Server 2	"OpenShift Enterprise"
	Red Hat OpenShift Enterprise Client Tools 2.1	"OpenShift Enterprise"
	Red Hat Software Collections 1	"OpenShift Enterprise"
rhc1	Red Hat Enterprise Linux Server	"Red Hat Enterprise Linux 6 Server"
	Red Hat OpenShift Enterprise Client Tools 2.1	"OpenShift Enterprise"
idm-srv1 idm-srv2	Red Hat Enterprise Linux Server	"Red Hat Enterprise Linux 7 Server"
admin1	Red Hat Enterprise Linux Server	"Red Hat Enterprise Linux 7 Server"

Table 3.3.1: Required Software Channels



3.3.2 Software Versions

The versions of each software component are shown in **Table 3.3.2: Software Versions**:

Host	Software	Version
broker1 broker2 broker3	openshift-origin-broker	1.16.1.6
	openshift-origin-broker-util	1.23.8.4
	openshift-origin-console	1.16.2.6
	ruby193-mcollective-client	2.4.1
	rubygem-openshift-origin-admin-console	1.20.2.1
	rubygem-openshift-origin-auth-remote-user	1.19.6.2
	rubygem-openshift-origin-dns-nsupdate	1.16.2.1
	rubygem-openshift-origin-msg-broker-mcollective	1.23.3.1
	activemq	5.9.0
	mongodb	2.4.6
	mongodb-server	2.4.6
	rhc	1.24.3.1
node1 node2 node3	rubygem-openshift-origin-node	1.22.2
	openshift-origin-node-util	1.22.9.1
	polycoreutils-python	2.0.83
	rsyslog7	7.4.7
	rsyslog7-mmopenshift	7.4.7
	ruby193-mcollective	2.4.1
	ruby193-rubygem-passenger-native	3.0.21
	rubygem-openshift-origin-container-selinux	0.8.1.2
	rubygem-openshift-origin-frontend-apache-mod-rewrite	0.5.2.1
	rubygem-openshift-origin-frontend-nodejs-websocket	0.4.1
	rubygem-openshift-origin-node	1.23.9.11

Table 3.3.2: Software Versions



Host	Software	Version
idm-srv1	ipa-server	3.3.3
	bind	9/9/4
	bind-dyndb-ldap	3.5
idm-srv2	ipa-server	3.3.3
	bind	9.9.4
	bind-dyndb-ldap	3.5
admin1	ipa-client	3.3.3
rhc1	ipa-client	3.3.3
	rhc	1.24.3.1-1

Table 3.3.2: Software Versions (continued)



3.3.3 Security

SELinux

All Red Hat Enterprise Linux systems are configured with SELinux enabled and enforcing.

3.3.4 Network Ports

A summary of the network ports and services is shown in **Table 3.3.4: Network Ports**:

Host	Port	Protocol	Service/Description
broker1 broker2 broker3 (ose-broker-chain)	22	TCP	Remote administration (ssh)
	80	TCP	HTTP access (automatically forwarded to HTTPS:443)
	443	TCP	HTTPS access to broker (via REST API) by rhc client. HTTPS access to Management Console
	27017	TCP	MongoDB
	61613	TCP	ActiveMQ
	61616	TCP	ActiveMQ communications. (network-of-brokers configurations only)
node1 node2 node3 (ose-node-chain)	22	TCP	Remote administration (ssh)
			rsync access to gears (for moving between nodes)
	80	TCP	HTTP requests to hosted applications (automatically forwarded to HTTPS:443)
	443	TCP	HTTPS requests to hosted applications
	8000	TCP	WebSocket connections to hosted applications (optional)
	8443	TCP	Secure WebSocket connections to hosted applications (optional)
	2303- 2308	TCP	SNI port proxy to gears (optional)
	35531- 65535	TCP	Port Proxy (applications/gears)

Table 3.3.4: Network Ports



Host	Port	Protocol	Service/Description
idm-srv1 idm-srv2 (ipa-server-chain)	80	TCP	HTTP console access
	443	TCP	HTTPS console access (automatically forwarded to HTTPS:443)
	389	TCP	LDAP
	636	TCP	Secured LDAP (LDAPS)
	88	TCP, UDP	Kerberos
	464	TCP, UDP	Kerberos
	53	TCP, UDP	DNS
	123	UDP	NTP
	7389	TCP	Dogtag Certificate System - LDAP
broker1 broker2 broker3 node1 node2 node3 rhc-1 admin1 (ipa-client-chain)	80	TCP	HTTP console access
	443	TCP	HTTPS console access (automatically forwarded to HTTPS:443)
	389	TCP	LDAP
	636	TCP	Secured LDAP (LDAPS)
	88	TCP, UDP	Kerberos
	464	TCP, UDP	Kerberos
	53	TCP, UDP	DNS
	123	UDP	NTP

Table 3.3.4: Network Ports (continued)

For greater control and flexibility, the firewalls on all hosts are configured using named chains - i.e. *ose-broker-chain*, *ose-node-chain*, *ipa-server-chain*, *ipa-client-chain*.



3.4 Hardware Configuration

3.4.1 Hypervisors

Both RHEV hypervisors are configured identically across two HP ProLiant BL460C blades housed within an HP BladeCenter C7000 chassis as listed in **Table 3.4.1: Hypervisor Hardware**:

Component	Description
Hostname	rhel-h1.interop, rhel-h2.interop
Operating System	Red Hat Enterprise Linux 6.5 (Santiago) (2.6.32-431.17.1.el6.x86_64 kernel)
System Type	HP ProLiant BL460c G6
Processor	Quad Socket, Quad Core (16 cores) Intel® Xeon® CPU X5550 @2.67GHz
Memory	48 GB
Storage	4 x 146 GB SATA internal disk drive (RAID 1) 2 x Qlogic QMH2562 8Gb FC HBA
Network	8 x Broadcom NetXtreme II BCM57711E XGb

Table 3.4.1: Hypervisor Hardware

3.4.2 Storage

Storage for all RHEV virtual machines is provided by a Dell-EqualLogic PS4110e iSCSI array as listed in **Table 3.4.2: Storage Hardware**:

Component	Description
Hostname	ra-ps4110e-01
System Type	Dell/EqualLogic PS-4110e
Controllers	2 x 70-0478 (Type 17) eth0 - 10Gbps (iSCSI) eth1 – 100Mbps (Management)
Firmware	V5.2.2 (R229536)
Physical Drives	12 x 2 TB SAS drives (7200 rpm)
RAID Policy	RAID 6

Table 3.4.2: Storage Hardware



3.5 Component Log Files

Log files are essential for monitoring the status of components and applications. The following sections are provided to assist in troubleshooting and tracking the state of common system level components and applications. **Table 3.5: System Log Files** provides a summary of the default system log file locations for each components:

Component	Hosts	Log File(s)	Description
Red Hat Enterprise Linux	All	/var/log/messages	System events
Red Hat Enterprise Linux	All	/var/log/dmesg	Boot messages
Red Hat Enterprise Linux	All	/var/log/secure	Security, authentication messages
DNS	All	/var/log/messages	DNS messages
SSSD	IdM Clients	/var/log/sss/sssd.log	SSSD daemon (sss d)
SSSD	IdM Clients	/var/log/sss/sssd_pam.log	PAM events
NTP	All	/var/log/messages	NTP events (ntpd)
Kerberos	IdM Clients	Standard out (terminal screen)	Kerberos client messages
Kerberos	IdM Server	/var/log/krb5kdc.log	Kerberos KDC server messages
Kerberos	IdM Servers	/var/log/kadmin.log	Kerberos admin server messages
IdM	IdM Servers	/var/log/ipaserver-install.log	IdM server installation log
IdM	IdM Replica	/var/log/ipareplica-install.log /var/log/ipareplica-conncheck.log	IdM replica installation log IdM replica connection check log
IdM	IdM Clients, Servers	/var/log/ipaclient-install.log /var/log/ipaclient-uninstall.log	IdM client installation log IdM client uninstallation log
IdM	IdM Clients, Servers	~/ipa/log/cli.log	IdM cli tools log

Table 3.5: System Log Files



Component	Hosts	Log File(s)	Description
Apache Server	All	/var/log/httpd/access /var/log/httpd/error	Apache access and error logs.
389 Directory Server	IdM Servers	/var/log/dirsrv/slapd- {REALM} /access /var/log/dirsrv/slapd- {REALM} /audit /var/log/dirsrv/slapd- {REALM} /errors	Directory server attempted access, operations and error messages
OpenShift	Brokers	/var/log/httpd/*	Apache web server messages
OpenShift	Brokers	/var/log/openshift/broker/httpd/access_log /var/log/openshift/broker/httpd/error_log	Broker web log
OpenShift	Brokers	/var/log/openshift/broker/production.log	Broker production log
OpenShift	Brokers	/var/log/openshift/broker/user_action.log	User action log
OpenShift	Brokers	/var/log/openshift/console/httpd/access_log /var/log/openshift/console/httpd/error_log	Console web log
OpenShift	Brokers	/var/log/openshift/console/production.log	Console (GUI) production log
OpenShift	Brokers	/var/log/activemq/*.log	Active MQ messages
OpenShift	Brokers	/var/log/mongodb/mongodb.log	MongoDB messages
OpenShift	Nodes	/var/log/openshift/node/ruby193-mcollective.log	MCollective (Agent) messages
OpenShift	Nodes	/var/log/openshift/node/cgroups.log	Cgroups log
OpenShift	RHC Clients	rhc tail {appName}	Current application log entries ³

Table 3.5: System Log Files (continued)

³ To view the entire log, **ssh** onto the gear(s) on which the language framework/cartridge is installed using [this FAQ](#) and run:
more ~/{cartridgeID}/logs/*.log where {cartridgeID} is the framework cartridge e.g. - nodejs-0.6, mysql-5.1.



By default, most events in Red Hat Enterprise Linux are sent via **syslog** to the default `/var/log/messages`. In OpenShift 2.1, the ability to consolidate the logging of broker and node events via syslog was introduced. See the OpenShift Administration 2 - Administration Guide⁴ for further details.

In addition, many daemons have debug mode capabilities that can be enabled through configuration files (e.g. - `/etc/krb5.conf`) or via command line flags. Consult the on-line man pages and Red Hat documentation for further details.

4 [OpenShift Enterprise 2- Administration Guide](#)



4 Staging the Infrastructure

4.1 Red Hat Enterprise Virtualization (RHEV)

This reference architecture uses Red Hat Enterprise Virtualization (**RHEV**) to host the virtual machines deployed for the OpenShift Enterprise 2.2 and Identity Management (**IdM**) in Red Hat Enterprise Linux 7 environments. The OpenShift RHC and IdM admin clients are also deployed as **RHEV** virtual machines. Storage for all **RHEV** machines is located on a single, 1TB iSCSI volume provisioned by a Dell-EqualLogic PS4110E array. Two HP ProLiant BL460C blades provisioned for use as hypervisors (**RHEL - H**) share access to the iSCSI volume across the **RHEV** cluster. Management of the **RHEV** environment is done from a **RHEV-M** virtual machine hosted on remote a **KVM** server.

Figure 4.1: RHEV Environment depicts the infrastructure providing the virtual machines:

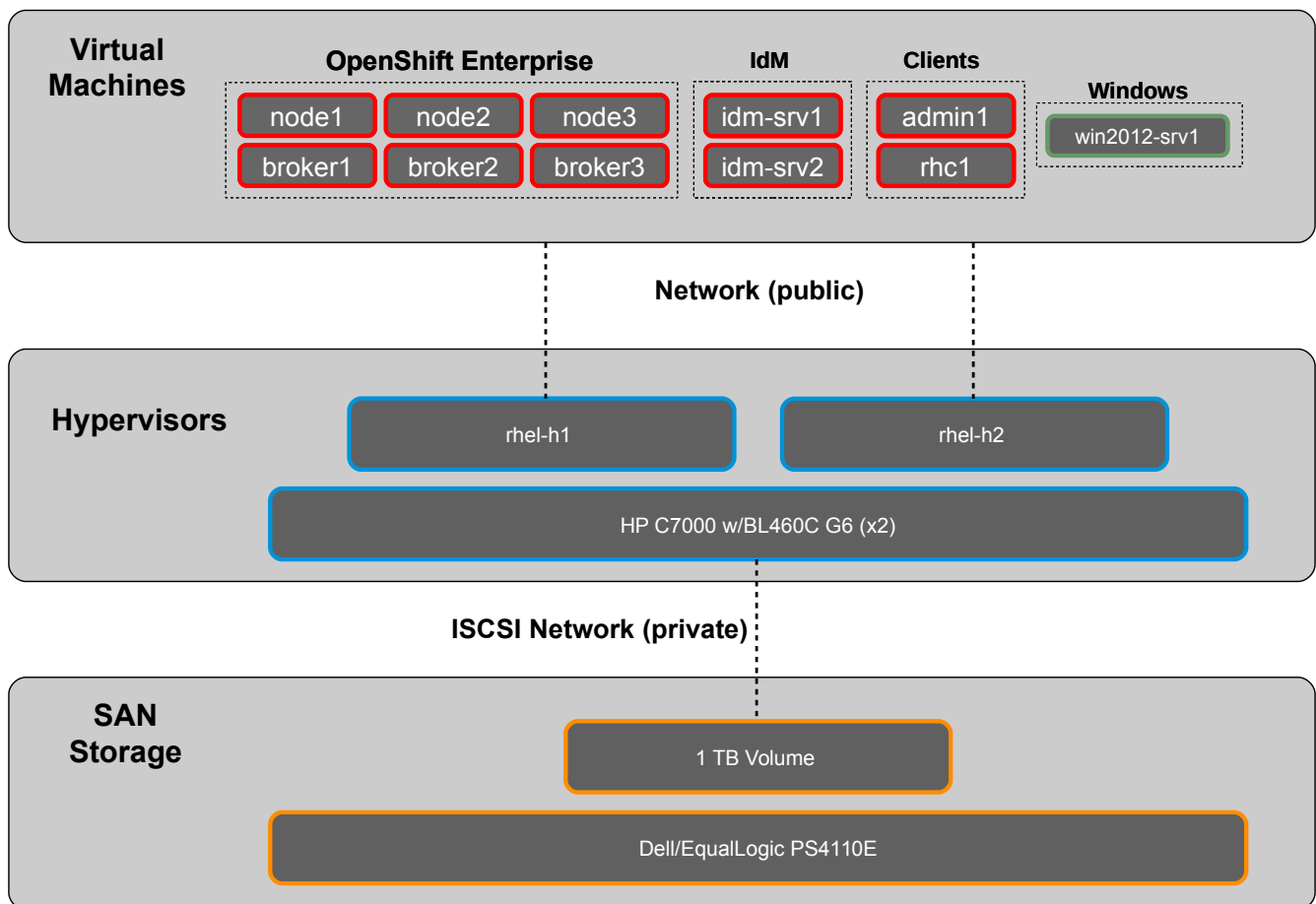


Figure 4.1: RHEV Environment



4.2 Install Red Hat Enterprise Linux

Deploy all OpenShift systems (**broker1**, **broker2**, **broker3**, **node1**, **node2**, **node3**, **rhc1**) on Red Hat Enterprise Linux 6.6 and all Identity Management systems (**idm-srv1**, **idm-srv2**, **admin1**) on Red Hat Enterprise Linux 7 by following the instructions appropriate for each release:

- Red Hat Enterprise Linux 6 Installation Guide⁵
- Red Hat Enterprise Linux 7 Installation Guide⁶

4.3 Configure Network

Configure the network interfaces on each system to use static IP addressing as per **Table 4.3: Host IP Addresses** below:

Hostname	IP Address
rhc1.interop.example.com	10.19.140.10
broker1.interop.example.com	10.19.140.11
broker2.interop.example.com	10.19.140.12
broker3.interop.example.com	10.19.140.13
node1.interop.example.com	10.19.140.21
node2.interop.example.com	10.19.140.22
node3.interop.example.com	10.19.140.23
admin1.interop.example.com	10.19.140.100
idm-srv1.interop.example.com	10.19.140.101
idm-srv2.interop.example.com	10.19.140.102

Table 4.3: Host IP Addresses

1. Edit the file `/etc/sysconfig/network-scripts/ifcfg-eth0` on each host using the appropriate IP addresses. The items in bold are unique to each host and must be adjusted accordingly.

```
DEVICE="eth0"
TYPE="Ethernet"
HWADDR="0A:0A:0A:00:00:65"
UUID="f7cd399c-132d-4206-8cbd-9508c7b0d101"
NM_CONTROLLED="no"
ONBOOT="yes"
BOOTPROTO="none"
IPADDR=10.19.140.101
NETMASK=255.255.248.0
GATEWAY=10.19.143.254
```

⁵ [Red Hat Enterprise Linux 6 - Installation Guide](#)

⁶ [Red Hat Enterprise Linux 7 - Installation Guide](#)



```
DEFROUTE="yes"  
PEERDNS="no"  
PEERNTP="yes"  
IPV4_FAILURE_FATAL="no"  
IPV6INIT="no"  
IPV6_FAILURE_FATAL="no"  
NAME="System eth0"
```

2. Restart the interface on each host after the changes are saved and verify the updates.

Red Hat Enterprise Linux 6

```
# service network restart  
# ip a
```

Red Hat Enterprise Linux 7

```
# systemctl restart network  
# ip a
```

3. Stop and disable **NetworkManager** to avoid conflicts caused by DHCP overwriting the static IP address configuration or the DNS resolver file (*/etc/resolv.conf*).

Red Hat Enterprise Linux 6

```
# service NetworkManager stop  
# chkconfig NetworkManager off
```

Red Hat Enterprise Linux 7

```
# systemctl stop NetworkManager  
# systemctl mask NetworkManager
```

4.4 Configure Domain Name System (DNS)

Configure DNS lookups on each system as follows.

1. Edit the file */etc/resolv.conf* to include the DNS servers in use in the local environment. Insure that the fully qualified domain name (FQDN) is specified for the **domain** and **search** parameters.

```
domain interop.example.com  
search interop.example.com  
nameserver 10.19.143.247  
nameserver 10.19.143.248  
nameserver 10.19.255.2
```

There are many approaches to configuring DNS. In this reference architecture, DNS lookups for OpenShift systems and all clients are changed to resolve through the IdM servers. This change is activated *after* the IdM infrastructure is deployed. For now, select the most appropriate DNS servers in your environment to allow access to Red Hat product updates and entitlements.



2. Configure the Fully Qualified Domain Name (FQDN) on each system.

Red Hat Enterprise Linux 6

Edit the file `/etc/sysconfig/network` and insert the appropriate hostname:

```
HOSTNAME={hostname}.interop.example.com
```

Red Hat Enterprise Linux 7

Edit the file `/etc/hostname` and insert the appropriate hostname:

```
{hostname}.interop.example.com
```

4.5 Configure Time Service (NTP)

Configure the time service on each system as follows.

Red Hat Enterprise Linux 6 Hosts (rhc1, broker1, broker2, broker3, node1, node2, node3)

1. Edit the file `/etc/ntp.conf` so that the time on each system is synchronized from a known, reliable time service.

```
# Enable writing of statistics records.
#statistics clockstats cryptostats loopstats peerstats
server 10.19.255.2
server 10.19.255.3
```

2. Activate the change by restarting the ntp daemon (**ntpd**) and enabling the daemon to start on boot on each system.

```
# service ntpd restart
Shutting down ntpd: [ OK ]
Starting ntpd: [ OK ]
# chkconfig ntpd on
# chkconfig ntpd --list
ntpd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Red Hat Enterprise Linux 7 Hosts (admin1, idm-srv1, idm-srv2)

1. Edit the file `/etc/ntp.conf` so that the time on each system is synchronized from a known, reliable time service.

```
# Enable writing of statistics records.
#statistics clockstats cryptostats loopstats peerstats
server 10.19.255.2
server 10.19.255.3
```




2. Disable the default time service daemon (**chronyd**) and disable it from starting on boot.

```
# systemctl stop chronyd
# systemctl disable chronyd
rm '/etc/systemd/system/multi-user.target.wants/chronyd.service'
# systemctl status chronyd
chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled)
   Active: inactive (dead)
```

3. Activate the change by restarting the ntp daemon (**ntpd**) and enabling the daemon to start on boot on each system.

```
# systemctl restart ntpd
# systemctl enable ntpd
ln -s '/usr/lib/systemd/system/ntpd.service' '/etc/systemd/system/multi-user.target.wants/ntpd.service'
# systemctl status ntpd
ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled)
   Active: active (running) since Tue 2014-06-17 14:50:13 EDT; 1s ago
   Process: 5475 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 5476 (ntpd)
   CGroup: /system.slice/ntpd.service
           └─5476 /usr/sbin/ntpd -u ntp:ntp -g
```

4.6 Configure SELinux

By default, SELinux is enabled during the installation of Red Hat Enterprise Linux. Red Hat recommends running with SELinux enabled at all times for maximum security. Ensure that SELinux is enabled and configured on boot for each system as follows.

1. Verify whether or not SELinux is enabled using the **getenforce** utility.

```
# getenforce
Enforcing
```

2. If **getenforce** returns "Permissive" then set to "Enforcing" and verify.

```
# getenforce
Permissive
# setenforce 1
# getenforce
Enforcing
```

3. Edit the file `/etc/selinux/config` and set SELinux to be persistent across reboots.

```
SELINUX=enforcing
```



4.7 Update Hosts File

On each host, edit the local `/etc/hosts` file to include the IP addresses, fully qualified domain name (FQDN) and aliases of all hosts.

```
#
# Master /etc/hosts file for OSE-IdM integration project
#
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
#
# Infrastructure - Win/AD, DNS
#
10.19.142.101 win2012-srv1.cloud.lab.eng.bos.redhat.com win2012-srv1
10.19.143.247 ra-ns1.cloud.lab.eng.bos.redhat.com      ra-ns1
#
# OpenShift Enterprise:
#
10.19.140.10   rhc1.interop.example.com                rhc1
10.19.140.11   broker1.interop.example.com              broker1
10.19.140.12   broker2.interop.example.com              broker2
10.19.140.13   broker3.interop.example.com              broker3
10.19.140.21   node1.interop.example.com                node1
10.19.140.22   node2.interop.example.com                node2
10.19.140.23   node3.interop.example.com                node3
#
# Identity Management (IdM) for RHEL:
#
10.19.140.100  admin1.interop.example.com                        admin1
10.19.140.101  idm-srv1.interop.example.com                idm-srv1
10.19.140.102  idm-srv2.interop.example.com                idm-srv2
```

Alternatively, the changes can be made on one host and then distributed to all OpenShift and IdM hosts, clients using a file transfer tool such as **scp**.

```
# scp -p /etc/hosts rhc1.interop.example.com:/etc/hosts
# scp -p /etc/hosts broker1.interop.example.com:/etc/hosts
# scp -p /etc/hosts broker2.interop.example.com:/etc/hosts
# scp -p /etc/hosts broker3.interop.example.com:/etc/hosts
# scp -p /etc/hosts node1.interop.example.com:/etc/hosts
# scp -p /etc/hosts node2.interop.example.com:/etc/hosts
# scp -p /etc/hosts node3.interop.example.com:/etc/hosts
# scp -p /etc/hosts admin1.interop.example.com:/etc/hosts
# scp -p /etc/hosts idm-srv1.interop.example.com:/etc/hosts
# scp -p /etc/hosts idm-srv2.interop.example.com:/etc/hosts
```



4.8 Register Hosts and Run Updates

Register each host using Red Hat Subscription Manager (RHSM) then update to the latest packages available.

1. Register the host and attach to subscriptions using the Customer Portal Red Hat Login.

```
# subscription-manager register
Username: username
Password: *****
The system has been registered with ID: f8b27857-9a02-43ee-97b9-2bd1d379f8c8

# subscription-manager list --available
+-----+
| Available Subscriptions |
+-----+

...output abbreviated...

Subscription Name: Red Hat Enterprise Linux Server, Premium (4 sockets)
(Unlimited guests)
SKU:                SKU1234
Pool ID:            1234567890123456789012345678901234
Quantity:           1
Service Level:      PREMIUM
Service Type:        L1-L3
Multi-Entitlement:   No
Ends:                01/01/2022
System Type:         Physical
```

2. Attach to the subscription using the Pool ID.

```
# subscription-manager attach --pool 1234567890123456789012345678901234
Successfully attached a subscription for: Red Hat Enterprise Linux Server,
Premium (4 sockets) (Unlimited guests)

Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status:       Subscribed
```

3. Update the host.

```
# yum -y update
rhel-6-server-rpms | 3.7 kB 00:00

... output abbreviated ...

Complete!
```

4. Reboot the host to activate the updates.

```
# reboot
```

This completes the staging of all hosts in the reference architecture infrastructure.



5 Deploying Identity Management in Red Hat Enterprise Linux

This section details the deployment tasks for configuring the IdM Server, Replica and IdM Admin client. The Replica server is deployed to increase the availability and resiliency of the services provided within IdM. Before proceeding with the IdM deployment tasks, each system must be configured as previously outlined in **Section 4 Staging the Infrastructure**:

- **4.2 Install Red Hat Enterprise Linux**
- **4.3 Configure Network**
- **4.4 Configure Domain Name System (DNS)**
- **4.5 Configure Time Service (NTP)**
- **4.6 Configure SELinux**
- **4.7 Update Hosts File**
- **4.8 Register Hosts and Run Updates**

Do not proceed until each of these steps has been fully completed.

5.1 Deploy IdM Server

Deploy the IdM server by configuring firewall ports, installing the required packages and configuring the IdM server. A set of user groups and accounts are also created for testing and verification purposes.

5.1.1 Configure Firewall Ports

On the IdM Server (**idm-srv1**) create a new chain (***ipa-server-chain***) and add the appropriate firewall rules for the ports required by IdM.

1. The default firewall service on Red Hat Enterprise Linux 7 uses **firewalld**. To avoid potential conflicts, stop and prevent the **iptables** (IPV4) and **ip6tables** (IPV6) services from running.

```
# systemctl stop iptables
# systemctl mask iptables
ln -s '/dev/null' '/etc/systemd/system/iptables.service'
# systemctl status iptables
iptables.service
  Loaded: masked (/dev/null)
  Active: inactive (dead)

Jun 20 19:06:55 idm-srv1.interop.example.com systemd[1]: Stopped IPv4
firewall with iptables.

# systemctl stop ip6tables
```



```
# systemctl mask ip6tables
ln -s '/dev/null' '/etc/systemd/system/ip6tables.service'
# systemctl status ip6tables
ip6tables.service
  Loaded: masked (/dev/null)
  Active: inactive (dead)

Jun 20 19:06:56 idm-srv1.interop.example.com systemd[1]: Stopped IPv6
firewall with ip6tables.
```

2. Start firewalld and enable it start on boot.

```
# systemctl start firewalld
# systemctl enable firewalld
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
  Active: active (running) since Fri 2014-06-20 15:00:05 EDT; 4h 7min ago
  Main PID: 678 (firewalld)
  CGroup: /system.slice/firewalld.service
          └─678 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Jun 20 19:07:34 idm-srv1.interop.example.com systemd[1]: Started firewalld -
dynamic firewall daemon.
```

3. Create a new chain (***ipa-server-chain***) and add the appropriate firewall rules for the ports required by IdM.

```
# firewall-cmd --permanent --direct --add-chain ipv4 filter ipa-server-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -m conntrack
--ctstate NEW -j ipa-server-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 80 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 80 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 443 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 389 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 636 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 88 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 464 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 88 --jump ACCEPT
```



```

success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 464 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 53 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 53 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 123 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 7389 --jump ACCEPT
success
# firewall-cmd --reload
success

```

Each of the ports are described in further detail in **Table 3.3.4 Network Ports**.

4. Verify the entries.

```

# firewall-cmd --permanent --direct --get-all-rules
ipv4 filter INPUT 0 -m conntrack --ctstate NEW -j ipa-server-chain
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 80 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 443 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 389 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 636 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 123 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 7389 --jump ACCEPT

```

5.1.2 Install Packages

On the IdM Server (**idm-srv1**) install the IPA server, DNS and DDNS packages.

```
# yum install ipa-server bind bind-dyndb-ldap
```

5.1.3 Install/configure IdM Server

Next, run the server installation script by specifying options to configure DNS (**--setup-dns**), Kerberos realm (**--realm**) and DNS forwarder (**--forwarder**). It is also recommended to not configure the NTP server (**--no-ntp**) when IdM is configured on a virtual machine.

```

# hostname
idm-srv1.interop.example.com

# ipa-server-install --setup-dns --realm=INTEROP.EXAMPLE.COM

```



```
--forwarder=10.19.143.247 --no-ntp
```

The installation script performs the following tasks:

- Configures a stand-alone CA (dogtag) for certificate management
- Creates/configures an instance of Directory Server (LDAP)
- Creates/configures a Kerberos Key Distribution Center (KDC)
- Configures Apache (httpd)
- Configures DNS (bind)
- Prevents the existing Network Time Protocol (NTP) configuration from being updated

During the installation, enter the following values as appropriate to your environment:

```
Existing BIND configuration detected, overwrite? [no]: yes
Server host name [idm-srv1.interop.example.com]:
Please confirm the domain name [interop.example.com]:
Directory Manager password: *****
Password (confirm): *****
IPA admin password: *****
Password (confirm): *****
Do you want to configure the reverse zone? [yes]:
Please specify the reverse zone name [140.16.10.in-addr.arpa.]:
```

The script provides a summary of the selections:

```
The IPA Master Server will be configured with:
Hostname: idm-srv1.interop.example.com
IP address: 10.19.140.101
Domain name: interop.example.com
Realm name: INTEROP.EXAMPLE.COM

BIND DNS server will be configured to serve IPA domain with:
Forwarders: 10.19.143.247
Reverse zone: 140.16.10.in-addr.arpa.

Continue to configure the system with these values? [no]: yes
```

Enter **Yes** to continue. Should any installation issues be encountered, the IdM server can be removed as follows:

```
# ipa-server-install --uninstall
```

and the installation restarted as previously described above.

Identity Management in Red Hat Enterprise Linux offers many other features and configuration options that are beyond the scope of this reference architecture. For more detail, please see the latest version of the [Red Hat Enterprise Linux 7- Linux Domain, Identity, Authentication and Policy Guide](#) on the Red Hat customer portal.



5.1.4 Verify IdM Server

Verify Kerberos functionality using the IdM admin account by obtaining a Kerberos ticket.

```
# klist
klist: No credentials cache found (ticket cache KEYRING:persistent:0:0)

# kinit admin
Password for admin@INTEROP.EXAMPLE.COM: ******

# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting          Expires              Service principal
06/21/2014 12:19:45    06/22/2014 12:19:41
krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM
```

View the IdM admin account details.

```
# ipa user-show admin
User login: admin
Last name: Administrator
Home directory: /home/admin
Login shell: /bin/bash
UID: 1550200000
GID: 1550200000
Account disabled: False
Password: True
Member of groups: admins, trust admins
Kerberos keys available: True
```

This confirms the base functionality of the IdM server is in place.

5.1.5 Configure IdM Groups and Users

Create groups and user accounts within IdM for the OpenShift administrators, developers and users.

```
# ipa group-add ose-admins --desc="OpenShift Administrators" --gid=889000001
-----
Added group "ose-admins"
-----
Group name: ose-admins
Description: OpenShift Administrators
GID: 889000001

# ipa group-add ose-developers --desc="OpenShift Developers" --gid=889000002
-----
Added group "ose-developers"
-----
Group name: ose-developers
```




```
Description: OpenShift Developers
GID: 889000002

# ipa group-add ose-users --desc="OpenShift Users" --gid=889000003
-----
Added group "ose-users"
-----
Group name: ose-users
Description: OpenShift Users
GID: 889000003

# ipa user-add ose-admin1 --first="OSE" --last="Admin 1"
--displayname="OpenShift Administrator 1" --homedir="/home/ose-admin1"
--shell="/bin/bash" --uid=889000001 --gidnumber=889000001 --password
Password: *****
Enter Password again to verify: *****
-----
Added user "ose-admin1"
-----
User login: ose-admin1
First name: OSE
Last name: Admin 1
Full name: OSE Admin 1
Display name: OpenShift Administrator 1
Initials: OA
Home directory: /home/ose-admin1
GECOS: OSE Admin 1
Login shell: /bin/bash
Kerberos principal: ose-admin1@INTEROP.EXAMPLE.COM
Email address: ose-admin1@interop.example.com
UID: 889000001
GID: 889000001
Password: True
Member of groups: ipausers
Kerberos keys available: True

# ipa user-add ose-dev1 --first="OSE" --last="Dev 1"
--displayname="OpenShift Developer 1" --homedir="/home/ose-dev1"
--shell="/bin/bash" --uid=889000002 --gidnumber=889000002 --password
Password: *****
Enter Password again to verify:
-----
Added user "ose-dev1"
-----
User login: ose-dev1
First name: OSE
Last name: Dev 1
Full name: OSE Dev 1
Display name: OpenShift Developer 1
Initials: OD
Home directory: /home/ose-dev1
GECOS: OSE Dev 1
Login shell: /bin/bash
Kerberos principal: ose-dev1@INTEROP.EXAMPLE.COM
Email address: ose-dev1@interop.example.com
```



```
UID: 889000002
GID: 889000002
Password: True
Member of groups: ipausers
Kerberos keys available: True

# ipa user-add ose-user1 --first="OSE" --last="User 1"
--displayname="OpenShift User 1" --homedir="/home/ose-user1"
--shell="/bin/bash" --uid=889000003 --gidnumber=889000003 --password
Password: *****
Enter Password again to verify:
-----
Added user "ose-user1"
-----
User login: ose-user1
First name: OSE
Last name: User 1
Full name: OSE User 1
Display name: OpenShift User 1
Initials: OU
Home directory: /home/ose-user1
GECOS: OSE User 1
Login shell: /bin/bash
Kerberos principal: ose-user1@INTEROP.EXAMPLE.COM
Email address: ose-user1@interop.example.com
UID: 889000003
GID: 889000003
Password: True
Member of groups: ipausers
Kerberos keys available: True
```

Note: The IdM groups must be created *before* user accounts can be created and added to them. User accounts and groups can be created on the IdM server itself (*idm-srv1*) or (*optionally*) from any IdM client previously configured with the **ipa-admintools**.

5.1.6 Verify IdM Groups and Users

Verify the details of the newly created IdM groups and users.

```
# ipa group-show ose-admins
Group name: ose-admins
Description: OpenShift Administrators
GID: 889000001

# ipa group-show ose-developers
Group name: ose-developers
Description: OpenShift Developers
GID: 889000002

# ipa group-show ose-users
Group name: ose-users
Description: OpenShift Users
GID: 889000003
```



```
# ipa user-show ose-admin1
User login: ose-admin1
First name: OSE
Last name: Admin 1
Home directory: /home/ose-admin1
Login shell: /bin/bash
Email address: ose-admin1@interop.example.com
UID: 889000001
GID: 889000001
Account disabled: False
Password: True
Member of groups: ipausers
Kerberos keys available: True

# ipa user-show ose-dev1
User login: ose-dev1
First name: OSE
Last name: Dev 1
Home directory: /home/ose-dev1
Login shell: /bin/bash
Email address: ose-dev1@interop.example.com
UID: 889000002
GID: 889000002
Account disabled: False
Password: True
Member of groups: ipausers
Kerberos keys available: True

# ipa user-show ose-user1
User login: ose-user1
First name: OSE
Last name: User 1
Home directory: /home/ose-user1
Login shell: /bin/bash
Email address: ose-user1@interop.example.com
UID: 889000003
GID: 889000003
Account disabled: False
Password: True
Member of groups: ipausers
Kerberos keys available: True
```

Note: In this reference architecture, user and group ID's have been explicitly set for demonstration purposes. In general it is preferable to let IdM manage the assignment of ID's to avoid potential conflicts. For more details, please see section **9.8. Managing Unique UID and GID Number Assignments**⁷ of the **Red Hat Enterprise Linux 7 - Linux Domain, Identity, Authentication and Policy Guide**.

⁷ [Linux Domain, Identity, Authentication and Policy Guide: Managing Unique UID and GID Attributes](#)



5.2 Deploy IdM Admin Client

The IdM Admin client is configured to allow administrators to perform common IdM server tasks such as user account management from an IdM client. The Admin client is configured similar to any IdM client, but is also configured with the **ipa-admintools** package. Red Hat Enterprise Linux 7 is demonstrated here but either Red Hat Enterprise Linux 6 or 7 can be deployed as an IdM Admin client.

5.2.1 Configure Firewall Ports

On the IdM Admin client (**admin1**) create a new chain (**ipa-client-chain**) and add the appropriate firewall rules for the ports required by IdM.

1. The default firewall service on Red Hat Enterprise Linux 7 uses **firewalld**. To avoid potential conflicts, stop and prevent the **iptables** (IPV4) and **ip6tables** (IPV6) services from running.

```
# systemctl stop iptables
# systemctl mask iptables
ln -s '/dev/null' '/etc/systemd/system/iptables.service'
# systemctl status iptables
iptables.service
  Loaded: masked (/dev/null)
  Active: inactive (dead)

Jun 20 19:12:35 admin1.interop.example.com systemd[1]: Stopped IPv4 firewall
with iptables.

# systemctl stop ip6tables
# systemctl mask ip6tables
ln -s '/dev/null' '/etc/systemd/system/ip6tables.service'
# systemctl status ip6tables
ip6tables.service
  Loaded: masked (/dev/null)
  Active: inactive (dead)

Jun 20 19:12:35 admin1.interop.example.com systemd[1]: Stopped IPv6 firewall
with ip6tables.
```

2. Start firewalld and enable it start on boot.

```
# systemctl start firewalld
# systemctl enable firewalld
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
  Active: active (running) since Fri 2014-06-20 15:00:14 EDT; 4h 12min ago
  Main PID: 637 (firewalld)
  CGroup: /system.slice/firewalld.service
          └─637 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Jun 20 19:12:51 admin1.interop.example.com systemd[1]: Started firewalld -
```



dynamic firewall daemon.

3. Create a new chain (*ipa-client-chain*) and add the appropriate firewall rules for the ports required by IdM.

```
# firewall-cmd --permanent --direct --add-chain ipv4 filter ipa-client-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -m conntrack
--ctstate NEW -j ipa-server-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 80 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 80 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 443 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 389 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 636 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 88 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 464 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto udp --destination-port 88 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto udp --destination-port 464 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto tcp --destination-port 53 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto udp --destination-port 53 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-client-chain 0
--proto udp --destination-port 123 --jump ACCEPT
success
# firewall-cmd --reload
success
```

Each of the ports are described in further detail in **Table 3.3.4 Network Ports**.

4. Verify the entries.

```
# firewall-cmd --permanent --direct --get-all-rules
ipv4 filter INPUT 0 -m conntrack --ctstate NEW -j ipa-client-chain
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 80 --jump ACCEPT
```



```
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 443 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 389 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 636 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto udp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto udp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto tcp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto udp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-client-chain 0 --proto udp --destination-port 123 --jump ACCEPT
```

5.2.2 Install Packages

On the IdM Admin client (**admin1**) install the IPA client package.

```
# yum install ipa-client
```

5.2.3 Configure DNS

Edit the DNS resolver file (*/etc/resolv.conf*) to include the name of the IdM server and replica.

```
domain interop.example.com
search interop.example.com
nameserver 10.19.140.101
nameserver 10.19.140.102
nameserver 10.19.143.247
```

5.2.4 Install/Configure IdM Client

Configure the IdM admin client to automatically update DNS on the the IdM server with any IP address changes (**--enable-dns-updates**), trust DNS SSH finger prints (**--ssh-trust-dns**) and create home directories (**--mkhomedir**) on first login.

```
# hostname
admin1.interop.example.com

# ipa-client-install --enable-dns-updates --ssh-trust-dns --mkhomedir
Discovery was successful!
Hostname: admin1.interop.example.com
Realm: INTEROP.EXAMPLE.COM
DNS Domain: interop.example.com
IPA Server: idm-srv1.interop.example.com
BaseDN: dc=interop,dc=example,dc=com

Continue to configure the system with these values? [no]: yes
User authorized to enroll computers: admin
Synchronizing time with KDC...
Unable to sync time with IPA NTP server, assuming the time is in sync.
Please check that 123 UDP port is opened.
Password for admin@INTEROP.EXAMPLE.COM: *****
Successfully retrieved CA cert
    Subject:      CN=Certificate Authority,O=INTEROP.EXAMPLE.COM
```



```
Issuer:      CN=Certificate Authority,O=INTEROP.EXAMPLE.COM
Valid From:  Tue Jun 17 14:25:45 2014 UTC
Valid Until: Sat Jun 17 14:25:45 2034 UTC
```

```
Enrolled in IPA realm INTEROP.EXAMPLE.COM
Created /etc/ipa/default.conf
New SSSD config will be created
Configured /etc/sss/sss.conf
Configured /etc/krb5.conf for IPA realm INTEROP.EXAMPLE.COM
trying https://idm-srv1.interop.example.com/ipa/xml
Forwarding 'ping' to server 'https://idm-srv1.interop.example.com/ipa/xml'
Forwarding 'env' to server 'https://idm-srv1.interop.example.com/ipa/xml'
Hostname (admin1.interop.example.com) not found in DNS
DNS server record set to: admin1.interop.example.com -> 10.19.140.100
Adding SSH public key from /etc/ssh/ssh_host_rsa_key.pub
Adding SSH public key from /etc/ssh/ssh_host_ecdsa_key.pub
Forwarding 'host_mod' to server 'https://idm-srv1.interop.example.com/ipa/xml'
SSSD enabled
Configured /etc/openldap/ldap.conf
NTP enabled
Configured /etc/ssh/ssh_config
Configured /etc/ssh/sshd_config
Client configuration complete.
```

Should any issues be encountered during the installation, the IdM client can be removed as follows:

```
# ipa-client-install --uninstall
```

and the installation restarted as previously described above.

Should any issues be encountered, consult the client installation chapter of the [Red Hat Enterprise Linux 7 - Linux Domain, Identity, Authentication and Policy Guide](#) on the Red Hat customer portal.

5.2.5 Verify IdM Admin Client

Confirm the client is able to correctly lookup user accounts within IdM.

```
# id
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

# id admin
uid=1550200000(admin) gid=1550200000(admins) groups=1550200000(admins)

# id ose-admin1
uid=889000001(ose-admin1) gid=889000001 groups=889000001

# id ose-dev1
uid=889000002(ose-dev1) gid=889000002 groups=889000002

# id ose-user1
```



```
uid=889000003(ose-user1) gid=889000003 groups=889000003
```

5.2.6 Install Admin Tools (optional)

On the IdM admin system (**admin1**) install and configure the IPA Admin Tools package.

```
# yum install ipa-admintools
```

The **IPA Admin Tools** package permits any IdM client to manage user accounts, groups. Installation is optional but recommended for any IdM client performing IdM server administration tasks.



5.3 Deploy IdM Replica Server

This reference architecture uses an IdM replica server to provide highly availability of IdM services to all IdM clients. In the sections that follow, the IdM replica server (**idm-srv2**) is configured and verified.

Similar to the IdM server (**idm-srv1**), the firewall ports are configured and the required packages installed on the IdM replica server. Configuring a replica starts by creating a replica information file on the IdM server, transferring it to the IdM replica server and then running the replica installation with it. Once the replica is created, it is functionally identical to the original IdM server.

For environments interested in deploying additional IdM replicas, the same steps can be applied to each additional replica server. IdM currently supports a maximum of twenty replica servers.

5.3.1 Configure Firewall Ports

On the IdM **Replica** server (**idm-srv2**) create a new chain (**ipa-server-chain**) and add the appropriate firewall rules for the ports required by IdM.

1. The default firewall service on Red Hat Enterprise Linux 7 uses **firewalld**. To avoid potential conflicts, stop and prevent the **iptables** (IPV4) and **ip6tables** (IPV6) services from running.

```
# systemctl stop iptables
# systemctl mask iptables
ln -s '/dev/null' '/etc/systemd/system/iptables.service'
# systemctl status iptables
iptables.service
  Loaded: masked (/dev/null)
  Active: inactive (dead)

Jun 20 19:09:26 idm-srv2.interop.example.com systemd[1]: Stopped IPv4
firewall with iptables.

# systemctl stop ip6tables
# systemctl mask ip6tables
ln -s '/dev/null' '/etc/systemd/system/ip6tables.service'
# systemctl status ip6tables
ip6tables.service
  Loaded: masked (/dev/null)
  Active: inactive (dead)

Jun 20 19:09:27 idm-srv2.interop.example.com systemd[1]: Stopped IPv6
firewall with ip6tables.
```

2. Start **firewalld** and enable it start on boot.

```
# systemctl start firewalld
# systemctl enable firewalld
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
```



```
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
Active: active (running) since Fri 2014-06-20 15:00:11 EDT; 4h 9min ago
Main PID: 681 (firewalld)
CGroup: /system.slice/firewalld.service
        └─681 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

Jun 20 19:09:45 idm-srv2.interop.example.com systemd[1]: Started firewalld -
dynamic firewall daemon.
```

3. Create a new chain (***ipa-server-chain***) and add the appropriate firewall rules for the ports required by IdM.

```
# firewall-cmd --permanent --direct --add-chain ipv4 filter ipa-server-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -m conntrack
--ctstate NEW -j ipa-server-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 80 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 80 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 443 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 389 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 636 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 88 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 464 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 88 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 464 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 53 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 53 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto udp --destination-port 123 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-server-chain 0
--proto tcp --destination-port 7389 --jump ACCEPT
```



```
success
# firewall-cmd --reload
success
```

Each of the ports are described in further detail in **Table 3.3.4 Network Ports**.

4. Verify the entries.

```
# firewall-cmd --permanent --direct --get-all-rules
ipv4 filter INPUT 0 -m conntrack --ctstate NEW -j ipa-server-chain
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 80 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 443 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 389 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 636 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 123 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 7389 --jump ACCEPT
```

5.3.2 Install Packages

On the IdM **Replica** server (**idm-srv2**) install the IPA server, DNS and DDNS packages.

```
# yum install ipa-server bind bind-dyndb-ldap
```

5.3.3 Run Replica Prepare

On the IdM **Server** (**idm-srv1**) run the replica prepare utility by specifying the FQDN and IP address of the **Replica** server. The **ipa-replica-prepare** utility creates a replica information file that is used to initially synchronize the replica to the master.

```
# ipa-replica-prepare idm-srv2.interop.example.com --ip-address
10.19.140.102
Directory Manager (existing master) password: *****

Preparing replica for idm-srv2.interop.example.com from idm-
srv1.interop.example.com
Creating SSL certificate for the Directory Server
Creating SSL certificate for the dogtag Directory Server
Saving dogtag Directory Server port
Creating SSL certificate for the Web Server
Exporting RA certificate
Copying additional files
Finalizing configuration
Packaging replica information into /var/lib/ipa/replica-info-idm-
srv2.interop.example.com.gpg
Adding DNS records for idm-srv2.interop.example.com
Using reverse zone 140.19.10.in-addr.arpa.
The ipa-replica-prepare command was successful
```



5.3.4 Copy Replica Information File

On the IdM **Server** (**idm-srv1**), copy the replica information file to the IdM **Replica** server (**idm-srv2**).

```
# scp /var/lib/ipa/replica-info-idm-srv2.interop.example.com.gpg root@idm-srv2:/var/lib/ipa/
```

5.3.5 Run Replica Installation

Create the replica by running **ipa-replica-install** on the IdM **Replica** server (**idm-srv2**). Install the replica by specifying the options to configure a Certificate Authority (**--setup-ca**), DNS (**--setup-dns**, **--forwarder**, **--reverse**) and the location of the replica information file.

```
# hostname
idm-srv2.interop.example.com

# ipa-replica-install --setup-ca --setup-dns --forwarder=10.19.143.247 \
  --reverse-zone=140.19.10.in-addr.arpa. /var/lib/ipa/replica-info-idm-srv2 \
  .interop.example.com.gpg
Directory Manager (existing master) password: *****

Run connection check to master
Check connection from replica to remote master 'idm-srv1.interop.example.com':
  Directory Service: Unsecure port (389): OK
  Directory Service: Secure port (636): OK
  Kerberos KDC: TCP (88): OK
  Kerberos Kpasswd: TCP (464): OK
  HTTP Server: Unsecure port (80): OK
  HTTP Server: Secure port (443): OK

The following list of ports use UDP protocol and would need to be
checked manually:
  Kerberos KDC: UDP (88): SKIPPED
  Kerberos Kpasswd: UDP (464): SKIPPED

Connection from replica to master is OK.
Start listening on required ports for remote master check
Get credentials to log in to remote master
admin@INTEROP.EXAMPLE.COM password: *****

Check SSH connection to remote master
Execute check on remote master
Check connection from master to remote replica 'idm-srv2.interop.example.com':
  Directory Service: Unsecure port (389): OK
  Directory Service: Secure port (636): OK
  Kerberos KDC: TCP (88): OK
  Kerberos KDC: UDP (88): OK
  Kerberos Kpasswd: TCP (464): OK
  Kerberos Kpasswd: UDP (464): OK
  HTTP Server: Unsecure port (80): OK
  HTTP Server: Secure port (443): OK
```



Connection from master to replica is OK.

Connection check OK

Configuring NTP daemon (ntpd)

[1/4]: stopping ntpd

[2/4]: writing configuration

[3/4]: configuring ntpd to start on boot

[4/4]: starting ntpd

Done configuring NTP daemon (ntpd).

...output abbreviated...

Applying LDAP updates

Restarting the directory server

Restarting the KDC

Restarting the certificate server

Using reverse zone 140.19.10.in-addr.arpa.

Configuring DNS (named)

[1/9]: adding NS record to the zone

[2/9]: setting up reverse zone

[3/9]: setting up our own record

[4/9]: setting up CA record

[5/9]: setting up kerberos principal

[6/9]: setting up named.conf

[7/9]: restarting named

[8/9]: configuring named to start on boot

[9/9]: changing resolv.conf to point to ourselves

Done configuring DNS (named).

Global DNS configuration in LDAP server is empty

You can use 'dnsconfig-mod' command to set global DNS options that would override settings in local named.conf files

Restarting the web server

5.3.6 Verify DNS

Verify that the correct DNS service entries (*aka - `_srv_` records*) were created so that IdM clients can discover both IdM servers.

IdM Server (idm-srv1)

```
# hostname
idm-srv1.interop.example.com
# DOMAIN=interop.example.com
# NAMESERVER=idm-srv1
# for i in _ldap._tcp _kerberos._tcp _kerberos._udp _kerberos-master._tcp
_kerberos-master._udp _ntp._udp; do echo ""; dig @$${NAMESERVER} ${i}.$
{DOMAIN} srv +nocmd +noquestion +nocomments +nostats +noaa +noadditional
+noauthority; done | egrep -v "^;" | egrep _
_ldap._tcp.interop.example.com. 86400 IN SRV 0 100 389 idm-
srv1.interop.example.com.
_ldap._tcp.interop.example.com. 86400 IN SRV 0 100 389 idm-
srv2.interop.example.com.
_kerberos._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
```



```

srv2.interop.example.com.
_kerberos._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_kerberos-master._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_kerberos-master._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos-master._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_kerberos-master._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_ntp._udp.interop.example.com. 86400 IN SRV 0 100 123 idm-
srv2.interop.example.com.

```

IdM Replica (idm-srv2)

```

# hostname
idm-srv2.interop.example.com
# DOMAIN=interop.example.com
# NAMESERVER=idm-srv2
# for i in _ldap._tcp._kerberos._tcp._kerberos._udp._kerberos-master._tcp
_kerberos-master._udp._ntp._udp; do echo ""; dig @$${NAMESERVER} ${i}.$
{DOMAIN} srv +nocmd +noquestion +nocomments +nostats +noaa +noadditional
+noauthority; done | egrep -v "^;" | egrep _
_ldap._tcp.interop.example.com. 86400 IN SRV 0 100 389 idm-
srv1.interop.example.com.
_ldap._tcp.interop.example.com. 86400 IN SRV 0 100 389 idm-
srv2.interop.example.com.
_kerberos._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_kerberos._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_kerberos._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos-master._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_kerberos-master._tcp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos-master._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv1.interop.example.com.
_kerberos-master._udp.interop.example.com. 86400 IN SRV 0 100 88 idm-
srv2.interop.example.com.
_ntp._udp.interop.example.com. 86400 IN SRV 0 100 123 idm-
srv2.interop.example.com.

```

The output should be identical on both IdM servers.



5.3.7 Verify Replication

Verify the replication agreement by running the **ipa-replica-manage** utility from either of the IdM servers (**idm-srv1**, **idm-srv2**).

```
# kinit admin
Password for admin@INTEROP.EXAMPLE.COM:

# ipa-replica-manage list
idm-srv1.interop.example.com: master
idm-srv2.interop.example.com: master
```

5.3.8 Test Replication

Replication can be tested by first rebooting or shutting down either of the IdM servers, testing Kerberos access to the realm and then verifying access to one or more IdM accounts. In the example that follows, the IdM Server (**idm-srv1**) is shutdown and access from the IdM admin client (**admin1**) verified using the IdM **admin** account.

1. Shutdown IdM Server (**idm-srv1**).

```
# hostname
idm-srv1.interop.example.com

# shutdown now
Broadcast message from root@idm-srv1.interop.example.com
(/dev/pts/0) at 16:15 ...

The system is going down for maintenance NOW!
```

2. Destroy any existing Kerberos tickets (**admin1**).

```
# hostname
admin1.interop.example.com

# kdestroy
# klist
klist: No credentials cache found (ticket cache KEYRING:persistent:0:0)
```

3. Clear out the SSSD cache (**admin1**).

```
# systemctl restart sssd
```

4. Obtain new Kerberos ticket (**admin1**).

```
# kinit admin
Password for admin@INTEROP.EXAMPLE.COM: *****
# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting          Expires              Service principal
06/21/2014 12:24:41    06/22/2014 12:24:37    krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM
```




5. Verify IdM lookups (**admin1**).

```
# id admin
uid=1550200000(admin) gid=1550200000(admins) groups=1550200000(admins)

# ipa user-show admin
User login: admin
Last name: Administrator
Home directory: /home/admin
Login shell: /bin/bash
UID: 1550200000
GID: 1550200000
Account disabled: False
Password: True
Member of groups: admins, trust admins
Kerberos keys available: True

# ipa group-show admins
Group name: admins
Description: Account administrators group
GID: 1550200000
Member users: admin

# id ose-admin1
uid=8890000001(ose-admin1) gid=8890000001 groups=8890000001

# ipa user-show ose-admin1
User login: ose-admin1
First name: OSE
Last name: Admin 1
Home directory: /home/ose-admin1
Login shell: /bin/bash
Email address: ose-admin1@interop.example.com
UID: 8890000001
GID: 8890000001
Account disabled: False
Password: True
Member of groups: ipausers
Kerberos keys available: True

# ipa group-show ose-admins
Group name: ose-admins
Description: OpenShift Administrators
GID: 8890000001
```

The completes the deployment, verification of the IdM Server, Replica and the admin client.

Should any replication issues be encountered, consult the replication chapter of the [Red Hat Enterprise Linux 7 - Linux Domain, Identity, Authentication and Policy Guide](#) on the Red Hat customer portal.

Restart the IdM Server (**idm-srv1**) before continuing on to the next section.



6 Deploying OpenShift Enterprise

This section details the deployment tasks for configuring the OpenShift Enterprise Broker hosts, Node hosts and RHC client. Before proceeding with the OSE deployment tasks, each host must be configured as previously outlined in **Section 4 Staging the Infrastructure**:

- **4.2 Install Red Hat Enterprise Linux**
- **4.3 Configure Network**
- **4.4 Configure Domain Name System (DNS)**
- **4.5 Configure Time Service (NTP)**
- **4.6 Configure SELinux**
- **4.7 Update Hosts File**
- **4.8 Register Hosts and Run Updates**

Do not proceed until each of these steps has been fully completed.

6.1 Configure Deployment Script

A variety of methods are available to deploy OpenShift Enterprise. OpenShift provides a comprehensive installation script called *openshift.sh*⁸. This reference architecture utilizes a front-end to the installation script called *deploy-ose.sh*⁹. A copy is also provided within **Appendix D: OpenShift Deployment Script**. The *deploy-ose.sh* script simplifies the deployment of broker and node hosts and can be easily modified to scale-out (or scale-in) the number of brokers and nodes required by your environment. Configure the *deploy-ose.sh* script as follows:

1. Save a copy of the script to to a convenient location - e.g. */var/tmp/deploy-ose.sh* on the first broker (**broker1**).
2. Edit the *deploy-ose.sh* script and adjust the following variables according to the number of brokers and nodes in your environment.

```
DOMAIN="interop.example.com"
BROKER1="broker1.${DOMAIN}"
BROKER2="broker2.${DOMAIN}"
BROKER3="broker3.${DOMAIN}"
NODE1="node1.${DOMAIN}"
NODE2="node2.${DOMAIN}"
NODE3="node3.${DOMAIN}"
```

⁸ <https://raw.githubusercontent.com/openshift/openshift-extras/enterprise-2.2/enterprise/install-scripts/generic/openshift.sh>

⁹ <https://raw.githubusercontent.com/mheslin/ose-idm-supplemental/master/deploy-ose-2.2.sh>



Note: If more or less brokers, nodes are required then adjust accordingly. For highly-available (HA) **MongoDB** deployments, an odd number of replica servers is required. Since the first broker host (**broker1**) is configured as the primary **MongoDB** server it is referenced from here on out as the *Primary* broker host.

- Configure the IP address of the NAMED server (**idm-srv1**) - IdM is used for DNS lookups and registration:

```
export CONF_NAMED_IP_ADDRESS="10.19.140.101"
```

- Set **BROKER_HOSTNAME** to the load balancer virtual server host name in use for broker hosts - e.g. **broker.example.com**. If no load balancer is in use, set this value to any broker host:

```
export BROKER_HOSTNAME="broker.${DOMAIN}"
```

Note: This reference architecture uses round-robin DNS configured on the IdM servers to demonstrate simple broker request balancing. Round-robin DNS is not a true load balancer and is used here for demonstration purposes only. For more details, see section **6.4.5 Configure Broker DNS Request Balancing (optional)**.

- Configure the **ActiveMQ** and **MongoDB** replicants to match the number of brokers, nodes being deployed:

```
export CONF_ACTIVEMQ_REPLICANTS="${BROKER1}, ${BROKER2}, ${BROKER3}"
export CONF_DATASTORE_REPLICANTS="${BROKER1}:27017, ${BROKER2}:27017, \
${BROKER3}:27017"
```

- Set the username and password pairs for the following components to match the policies of the local environment:

```
export CONF_OPENSIFT_USER1="user1"
export CONF_OPENSIFT_PASSWORD1="password"
export CONF_MONGODB_BROKER_USER="openshift"
export CONF_MONGODB_BROKER_PASSWORD="mongopass"
export CONF_MONGODB_ADMIN_USER="admin1"
export CONF_MONGODB_ADMIN_PASSWORD="mongopass"
export CONF_MONGODB_REPLSET="ose"
export CONF_MONGODB_KEY="OSEnterprise"
export CONF_MCOLLECTIVE_USER="mcollective"
export CONF_MCOLLECTIVE_PASSWORD="mcollective"
export CONF_ACTIVEMQ_ADMIN_PASSWORD="password"
export CONF_ACTIVEMQ_AMQ_USER_PASSWORD="password"
```



Each of these variables is highlighted and described within the *deploy-ose.sh* script. All parameters prefixed by **CONF_** are passed through to the OpenShift installation script - *openshift.sh*. Consult the installation script for additional details.

3. Save the file and configure the correct file permissions.

```
# chmod 755 /var/tmp/deploy-ose.sh
```

4. Copy the script from **broker1** to each of the remaining brokers (**broker2**, **broker3**) and all nodes (**node1**, **node2**, **node3**).

```
# scp -p /var/tmp/deploy-ose.sh broker2:/var/tmp/deploy-ose.sh
# scp -p /var/tmp/deploy-ose.sh broker3:/var/tmp/deploy-ose.sh
# scp -p /var/tmp/deploy-ose.sh node1:/var/tmp/deploy-ose.sh
# scp -p /var/tmp/deploy-ose.sh node2:/var/tmp/deploy-ose.sh
# scp -p /var/tmp/deploy-ose.sh node3:/var/tmp/deploy-ose.sh
```

5. Verify the script transfers and file permissions are configured correctly.

```
# ssh broker2 ls -l /var/tmp/deploy-ose.sh
-rwxr-xr-x. 1 root root 6140 Jul 10 10:50 /var/tmp/deploy-ose.sh
# ssh broker3 ls -l /var/tmp/deploy-ose.sh
-rwxr-xr-x. 1 root root 6140 Jul 10 10:50 /var/tmp/deploy-ose.sh
# ssh node1 ls -l /var/tmp/deploy-ose.sh
-rwxr-xr-x. 1 root root 6140 Jul 10 10:50 /var/tmp/deploy-ose.sh
# ssh node2 ls -l /var/tmp/deploy-ose.sh
-rwxr-xr-x. 1 root root 6140 Jul 10 10:50 /var/tmp/deploy-ose.sh
# ssh node3 ls -l /var/tmp/deploy-ose.sh
-rwxr-xr-x. 1 root root 6140 Jul 10 10:50 /var/tmp/deploy-ose.sh
```



6.2 Deploy Brokers

This section details how to configure and deploy the OpenShift Enterprise broker hosts.

6.2.1 Configure Entitlements

Table 6.2.1: Broker Host Channels describes the channels required for deploying OpenShift Enterprise broker hosts:

Channel Name	Description	Required?	Subscription Requirement
Red Hat OpenShift Enterprise 2.2 Infrastructure - x86_64	Base channel for OpenShift Enterprise 2.2 broker hosts	Yes	“OpenShift Enterprise Broker Infrastructure”
Red Hat OpenShift Enterprise 2.2 Client Tools - x86_64	Provides access to OpenShift Enterprise 2.2 Client Tools	No	“OpenShift Enterprise”
Red Hat Software Collections 1	Provides access to programming languages, databases and related packages	Yes	“OpenShift Enterprise”

Table 6.2.1: Broker Host Channels

Each broker host must be registered to these channels through either Red Hat Subscription Manager (RHSM) or Red Hat Network (RHN) Classic. This reference architecture utilizes Red Hat Subscription Manager.

Perform the following steps on on each broker (**broker1**, **broker2**, **broker3**).

1. Register the host (*if not already registered*).

```
# subscription-manager register
Username: username
Password: *****
The system has been registered with ID: e2f299d1-4daf-4cd4-9101-88460b4a66ca
```

2. Identify the OpenShift Enterprise subscription Pool ID.

```
# subscription-manager list --available
+-----+
Available Subscriptions
+-----+

...output abbreviated...

Subscription Name: OpenShift Enterprise Broker Infrastructure
SKU:                SKU1234
Pool ID:            1234567890123456789012345678901234
Quantity:          1
```



```
Service Level: Layered
Service Type: L1-L3
Multi-Entitlement: No
Ends: 01/01/2022
System Type: Physical
```

3. Attach to the subscriptions using the Pool IDs.

```
# subscription-manager attach --pool 1234567890123456789012345678901234
Successfully attached a subscription for: OpenShift Subscription
1 local certificate has been deleted.
```

4. Enable the **Red Hat OpenShift Enterprise 2.2 Infrastructure** channel.

```
# subscription-manager repos --enable rhel-6-server-ose-2.2-infra-rpms
Repo rhel-6-server-ose-2.2-infra-rpms is enabled for this system.
```

5. Verify the enabled channel is available.

```
# yum repolist
Loaded plugins: priorities, security, subscription-manager
This system is receiving updates from Red Hat Subscription Management.

...output abbreviated...

repo id                                repo name
rhel-6-server-ose-2.2-infra-rpms      Red Hat OpenShift Enterprise 2.2
Infrastructure (RPMs)
rhel-6-server-ose-2.2-rhc-rpms        Red Hat OpenShift Enterprise 2.2 Client
Tools (RPMs)
rhel-6-server-rpms                    Red Hat Enterprise Linux 6 Server (RPMs)
rhel-server-rhsc1-6-rpms              Red Hat Software Collections RPMs for Red
Hat Enterprise Linux 6 Server
```

Consult the OpenShift Enterprise 2 - Deployment Guide¹⁰ for details on configuring entitlements with Red Hat Network (RHN) Classic.

6.2.2 Configure Yum

The packages required for running OpenShift Enterprise are available from Red Hat Network (RHN). Third-party RPM repositories and even other products provided by Red Hat can create conflicts with OpenShift Enterprise during initial deployment or later when applying updates. To avoid potential issues, the **oo-admin-yum-validator** tool can be used to report on any problems and provide recommendations. By default, the tool halts so that you can review each set of proposed changes. You then have the option to apply the changes manually, or let the tool attempt to fix the issues that have been found. This process may require you to run the tool several times. You also have the option of having the tool both report all found issues, and attempt to fix all issues.

Use the **oo-admin-yum-validator** tool to configure yum and prepare to install the broker host components. Perform these steps on each node host (**broker1**, **broker2**, **broker3**) as follows:

¹⁰ [OpenShift Enterprise 2 - Deployment Guide](#)



1. Install the latest **openshift-enterprise-release** package.

```
# yum install openshift-enterprise-release
Loaded plugins: priorities, product-id, security, subscription-manager
Setting up Install Process

...output abbreviated...

Transaction Test Succeeded
Running Transaction
  Updating   : openshift-enterprise-yum-validator-2.2.1-1.el6op.noarch 1/3
  Installing : openshift-enterprise-release-2.2.1-1.el6op.noarch 2/3
  Cleanup    : openshift-enterprise-yum-validator-2.1.9-1.el6op.noarch 3/3
  Verifying  : openshift-enterprise-release-2.2.1-1.el6op.noarch 1/3
  Verifying  : openshift-enterprise-yum-validator-2.2.1-1.el6op.noarch 2/3
  Verifying  : openshift-enterprise-yum-validator-2.1.9-1.el6op.noarch 3/3

Installed:
  openshift-enterprise-release.noarch 0:2.2.1-1.el6op

Dependency Updated:
  openshift-enterprise-yum-validator.noarch 0:2.2.1-1.el6op

Complete!
```

2. Run the **oo-admin-yum-validator** command with the **-o** option for version 2.1 and the **-r** option for the *broker* role. This reports the first detected set of problems, provides a set of proposed changes, and halts.

```
# oo-admin-yum-validator -o 2.1 -r broker
```

Alternatively, use the **--report-all** option to report all detected problems.

```
# oo-admin-yum-validator -o 2.1 -r broker --report-all
```

3. After reviewing the reported problems and their proposed changes, either fix them manually or let the tool attempt to fix the first set of problems using the same command with the **--fix** option. This may require several repeats of steps 2 and 3.

```
# oo-admin-yum-validator -o 2.1 -r broker --fix
```

Alternatively, use the **--fix-all** option to allow the tool to attempt to fix all of the problems that are found:

```
# oo-admin-yum-validator -o 2.1 -r broker --fix-all
```

4. Repeat steps 2 and 3 until the **oo-admin-yum-validator** command displays the following message:

```
No problems could be detected!
```

For additional details on configuring yum on broker hosts, consult the OpenShift Enterprise 2 - Deployment Guide¹¹.

¹¹ [OpenShift Enterprise 2 - Deployment Guide: Configuring Yum on Broker Hosts](#)



6.2.3 Install OpenShift

Install OpenShift Enterprise on the *broker* hosts by running the deployment script on each secondary broker (**broker2**, **broker3**), then the primary broker (**broker1**) as follows.

Secondary Brokers (broker2, broker3)

```
# /var/tmp/deploy-ose.sh secondary
Deploying OpenShift Enterprise in a distributed environment

...Installation type set to *secondary*...

*** Control-C now if this is not a *Secondary* Broker or the wrong host ***

...Continuing with installation...

...Downloading openshift.sh script...

% Total    % Received % Xferd  Average Speed   Time    Time     Time      Current
           Dload  Upload   Total   Spent    Left     Speed
100 131k 100 131k  0     0  181k    0  --:--:--  --:--:--  --:--:--  483k

* Script download complete *

...Beginning installation - Wed Nov 26 19:50:13 EST 2014 ...

+ environment=sh
+ case "$environment" in
+ parse_cmdline
+ parse_args
+ lokkit=lokkit
+ declare -A passwords
+ PASSWORDS_TO_DISPLAY=false
+ RESTART_NEEDED=false
+ RESTART_COMPLETED=false

...output abbreviated...

+ echo 'OpenShift: Completed restarting services.'
OpenShift: Completed restarting services.
+ true
+ display_passwords
+ set +x
ActiveMQ admin password: password
MongoDB admin user: admin1 password: mongopass
ActiveMQ amq user user: admin1 password: password
routing plugin user: routinginfo pass: SoIP57HgdDeivx7PI8cSpDb7ls
MongoDB key user: routinginfo: OSEnterprise
MongoDB broker user: openshift password: mongopass
OpenShift user1: user1 password1: password
MCollective user: mcollective password: mcollective
+ :

* Installation completed - Wed Nov 26 19:54:42 EST 2014 *
```



Primary Broker (broker1)

```
# /var/tmp/deploy-ose.sh primary
Deploying OpenShift Enterprise in a distributed environment

...Installation type set to *primary*...

*** Control-C now if this is not the *Primary* Broker or the wrong host ***

...Continuing with installation...

...Downloading openshift.sh script...

% Total    % Received % Xferd  Average Speed   Time    Time     Time      Current
           Dload    Upload   Total   Spent    Left     Speed
100 131k 100 131k  0      0  212k    0  --:--:-- --:--:-- --:--:--   507k

* Script download complete *

...Beginning installation - Wed Nov 26 19:56:25 EST 2014 ...

+ environment=sh
+ case "$environment" in
+ parse_cmdline
+ parse_args
+ declare -A passwords
+ PASSWORDS_TO_DISPLAY=false
+ RESTART_NEEDED=false
+ RESTART_COMPLETED=false

...output abbreviated...

+ echo 'OpenShift: Completed restarting services.'
OpenShift: Completed restarting services.
+ true
+ display_passwords
+ set +x
ActiveMQ admin password: password
MongoDB admin user: admin1 password: mongopass
ActiveMQ amq user user: admin1 password: password
routing plugin user: routinginfo pass: vrVKn8m6XhsI4fs8EjiLDbEksw
MongoDB key user: routinginfo: OSEnterprise
MongoDB broker user: openshift password: mongopass
OpenShift user1: user1 password1: password
MCollective user: mcollective password: mcollective
+ :

* Installation completed - Wed Nov 26 20:01:47 EST 2014 *
```

Note: Prior to OpenShift Enterprise 2.2, when the primary broker was deployed before the secondaries the **MongoDB** replica set would not form properly. This behavior has since been corrected but it is still suggested to deploy all secondaries before the primary broker.



6.2.3.1 Recovering from Installation Failures

In the rare event that any **Broker** host is not able to fully complete the installation, the following steps can be used to cleanup the deployment.

1. Remove ActiveMQ packages.

```
# yum remove activemq
```

2. Remove MongoDB packages and on-disk configuration files.

```
# yum remove mongodb-server mongodb
# rm -rf /var/lib/mongodb          # Data files
# rm -f /etc/mongodb.*            # Config and keyfiles
```

Note: If the on-disk configuration files are not removed then the Mongo replica set may not form properly

3. Remove Broker packages.

```
# yum remove openshift-origin-broker openshift-origin-broker-util rubygem-
openshift-origin-msg-broker-mcollective ruby193-mcollective-client rubygem-
openshift-origin-auth-remote-user rubygem-openshift-origin-dns-nsupdate
openshift-origin-console rubygem-openshift-origin-admin-console
```

4. Remove RHC client (*installed by default on all broker hosts*).

```
# yum remove rhc
```

Once the above steps have been completed the Broker deployment can be re-run as previously described.

6.2.4 Re-configure Firewall Ports (optional)

Earlier versions of the *openshift.sh* installation script configure the firewall using the **lokkit** utility. For enterprise deployments requiring a greater degree of control and flexibility, change the firewall configuration to use **iptables**. Perform these steps on each OpenShift Enterprise broker host (**broker1**, **broker2**, **broker3**):

1. Disable the current **lokkit** firewall configuration.

```
# lokkit --disabled
```

2. Save the existing firewall configuration and create a new, default configuration.

```
# mv /etc/sysconfig/iptables{,.orig}
# service iptables restart
```

3. Create a new firewall chain (**ose-broker-chain**) and add the appropriate rules for the ports required by OpenShift Enterprise brokers.



```
# iptables --list --line-numbers --numeric --verbose
Chain INPUT (policy ACCEPT 7 packets, 904 bytes)
num  pkts bytes target    prot opt in     out     source    destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source    destination

Chain OUTPUT (policy ACCEPT 3 packets, 668 bytes)
num  pkts bytes target    prot opt in     out     source    destination

# iptables --new-chain ose-broker-chain
# iptables --insert INPUT --jump ose-broker-chain
# iptables --append INPUT --match state --state RELATED,ESTABLISHED --jump
ACCEPT
# iptables --append INPUT --protocol icmp --jump ACCEPT
# iptables --append INPUT --in-interface lo --jump ACCEPT
# iptables --append INPUT --jump REJECT --reject-with icmp-host-prohibited
# iptables --append FORWARD --jump REJECT --reject-with icmp-host-prohibited

# iptables --append ose-broker-chain --match state --state NEW --proto tcp
--destination-port 22 --jump ACCEPT
# iptables --append ose-broker-chain --match state --state NEW --proto tcp
--destination-port 80 --jump ACCEPT
# iptables --append ose-broker-chain --match state --state NEW --proto tcp
--destination-port 443 --jump ACCEPT
# iptables --append ose-broker-chain --match state --state NEW --proto tcp
--destination-port 27017 --jump ACCEPT
# iptables --append ose-broker-chain --match state --state NEW --proto tcp
--destination-port 61613 --jump ACCEPT
# iptables --append ose-broker-chain --match state --state NEW --proto tcp
--destination-port 61616 --jump ACCEPT

# iptables --list --line-numbers --numeric --verbose
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source
destination
1    1067 97985 ose-broker-chain  all  --  *      *      0.0.0.0/0
0.0.0.0/0
2     862 81609 ACCEPT      all  --  *      *      0.0.0.0/0
0.0.0.0/0          state RELATED,ESTABLISHED
3       1   80 ACCEPT      icmp  --  *      *      0.0.0.0/0
0.0.0.0/0
4       0    0 ACCEPT      all  --  lo     *      0.0.0.0/0
0.0.0.0/0
5     112 7297 REJECT      all  --  *      *      0.0.0.0/0
0.0.0.0/0          reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source
destination
1       0    0 REJECT      all  --  *      *      0.0.0.0/0
0.0.0.0/0          reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 155 packets, 18311 bytes)
num  pkts bytes target    prot opt in     out     source
destination
```



Chain ose-broker-chain (1 references)

num	pkts	bytes	target	prot	opt	in	out	source
destination								
1	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0
			state NEW	tcp	dpt:22			
2	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0
			state NEW	tcp	dpt:80			
3	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0
			state NEW	tcp	dpt:443			
4	4	240	ACCEPT	tcp	--	*	*	0.0.0.0/0
			state NEW	tcp	dpt:27017			
5	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0
			state NEW	tcp	dpt:61613			
6	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0
			state NEW	tcp	dpt:61616			

4. Save the new firewall configuration, restart iptables to activate the changes and ensure the iptables service starts on boot.

```
# iptables-save > /etc/sysconfig/iptables
# service iptables restart
# chkconfig iptables on
```



6.3 Deploy Nodes

This section details how to configure and deploy the OpenShift Enterprise node hosts.

6.3.1 Configure Entitlements

Table 6.3.1: Node Host Channels describes the channels required for deploying OpenShift Enterprise node hosts:

Channel Name	Description	Required?	Subscription
Red Hat OpenShift Enterprise Node 2.2	Base channel for OpenShift Enterprise 2.2 node hosts	Yes	“OpenShift Enterprise”
Red Hat OpenShift Enterprise JBoss EAP 2.2	Provides support for JBoss Enterprise Application Platform	Only required to support JBoss Enterprise Application Platform	“JBoss Enterprise Application Platform for OpenShift Enterprise”
JBoss Enterprise Application Platform	Provides JBoss Enterprise Application Platform	No	“JBoss Enterprise Application Platform for OpenShift Enterprise”
JBoss Enterprise Web Server 2.2	Provides Tomcat 6, Tomcat 7	No	“OpenShift Enterprise”
Red Hat OpenShift Enterprise Client Tools 2.2	Provides access to OpenShift Enterprise 2.2 Client Tools	No	“OpenShift Enterprise”
Red Hat Software Collections 1	Provides access to programming languages, databases and related packages	Yes	“OpenShift Enterprise”

Table 6.3.1: Node Host Channels

Each node host must be registered to these channels through either Red Hat Subscription Manager (RHSM) or Red Hat Network (RHN) Classic. This reference architecture utilizes Red Hat Subscription Manager.

Perform the following steps on on each node (**node1**, **node2**, **node3**):

1. Register the host (*if not already registered*).

```
# subscription-manager register
Username: username
Password: *****
```



The system has been registered with ID: 1e6430e5-e136-4419-b90f-ca296ad40732

2. Identify the OpenShift Enterprise subscription Pool ID.

```
# subscription-manager list --available
```

```
+-----+
| Available Subscriptions |
+-----+
```

...output abbreviated...

```
Subscription Name: OpenShift Enterprise
SKU:               SKU1234
Pool ID:           1234567890123456789012345678901234
Quantity:          1
Service Level:     Layered
Service Type:      L1-L3
Multi-Entitlement: No
Ends:              01/01/2022
System Type:       Physical
```

```
Subscription Name: JBoss Enterprise Application Platform for OpenShift
SKU:               SKU1234
Pool ID:           1234567890123456789012345678901234
Quantity:          1
Service Level:     Layered
Service Type:      L1-L3
Multi-Entitlement: No
Ends:              01/01/2022
System Type:       Physical
```

3. Attach the desired subscription using the Pool ID.

```
# subscription-manager attach --pool 1234567890123456789012345678901234
Successfully attached a subscription for: OpenShift Subscription
1 local certificate has been deleted.
```

4. Enable the Red Hat OpenShift Enterprise 2.2 Application Node channel.

```
# subscription-manager repos --enable rhel-6-server-ose-2.2-node-rpms
Repo rhel-6-server-ose-2.1-node-rpms is enabled for this system.
```

5. Verify the enabled channel is available.

```
# yum repolist
```

```
Loaded plugins: priorities, security, subscription-manager
This system is receiving updates from Red Hat Subscription Management.
```

...output abbreviated...

repo id	repo name
jb-eap-6-for-rhel-6-server-rpms (RHEL 6 Server)	JBoss Enterprise Application Platform 6
jb-ews-2-for-rhel-6-server-rpms Server)	JBoss Enterprise Web Server 2 (RHEL 6



rhel-6-server-ose-2.2-jbosseap-rpms	Red Hat OpenShift Enterprise 2.2 JBoss EAP add-on
rhel-6-server-ose-2.2-node-rpms	Red Hat OpenShift Enterprise 2.2 Application Node
rhel-6-server-rpms	Red Hat Enterprise Linux 6 Server
rhel-server-rhsc1-6-rpms	Red Hat Software Collections RPMs for Red Hat Enterprise Linux 6 Server

Consult the OpenShift Enterprise 2 - Deployment Guide¹² for details on configuring entitlements with Red Hat Network (RHN) Classic.

6.3.2 Configure Yum

The packages required for running OpenShift Enterprise are available from Red Hat Network (RHN). Third-party RPM repositories and even other products provided by Red Hat can create conflicts with OpenShift Enterprise during initial deployment or later when applying updates. To avoid potential issues, the **oo-admin-yum-validation** tool can be used to report on any problems and provide recommendations. By default, the tool halts so that you can review each set of proposed changes. You then have the option to apply the changes manually, or let the tool attempt to fix the issues that have been found. This process may require you to run the tool several times. You also have the option of having the tool both report all found issues, and attempt to fix all issues.

Use the **oo-admin-yum-validator** tool to configure yum and prepare to install the node host components. Perform these steps on each node host (**node1**, **node2**, **node3**) as follows.

1. Install the latest **openshift-enterprise-release** package.

```
# yum install openshift-enterprise-release
Loaded plugins: security, subscription-manager
This system is receiving updates from Red Hat Subscription Management.
rhel-6-server-ose-2.2-node-rpms           | 3.1 kB      00:00
rhel-6-server-rpms                       | 3.7 kB      00:00
Setting up Install Process
Resolving Dependencies

...output abbreviated...

Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : yum-plugin-priorities-1.1.30-17.el6_5.noarch      1/6
  Installing : openshift-enterprise-yum-validator-2.2.1-1.el6op.noarch 2/6
  Installing : compat-readline5-5.2-17.1.el6.x86_64            3/6
  Installing : ruby-libs-1.8.7.352-13.el6.x86_64               4/6
  Installing : ruby-1.8.7.352-13.el6.x86_64                   5/6
  Installing : openshift-enterprise-release-2.2.1-1.el6op.noarch 6/6
  Verifying  : openshift-enterprise-release-2.2.1-1.el6op.noarch 1/6
  Verifying  : ruby-1.8.7.352-13.el6.x86_64                   2/6
  Verifying  : compat-readline5-5.2-17.1.el6.x86_64           3/6
  Verifying  : openshift-enterprise-yum-validator-2.2.1-1.el6op.noarch 4/6
```

¹² [OpenShift Enterprise 2 - Deployment Guide](#)



```
Verifying   : yum-plugin-priorities-1.1.30-17.el6_5.noarch      5/6
Verifying   : ruby-libs-1.8.7.352-13.el6.x86_64                6/6

Installed:
  openshift-enterprise-release.noarch 0:2.2.1-1.el6op

Dependency Installed:
  compat-readline5.x86_64 0:5.2-17.1.el6  openshift-enterprise-yum-
validator.noarch 0:2.2.1-1.el6op  ruby.x86_64 0:1.8.7.352-13.el6
ruby-libs.x86_64 0:1.8.7.352-13.e  yum-plugin-priorities.noarch 0:1.1.30-
17.el6_5

Complete!
```

2. Run the **oo-admin-yum-validator** command with the **-o** option for version 2.2 and the **-r** option for the *node-eap* role. This reports the first detected set of problems, provides a set of proposed changes, and halts.

```
# oo-admin-yum-validator -o 2.2 -r node-eap
```

Alternatively, use the **--report-all** option to report all detected problems.

```
# oo-admin-yum-validator -o 2.2 -r node-eap --report-all
```

3. After reviewing the reported problems and their proposed changes, either fix them manually or let the tool attempt to fix the first set of problems using the same command with the **--fix** option. This may require several repeats of steps 2 and 3.

```
# oo-admin-yum-validator -o 2.2 -r node-eap --fix
```

Alternatively, use the **--fix-all** option to allow the tool to attempt to fix all of the problems that are found:

```
# oo-admin-yum-validator -o 2.2 -r node-eap --fix-all
```

4. Repeat steps 2 and 3 until the **oo-admin-yum-validator** command displays the following message:

```
No problems could be detected!
```

For additional details on configuring yum on broker hosts, consult the OpenShift Enterprise 2 - Deployment Guide¹³.

¹³ [OpenShift Enterprise 2 - Deployment Guide: Configuring Yum on Node Hosts](#)



6.3.3 Install OpenShift

Install OpenShift Enterprise on the *node* hosts by running the deployment script on each of the node hosts (**node1**, **node2**, **node3**).

```
# /var/tmp/deploy-ose.sh node

Deploying OpenShift Enterprise in a distributed environment

...Installation type set to *node*...

*** Control-C now if this is not a *Node* or the wrong host ***

...Continuing with installation...

...Downloading openshift.sh script...
% Total    % Received % Xferd  Average Speed   Time    Time     Time      Current
   100    131k   100    131k    0      0    169k      0  --:--:-- --:--:-- --:--:--    494k

* Script download complete *

...Beginning installation - Wed Nov 26 19:51:49 EST 2014 ...

+ environment=sh
+ case "$environment" in
+ parse_cmdline
+ parse_args
+ declare -A passwords
+ PASSWORDS_TO_DISPLAY=false
+ RESTART_NEEDED=false
+ RESTART_COMPLETED=false
+ declare -A firewall_allow
+ set_defaults

...output abbreviated...

+ service openshift-watchman restart
Stopping Watchman
Starting Watchman
+ node
+ :
+ /etc/cron.minutely/openshift-facts
+ echo 'OpenShift: Completed restarting services.'
OpenShift: Completed restarting services.
+ true
+ display_passwords
+ set +x
MCollective user: mcollective password: mcollective
+ :

* Installation completed - Wed Nov 26 20:06:48 EST 2014 *
```




6.3.3.1 Recovering from Installation Failures

In the rare event that any Node host is not able to fully complete the installation, the following steps can be used to cleanup the deployment.

1. Remove Node packages.

```
# yum remove rubygem-openshift-origin-node ruby193-rubygem-passenger-native
openshift-origin-node-util ruby193-mcollective openshift-origin-msg-node-
mcollective policycoreutils-python rubygem-openshift-origin-container-
selinux rubygem-openshift-origin-frontend-nodejs-websocket rsyslog7
rsyslog7-mmopenshift rubygem-openshift-origin-frontend-apache-mod-rewrite
```

2. Remove on-disk configuration files.

```
# rm -f /var/lib/openshift/ # Configuration files
```

Once the above steps have been completed the Node deployment can be re-run as previously described.

6.3.4 Re-configure Firewall Ports (optional)

Earlier versions of the *openshift.sh* installation script configure the firewall using the **lokkit** utility. For enterprise deployments requiring a greater degree of control and flexibility, change the firewall configuration to use **iptables**. Perform these steps on each OpenShift Enterprise node host (**node1**, **node2**, **node3**).

1. Disable the current **lokkit** firewall configuration.

```
# lokkit --disabled
```

2. Save the existing firewall configuration and create a new, default configuration.

```
# mv /etc/sysconfig/iptables{,.orig}
# service iptables restart
```

3. Modify the default new firewall chain (**rhc-app-comm**) for internal gear communications across the loop back interface. Create a new chain (**ose-node-chain**) for all external node communications and add the appropriate rules.

```
# iptables --list --line-numbers --numeric --verbose
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target      prot opt in      out      source
destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target      prot opt in      out      source
destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target      prot opt in      out      source
destination
```



```
# iptables --new-chain rhc-app-comm
# iptables --new-chain ose-node-chain
# iptables --append INPUT --jump ose-node-chain
# iptables --append INPUT --match state --state RELATED,ESTABLISHED --jump
ACCEPT
# iptables --append INPUT --protocol icmp --jump ACCEPT
# iptables --append INPUT --in-interface lo --jump ACCEPT
# iptables --append INPUT --jump rhc-app-comm
# iptables --append INPUT --jump REJECT --reject-with icmp-host-prohibited
# iptables --append FORWARD --jump REJECT --reject-with icmp-host-prohibited
# iptables --append ose-node-chain --match state --state NEW --protocol tcp
--match tcp --dport 22 --jump ACCEPT
# iptables --append ose-node-chain --match state --state NEW --protocol tcp
--match tcp --dport 80 --jump ACCEPT
# iptables --append ose-node-chain --match state --state NEW --protocol tcp
--match tcp --dport 443 --jump ACCEPT
# iptables --append ose-node-chain --match state --state NEW --protocol tcp
--match tcp --dport 8000 --jump ACCEPT
# iptables --append ose-node-chain --match state --state NEW --protocol tcp
--match tcp --dport 8443 --jump ACCEPT

# iptables --list --line-numbers --numeric --verbose
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target      prot opt in      out      source
destination
1      84  7045 ose-node-chain  all  --  *      *      0.0.0.0/0
0.0.0.0/0
2      72  5711 ACCEPT        all  --  *      *      0.0.0.0/0
0.0.0.0/0          state RELATED,ESTABLISHED
3       0    0 ACCEPT        icmp  --  *      *      0.0.0.0/0
0.0.0.0/0
4       0    0 ACCEPT        all  --  lo     *      0.0.0.0/0
0.0.0.0/0
5       3   532 rhc-app-comm  all  --  *      *      0.0.0.0/0
0.0.0.0/0
6       3   532 REJECT        all  --  *      *      0.0.0.0/0
0.0.0.0/0          reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target      prot opt in      out      source
destination
1       0    0 REJECT        all  --  *      *      0.0.0.0/0
0.0.0.0/0          reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 9 packets, 798 bytes)
num  pkts bytes target      prot opt in      out      source
destination

Chain ose-node-chain (1 references)
num  pkts bytes target      prot opt in      out      source
destination
1       0    0 ACCEPT        tcp  --  *      *      0.0.0.0/0
0.0.0.0/0          state NEW tcp dpt:22
2       0    0 ACCEPT        tcp  --  *      *      0.0.0.0/0
```



```

0.0.0.0/0          state NEW tcp dpt:80
3          0          0 ACCEPT      tcp -- *          *          0.0.0.0/0
0.0.0.0/0          state NEW tcp dpt:443
4          0          0 ACCEPT      tcp -- *          *          0.0.0.0/0
0.0.0.0/0          state NEW tcp dpt:8000
5          0          0 ACCEPT      tcp -- *          *          0.0.0.0/0
0.0.0.0/0          state NEW tcp dpt:8443

```

Chain rhc-app-comm (1 references)

```

num  pkts bytes target    prot opt in      out     source
destination

```

4. Save the new firewall configuration, restart iptables to activate the changes and ensure the iptables service starts on boot.

```

# iptables-save > /etc/sysconfig/iptables
# service iptables restart
# chkconfig iptables on

```

6.3.5 Verify Communications

Verify basic firewall communications between the broker hosts and node hosts by running the **oo-mco ping** utility from each broker host (**broker1**, **broker2**, **broker3**) as follows.

```

# oo-mco ping
node3.interop.example.com      time=87.73 ms
node1.interop.example.com      time=127.33 ms
node2.interop.example.com      time=128.06 ms

---- ping statistics ----
3 replies max: 128.06 min: 87.73 avg: 114.37

```

If any node host does not respond then check the firewall configuration.



6.4 Post Deployment Tasks

This section details the post deployment tasks to be performed after the initial deployment of all OpenShift Enterprise broker and node hosts.

6.4.1 Run Post_Deploy

Run the deployment script with the post deployment parameter (***post-deploy***) on the first broker host (**broker1**) only.

```
# hostname
broker1.interop.example.com

# /var/tmp/deploy-ose.sh post-deploy
Deploying OpenShift Enterprise in a distributed environment

...Installation type set to *post-deploy*...

...Continuing with post deployment tasks...

...Downloading openshift.sh script...

% Total    % Received % Xferd  Average Speed   Time    Time     Time        Current
% Total    % Received % Xferd  Average Speed   Time    Time     Time        Current
100 131k 100 131k  0    0  240k    0  ---:---:--  ---:---:--  ---:---:--  464k

* Script download complete *

...Beginning post deployment tasks - Wed Nov 26 21:22:32 ...
+ environment=sh
+ case "$environment" in
+ parse_cmdline actions=post_deploy
+ parse_args actions=post_deploy
+ for word in "$@"
+ key=actions
+ case "$word" in
+ val=post_deploy

...output abbreviated...

+ echo 'OpenShift: Completed configuring districts.'
OpenShift: Completed configuring districts.
+ echo 'OpenShift: Completed post deployment steps.'
OpenShift: Completed post deployment steps.
+ false
+ false
+ :

* Post deployment tasks completed - Wed Nov 26 21:23:00 *
```

The post deployment action configures districts and imports cartridges on node hosts.



6.4.1.1 Recovering from Post Deployment Failures

In the rare event that the Post_Deploy tasks are not able to fully complete then the Node hosts districts may not be properly configured. This can be seen running **oo-diagnostics** on any Broker host.

```
# oo-diagnostics

WARN: block in test_node_profiles_districts_from_broker
      There are no node hosts in district 'default-small'
FAIL: test_node_profiles_districts_from_broker
      Default gear profile 'small' has no active node hosts supplying it in
any district.
      Attempts to create apps without specifying gear size will fail.
      Please add active node hosts to a district with profile 'small'
      using oo-admin-ctl-district or fix the settings in
/etc/openshift/broker.conf
```

The following steps can be used to recover from the post deployment failure:

1. Remove District Info File (**node1, node2, node3**).

```
# rm -f /var/lib/openshift/.settings/district.info
```

2. Run **oo-admin-ctl-district** on any Broker host.

```
# oo-admin-ctl-district -p small -n default-small -c add-node --available
Success for node 'node1.interop.example.com'!
Success for node 'node2.interop.example.com'!
Success for node 'node2.interop.example.com'!
```

Alternatively, the full post deployment can be run after removing the district files in Step 1.

6.4.2 Copy public rsync key to enable moving gears

The broker rsync public key needs to be authorized on nodes to enable the moving of gears. Without the public key in place, each gear move operation will require entering the root password for each node host involved.

During installation, the *openshift.sh* script (called by *deploy-ose.sh*) puts a copy of the public key on the broker web server so that nodes can access the key. If the installation fails for any reason, manually copy the public rsync key to each node host as follows.

```
# wget -O- --no-check-certificate https://broker1/rsync_id_rsa.pub >> \
/root/.ssh/authorized_keys
# wget -O- --no-check-certificate https://broker2/rsync_id_rsa.pub >> \
/root/.ssh/authorized_keys
# wget -O- --no-check-certificate https://broker3/rsync_id_rsa.pub >> \
/root/.ssh/authorized_keys
```

Perform this step on each node host (**node1, node2, node3**) only if required.



6.4.3 Copy ssh host keys, httpd key/cert across Nodes

All node hosts should identify themselves as the same host so that when gears are moved between hosts, **ssh** and **git** don't display warnings about host keys changing. Perform the following steps to ensure that all node hosts have the same ssh host keys.

1. On each node host (**node1**, **node2**, **node3**), make a backup copy of all `/etc/ssh/ssh_host_*` files.

```
# cd /etc/ssh/
# mkdir -p host-key-backup
# cp -p ssh_host_* host-key-backup/.
```

2. On the first node host (**node1**), copy the `/etc/ssh/ssh_host_*` files to the other node hosts (**node2**, **node3**).

```
# scp -p /etc/ssh/ssh_host_* node2:/etc/ssh/.
Warning: Permanently added 'node2,10.19.140.22' (RSA) to the list of known
hosts.
root@node3's password: *****
ssh_host_dsa_key                100% 668      0.7KB/s   00:00
ssh_host_dsa_key.pub            100% 590      0.6KB/s   00:00
ssh_host_key                    100% 963      0.9KB/s   00:00
ssh_host_key.pub                100% 627      0.6KB/s   00:00
ssh_host_rsa_key                100% 1675     1.6KB/s   00:00
ssh_host_rsa_key.pub            100% 382      0.4KB/s   00:00

# scp -p /etc/ssh/ssh_host_* node3:/etc/ssh/.
Warning: Permanently added 'node2,10.19.140.23' (RSA) to the list of known
hosts.
root@node3's password: *****
ssh_host_dsa_key                100% 668      0.7KB/s   00:00
ssh_host_dsa_key.pub            100% 590      0.6KB/s   00:00
ssh_host_key                    100% 963      0.9KB/s   00:00
ssh_host_key.pub                100% 627      0.6KB/s   00:00
ssh_host_rsa_key                100% 1675     1.6KB/s   00:00
ssh_host_rsa_key.pub            100% 382      0.4KB/s   00:00
```

3. Restart **sshd** on each node host (**node1**, **node2**, **node3**).

```
# service sshd restart
Stopping sshd:                [ OK ]
Starting sshd:                 [ OK ]
```

Similarly, **https** access to gears that have been moved will display errors if the certificate is not identical across node hosts. Perform the following steps to ensure that all node hosts have the same **http** keys and certificates.

1. On each node host (**node1**, **node2**, **node3**), make a backup copy of the localhost key and certificate files:

```
# cp -p /etc/pki/tls/private/localhost.key{,.orig}
```



```
# cp -p /etc/pki/tls/certs/localhost.crt{,.orig}
```

2. On the first node host (**node1**), copy the localhost key and cert files to the other nodes (**node2**, **node3**).

```
# scp -p /etc/pki/tls/private/localhost.key node2:/etc/pki/tls/private/localhost.key
localhost.key                                100% 1704      1.7KB/s   00:00
# scp -p /etc/pki/tls/private/localhost.key node3:/etc/pki/tls/private/localhost.key
localhost.key                                100% 1704      1.5KB/s   00:00

# scp -p /etc/pki/tls/certs/localhost.crt node2:/etc/pki/tls/certs/localhost.crt
localhost.crt                                100% 1574      1.5KB/s   00:00
# scp -p /etc/pki/tls/certs/localhost.crt node3:/etc/pki/tls/certs/localhost.crt
localhost.crt                                100% 1574      1.6KB/s   00:00
```

3. Restart **httpd** on each node host (**node1**, **node2**, **node3**).

```
# service httpd restart
Stopping httpd:                                [ OK ]
Starting httpd:                                [ OK ]
```

Any warnings displayed about modules already loaded can be safely ignored.

6.4.4 Copy Auth Keys across Brokers

The broker authentication keys need to be identical across all broker hosts so that gear operations (auto-scaling, Jenkins builds, recording deployments) can complete without authentication failures. Perform the following step to ensure that all broker hosts have the same authentication key:

1. On each broker host (**broker1**, **broker2**, **broker3**), make a backup copy of all `/etc/openshift/server_*.pem` files.

```
# cd /etc/openshift
# mkdir -p auth-key-backup
# cp -p /etc/openshift/server_*.pem auth-key-backup/.
```

2. On the first broker host (**broker1**), copy the `/etc/openshift/server_*.pem` files to the other broker hosts (**broker2**, **broker3**).

```
# scp -p /etc/openshift/server_*.pem broker2:/etc/openshift/.
Warning: Permanently added 'broker2,10.19.140.12' (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

# scp -p /etc/openshift/server_*.pem broker3:/etc/openshift/.
Warning: Permanently added 'node3,10.19.140.13' (RSA) to the list of known
hosts.
root@node3's password: *****
server_priv.pem                                100% 1679      1.6KB/s   00:00
server_pub.pem                                100%  451      0.4KB/s   00:00
```



6.4.5 Configure Broker DNS Request Balancing (optional)

The proper balancing of client requests across all OpenShift broker hosts is essential to providing improved performance and availability of services. Most enterprise environments utilize an external (hardware or software) load balancer solution.

In this reference architecture, a simple round-robin DNS configuration is used to automatically cycle requests to the host named **broker.interop.example.com** across all three OpenShift broker hosts (**broker1**, **broker2**, **broker3**). Configure round-robin DNS on the IdM server (**idm-srv1**) as follows:

1. Create DNS forward zone entries in the *interop.example.com* DNS zone for a host named **broker.interop.example.com**. Each DNS A record entry contains the IP address of an OpenShift broker host and all entries resolve to the same hostname – **broker.interop.example.com**.

```
# hostname
idm-srv1.interop.example.com

# kinit admin
Password for admin@INTEROP.EXAMPLE.COM: *****

# ipa dnsrecord-add interop.example.com broker --a-ip-address=10.19.140.11
Record name: broker
A record: 10.19.140.11

# ipa dnsrecord-add interop.example.com broker --a-ip-address=10.19.140.12
Record name: broker
A record: 10.19.140.11, 10.19.140.12

# ipa dnsrecord-add interop.example.com broker --a-ip-address=10.19.140.13
Record name: broker
A record: 10.19.140.11, 10.19.140.12, 10.19.140.13

# ipa dnsrecord-show interop.example.com broker
Record name: broker
A record: 10.19.140.11, 10.19.140.12, 10.19.140.13

# nslookup broker.interop.example.com
Server:      10.19.140.101
Address:     10.19.140.101#53

Name:      broker.interop.example.com
Address: 10.19.140.13
Name:      broker.interop.example.com
Address: 10.19.140.12
Name:      broker.interop.example.com
Address: 10.19.140.11
```

2. Verify all requests to **broker.interop.example.com** now alternate across the three broker hosts.

```
# hostname
idm-srv2.interop.example.com
```




```
# nslookup broker.interop.example.com
Server:      10.19.140.101
Address:     10.19.140.101#53

Name:   broker.interop.example.com
Address: 10.19.140.13
Name:   broker.interop.example.com
Address: 10.19.140.12
Name:   broker.interop.example.com
Address: 10.19.140.11

# ping broker
PING broker.interop.example.com (10.19.140.11) 56(84) bytes of data.
64 bytes from broker1.interop.example.com (10.19.140.11): icmp_seq=1 ttl=64
time=0.374 ms
^C
--- broker.interop.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.374/0.374/0.374/0.000 ms

# ping broker
PING broker.interop.example.com (10.19.140.12) 56(84) bytes of data.
64 bytes from broker2.interop.example.com (10.19.140.12): icmp_seq=1 ttl=64
time=1.01 ms
^C
--- broker.interop.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.015/1.015/1.015/0.000 ms
#
```

6.4.6 Update DNS Entries

Note: In **Section 7.2.3 Configure IdM Clients** DNS entries for all OpenShift hosts are created in IdM when the **ipa-client-install** utility is run. If any OpenShift hosts have already been added to the IdM domain then this step is not required.

1. Create DNS forward and reverse zone entries in the *interop.example.com* DNS zone for each of the OpenShift broker hosts, node hosts and the rhc client. Perform these steps on either one of the IdM servers (**idm-srv1**, **idm-srv2**).

```
# hostname
idm-srv1.interop.example.com

# kinit admin
Password for admin@INTEROP.EXAMPLE.COM:

# ipa dnsrecord-add interop.example.com broker1 --a-ip-address=10.19.140.11
```



```
--a-create-reverse
# ipa dnsrecord-add interop.example.com broker2 --a-ip-address=10.19.140.12
--a-create-reverse
# ipa dnsrecord-add interop.example.com broker3 --a-ip-address=10.19.140.13
--a-create-reverse

# ipa dnsrecord-add interop.example.com node1 --a-ip-address=10.19.140.21
--a-create-reverse
# ipa dnsrecord-add interop.example.com node2 --a-ip-address=10.19.140.22
--a-create-reverse
# ipa dnsrecord-add interop.example.com node3 --a-ip-address=10.19.140.23
--a-create-reverse

# ipa dnsrecord-add interop.example.com rhc1 --a-ip-address=10.19.140.10
--a-create-reverse
```

2. Verify the entries are correct within IdM.

```
# ipa dnsrecord-show interop.example.com broker1
Record name: broker1
A record: 10.19.140.11
# ipa dnsrecord-show interop.example.com broker2
Record name: broker2
A record: 10.19.140.12
# ipa dnsrecord-show interop.example.com broker3
Record name: broker3
A record: 10.19.140.13

# ipa dnsrecord-show interop.example.com node1
Record name: node1
A record: 10.19.140.21
# ipa dnsrecord-show interop.example.com node2
Record name: node2
A record: 10.19.140.22
# ipa dnsrecord-show interop.example.com node3
Record name: node3
A record: 10.19.140.23

# ipa dnsrecord-show interop.example.com rhc1
Record name: rhc1
A record: 10.19.140.10
```

3. Verify the entries can be found from any OpenShift host.

```
# hostname
node1.interop.example.com

# nslookup node1
Server:          10.19.140.101
Address:         10.19.140.101#53

Name:   node1.interop.example.com
Address: 10.19.140.21
```



6.5 Deploy RHC Client

This section details how to configure and deploy the OpenShift Enterprise RHC clients.

6.5.1 Configure Entitlements

Table 6.5.1: RHC Client Channel describes the channels required for deploying OpenShift Enterprise RHC client:

Channel Name	Description	Required?	Subscription
Red Hat OpenShift Enterprise Client Tools 2.2	Provides access to OpenShift Enterprise 2.2 Client Tools	Yes	"OpenShift Enterprise"

Table 6.5.1: RHC Client Channel

Each RHC client must be registered to the channel through either Red Hat Subscription Manager (RHSM) or Red Hat Network (RHN) Classic. This reference architecture utilizes Red Hat Subscription Manager.

Perform the following steps on on each RHC client (**rhc1**).

1. Register the host (*if not already registered*).

```
# subscription-manager register
Username: username
Password: *****
The system has been registered with ID: 1e6430e5-e136-4419-b90f-ca296ad40732
```

2. Identify the OpenShift Enterprise subscription Pool ID.

```
# subscription-manager list --available
+-----+
| Available Subscriptions |
+-----+

...output abbreviated...

Subscription Name: OpenShift Enterprise
SKU:              SKU1234
Pool ID:          1234567890123456789012345678901234
Quantity:         1
Service Level:    Layered
Service Type:     L1-L3
Multi-Entitlement: No
Ends:             01/01/2022
System Type:      Physical
Subscription Name: JBoss Enterprise Application Platform for OpenShift
SKU:              SKU1234
Pool ID:          1234567890123456789012345678901234
Quantity:         1
```



```
Service Level: Layered
Service Type: L1-L3
Multi-Entitlement: No
Ends: 01/01/2022
System Type: Physical
```

3. Attach the desired subscription using the Pool ID.

```
# subscription-manager attach --pool-id 1234567890123456789012345678901234
Successfully attached a subscription for: OpenShift Subscription
1 local certificate has been deleted
```

4. Enable the **Red Hat OpenShift Enterprise Client Tools 2.2** channel.

```
# subscription-manager repos --enable rhel-6-server-ose-2.2-rhc-rpms
Repo rhel-6-server-ose-2.1-rhc-rpms is enabled for this system.
```

5. Verify the enabled channel is available.

```
# yum repolist
Loaded plugins: priorities, security, subscription-manager
This system is receiving updates from Red Hat Subscription Management.

...output abbreviated...

repo id                                repo name
rhel-6-server-ose-2.2-rhc-rpms        Red Hat OpenShift Enterprise 2.2 Client Tools
(RPMs) 15
rhel-6-server-rpms                    Red Hat Enterprise Linux 6 Server (RPMs)
12,701
```

Consult the OpenShift Enterprise 2 - Client Tools Installation Guide¹⁴ for details on configuring entitlements with Red Hat Network (RHN) Classic.

6.5.2 Install Package

Install the **RHC** client package on **rhc1**.

```
# yum install rhc
Loaded plugins: product-id, security, subscription-manager
This system is receiving updates from Red Hat Subscription Management.
Installed:
  rhc.noarch 0:1.24.3.1-1.el6op

...output abbreviated...

Dependency Installed:
  compat-readline5.x86_64 0:5.2-17.1.el6  git.x86_64 0:1.7.1-3.el6_4.1
  perl-Error.noarch 1:0.17015-4.el6
  perl-Git.noarch 0:1.7.1-3.el6_4.1  ruby.x86_64 0:1.8.7.352-13.el6_2
  ruby-libs.x86_64 0:1.8.7.352-13.el6_2
  rubygem-archive-tar-minitar.noarch 0:0.5.2-3.1.el6op  rubygem-
  commander.noarch 0:4.0.3-4.el6op  rubygem-highline.noarch 0:1.6.16-
  1.el6op
```

¹⁴ [OpenShift Enterprise 2 - Client Tools Installation Guide](#)



```
rubygem-httpclient.noarch 0:2.3.2-1.el6op rubygem-net-scp.noarch 0:1.1.2-  
2.el6op          rubygem-net-ssh.noarch 0:2.7.0-1.el6op  
rubygem-net-ssh-gateway.noarch 0:1.2.0-1.el6op          rubygem-net-ssh-  
multi.noarch 0:1.2.0-1.el6op          rubygem-open4.noarch 0:1.3.0-2.el6op  
rubygem-parseconfig.noarch 0:0.5.2-5.el6op          rubygem-test-  
unit.noarch 0:2.2.0-3.el6op          rubygems.noarch 0:1.8.24-6.el6op
```

Complete!

Note: At this stage the setup and verification of the RHC client is normally done. Since Dynamic DNS has not yet been configured, deployed applications can not be registered until all integration prerequisites have been met in **(Section 7.2 Integration Prerequisites)** and the first integration scenario **IS1: Developer Authorization to Brokers (mandatory)** has been completed.



6.6 Verify OpenShift Enterprise Environment

Perform the following set of steps to verify all components of the OpenShift Enterprise environment are fully functional.

6.6.1 Reboot Hosts

Reboot each OpenShift Enterprise host to confirm all services have been correctly configured. First reboot each broker host then reboot each node host after all brokers have completed booting.

```
# shutdown -r now # broker1.interop.example.com
# shutdown -r now # broker2.interop.example.com
# shutdown -r now # broker3.interop.example.com
# shutdown -r now # node1.interop.example.com
# shutdown -r now # node2.interop.example.com
# shutdown -r now # node2.interop.example.com
```

6.6.2 Run Diagnostics

Run the **oo-diagnostics** utility on all broker and node hosts to confirm the status of the OpenShift environment.

```
# oo-diagnostics
```

All broker hosts (**broker1**, **broker2**, **broker3**) should report the following error.

```
FAIL: run_script
oo-accept-broker had errors:
--BEGIN OUTPUT--
could not read key secret
syntax error
FAIL: error adding txt record name testrecord.interop.example.com to server
10.19.140.101: this_is_a_test
    -- is the nameserver running, reachable, and key auth working?
FAIL: txt record testrecord.interop.example.com does not resolve on server
10.19.140.101
could not read key secret
syntax error
FAIL: error deleting txt record name testrecord.interop.example.com to
server 10.19.140.101:
    -- is the nameserver running, reachable, and key auth working?
3 ERRORS
--END oo-accept-broker OUTPUT--
```

Note: This error is normal since Dynamic DNS is not configured until all integration prerequisites have been met (**Section 7.2 Integration Prerequisites**) and **IS1: Developer Authorization to Brokers (mandatory)** completed.



If any other issues are found, check the appropriate log files and restart any services that are not running as per **Table 6.6.2: Common OpenShift Services and Logs** below:

Component	Location	Log File(s)	Service Verify
ActiveMQ	All brokers	/var/log/messages /var/log/activemq/*.log	# service activemq status # service activemq restart
MCollective (client)	All brokers	/var/log/messages /var/log/openshift/broker/ruby193-mcollective-client.log	n/a
MCollective (agent)	All nodes	/var/log/messages /var/log/openshift/node/ruby193-mcollective.log	# service ruby193-mcollective status # service ruby193-mcollective restart
MongoDB	All brokers	/var/log/messages /var/log/mongodb/mongodb.log	# service mongod status # service mongod restart

Table 6.6.2: Common OpenShift Services and Logs

Consult the OpenShift Enterprise 2 Troubleshooting Guide¹⁵ for further troubleshooting guidelines.

¹⁵[OpenShift Enterprise 2 - Troubleshooting Guide](#)



7 Integrating OpenShift Enterprise with IdM

This section details areas where OpenShift Enterprise can leverage various features and services provided by Identity Management in Red Hat Enterprise Linux. The authentication and authorization framework in IdM is ideal for OpenShift integration as it supports industry standard protocols and services (Kerberos, LDAP, DNS, NTP, X509 Certificates), and combines them into a secure, reliable and scalable identity management framework.

7.1 Overview

Figure 7.1: Integration Scenarios - Overview provides an overview of each integration scenario as detailed in the sections that follow:

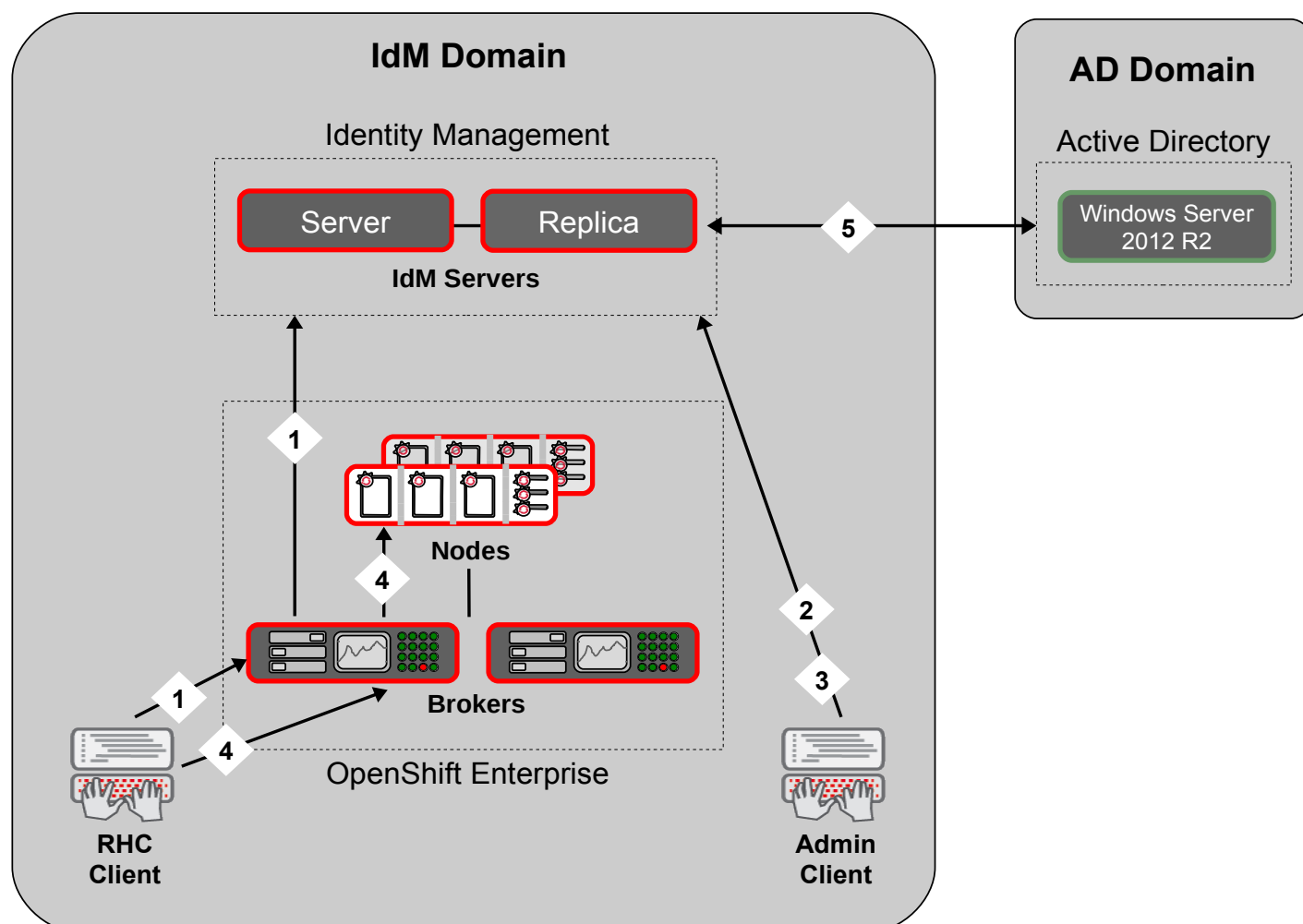


Figure 7.1: Integration Scenarios - Overview



During the systems engineering phase of this reference architecture, over a dozen potential integration scenarios were identified and subsequently narrowed down to those shown here:

- IS1: Developer Authorization to Brokers (**mandatory**)
- IS2: Centralized Management of SSH Keys (optional)
- IS3: Centralized Management of UID's (optional)
- IS4: Kerberized Gear Access (optional)
- IS5: Cross-realm Kerberos Trust (optional)

Note: The first integration scenario is **mandatory** and must be fully completed. Until this scenario is completed, no applications can be deployed and dynamically registered within the IdM DNS service.

Each of these scenarios has been validated, sequenced and streamlined for deployment purposes. Most of the integration scenarios are considered optional, but most environments will find these beneficial to implement as they extend the core functionality of the integration framework.



7.2 Integration Prerequisites

Prior to proceeding with the first integration scenario, each OpenShift host must have the firewall ports, DNS configurations verified and then be made a member (client) of the IdM domain. Perform the following tasks on each OpenShift Enterprise broker host (**broker1**, **broker2**, **broker3**), node host (**node1**, **node2**, **node3**) and client (**rhc1**):

- 7.2.1 Configure Firewall Ports
- 7.2.2 Verify DNS
- 7.2.3 Configure IdM Clients

Do not proceed with any of the integration scenarios in **Section 7.3 Integration Scenarios** until each of the tasks above has been fully completed.

7.2.1 Configure Firewall Ports

Create a new chain containing the firewall rules required by IdM clients. Perform these steps on all OpenShift hosts (**broker1**, **broker2**, **broker3**, **node1**, **node2**, **node3**, **rhc1**).

1. Save the existing firewall configuration.

```
# cp -p /etc/sysconfig/iptables{,.pre-idm}
```

2. Create a new chain (**ipa-client-chain**) containing the firewall rules for the ports required by all IdM clients. The output from an OpenShift broker host is shown here.

```
# iptables --new-chain ipa-client-chain
# iptables --insert INPUT --jump ipa-client-chain

# iptables --append ipa-client-chain --proto tcp --destination-port 80
--jump ACCEPT
# iptables --append ipa-client-chain --proto tcp --destination-port 443
--jump ACCEPT
# iptables --append ipa-client-chain --proto tcp --destination-port 389
--jump ACCEPT
# iptables --append ipa-client-chain --proto tcp --destination-port 636
--jump ACCEPT
# iptables --append ipa-client-chain --proto tcp --destination-port 88
--jump ACCEPT
# iptables --append ipa-client-chain --proto tcp --destination-port 464
--jump ACCEPT
# iptables --append ipa-client-chain --proto udp --destination-port 88
--jump ACCEPT
# iptables --append ipa-client-chain --proto udp --destination-port 464
--jump ACCEPT
# iptables --append ipa-client-chain --proto tcp --destination-port 53
--jump ACCEPT
# iptables --append ipa-client-chain --proto udp --destination-port 53
--jump ACCEPT
# iptables --append ipa-client-chain --proto udp --destination-port 123
```



```
--jump ACCEPT
```

```
# iptables --list --line-numbers --numeric --verbose
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
```

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	4890	495K	ipa-client-chain	all	--	*	*	0.0.0.0/0	0/0/0/0/0
2	4889	495K	ose-broker-chain	all	--	*	*	0.0.0.0/0	0.0.0.0/0
3	4821	489K	ACCEPT	all	--	*	*	0.0.0.0/0	0.0.0.0/0
state RELATED,ESTABLISHED									
4	0	0	ACCEPT	icmp	--	*	*	0.0.0.0/0	0.0.0.0/0
5	1	58	ACCEPT	all	--	lo	*	0.0.0.0/0	0.0.0.0/0
6	24	3584	REJECT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

```
reject-with icmp-host-prohibited
```

```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
```

num	pkts	bytes	target	prot	opt	in	out	source	destination
1	0	0	REJECT	all	--	*	*	0.0.0.0/0	0.0.0.0/0

```
reject-with icmp-host-prohibited
```

```
Chain OUTPUT (policy ACCEPT 4799 packets, 471K bytes)
```

num	pkts	bytes	target	prot	opt	in	out	source	destination
-----	------	-------	--------	------	-----	----	-----	--------	-------------

```
Chain ipa-client-chain (1 references)
```

num	pkts	bytes	target	prot	opt	in	out	source	destination	
1	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:80
2	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:443
3	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:389
4	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:636
5	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:88
6	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:464
7	0	0	ACCEPT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	udp dpt:88
8	0	0	ACCEPT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	udp dpt:464
9	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:53
10	0	0	ACCEPT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	udp dpt:53
11	1	76	ACCEPT	udp	--	*	*	0.0.0.0/0	0.0.0.0/0	udp dpt:123

```
Chain ose-broker-chain (1 references)
```

num	pkts	bytes	target	prot	opt	in	out	source	destination	
1	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	state NEW
tcp dpt:22										
2	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	state NEW
tcp dpt:80										
3	0	0	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	state NEW
tcp dpt:443										
4	39	2467	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	state NEW
tcp dpt:27017										
5	2	106	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	state NEW
tcp dpt:61613										
6	2	126	ACCEPT	tcp	--	*	*	0.0.0.0/0	0.0.0.0/0	state NEW
tcp dpt:61616										

Each of the ports are described in further detail in **Table 3.3.4 Network Ports**.



3. Save the new firewall configuration, restart iptables to activate the changes and ensure the iptables service starts on boot.

```
# iptables-save > /etc/sysconfig/iptables
# service iptables restart
iptables: Setting chains to policy ACCEPT: filter      [ OK ]
iptables: Flushing firewall rules:                    [ OK ]
iptables: Unloading modules:                           [ OK ]
iptables: Applying firewall rules:                     [ OK ]

# chkconfig iptables on
```

7.2.2 Verify DNS

Verify that the DNS resolver file (*/etc/resolv.conf*) includes the names of the IdM server and replica. Any additional DNS servers should be located *after* the IdM servers. Perform this task on all OpenShift hosts (**broker1**, **broker2**, **broker3**, **node1**, **node2**, **node3**, **rhc1**).

```
domain interop.example.com
search interop.example.com
nameserver 10.19.140.101
nameserver 10.19.140.102
nameserver 10.19.nnn.nnn
```

7.2.3 Configure IdM Clients

Configure each OpenShift host to become an IdM client and verify Kerberos, IdM lookups. Perform this on OpenShift hosts (**broker1**, **broker2**, **broker3**, **node1**, **node2**, **node3**, **rhc1**).

1. Install the IPA client package.

```
# yum install ipa-client
Loaded plugins: priorities, security, subscription-manager
Setting up Install Process

...output abbreviated...

Downloading Packages:
ipa-client-3.0.0-42.el6.x86_64.rpm
| 145 kB      00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : ipa-client-3.0.0-42.el6.x86_64      1/1
  Verifying   : ipa-client-3.0.0-42.el6.x86_64      1/1

Installed:
  ipa-client.x86_64 0:3.0.0-42.el6

Complete!
```



2. Run the IPA client install - the output from an OpenShift broker host is shown here.

```
# ipa-client-install --enable-dns-updates --ssh-trust-dns --mkhomedir
Discovery was successful!
Hostname: broker1.interop.example.com
Realm: INTEROP.EXAMPLE.COM
DNS Domain: interop.example.com
IPA Server: idm-srv1.interop.example.com
BaseDN: dc=interop,dc=example,dc=com

Continue to configure the system with these values? [no]: yes
User authorized to enroll computers: admin
Password for admin@INTEROP.EXAMPLE.COM: *****
Synchronizing time with KDC...
Unable to sync time with IPA NTP server, assuming the time is in sync. Please
check that 123 UDP port is opened.

Successfully retrieved CA cert
  Subject:      CN=Certificate Authority,O=INTEROP.EXAMPLE.COM
  Issuer:       CN=Certificate Authority,O=INTEROP.EXAMPLE.COM
  Valid From:   Tue Jun 17 14:25:45 2014 UTC
  Valid Until:  Sat Jun 17 14:25:45 2034 UTC

Enrolled in IPA realm INTEROP.EXAMPLE.COM
Created /etc/ipa/default.conf
New SSSD config will be created
Configured /etc/sss/sss.conf
Configured /etc/krb5.conf for IPA realm INTEROP.EXAMPLE.COM
trying https://idm-srv1.interop.example.com/ipa/xml
Forwarding 'env' to server u'https://idm-srv1.interop.example.com/ipa/xml'
DNS server record set to: broker1.interop.example.com -> 10.19.140.11
Adding SSH public key from /etc/ssh/ssh_host_rsa_key.pub
Adding SSH public key from /etc/ssh/ssh_host_dsa_key.pub
Forwarding 'host_mod' to server u'https://idm-srv1.interop.example.com/ipa/xml'
SSSD enabled
Configured /etc/openldap/ldap.conf
NTP enabled
Configured /etc/ssh/ssh_config
Configured /etc/ssh/sshd_config
Client configuration complete.
```

The command options are as follows:

- The **--enable-dns-updates** flag permits an IdM client to dynamically register its IP address with the DNS service on the IdM servers. In environments where client IP addresses are configured through DHCP, this flag is essential.
- The **--ssh-trust-dns** flag configures OpenSSH to trust the IdM DNS records where the host keys are stored.
- The **--mkhomedir** flag automatically creates new home directories on the client upon a user's first login.



Note: If DNS is properly configured then **ipa-client-install** will detect the IdM server through DNS autodiscovery. If autodiscovery fails, the client installation can be re-run by specifying the **--server IdM-server.FQDN** command option. To avoid future issues, the underlying DNS issue should be further investigated. Checking the DNS resolver file (*/etc/resolv.conf*) and appropriate log files.

3. Verify that Kerberos and IdM lookups are functioning correctly on each OpenShift host.

```
# kinit admin
Password for admin@INTEROP.EXAMPLE.COM: *****
# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting      Expires            Service principal
12/19/14 15:45:31  12/20/14 15:45:28  krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM

# id admin
uid=1550200000(admin) gid=1550200000(admins) groups=1550200000(admins)
# id ose-admin1
uid=8890000001(ose-admin1) gid=8890000001 groups=8890000001
# id ose-dev1
uid=8890000002(ose-dev1) gid=8890000002 groups=8890000002
# id ose-user1
uid=8890000003(ose-user1) gid=8890000003 groups=8890000003
```

Note: Another client install issue that may occur is when the client CA certificate is out of sync with the IdM server. This can occur if the server has been re-deployed but the client still has the old CA certificate. The error is reported back as:

```
LDAP Error: Connect error: TLS error -8054:You are attempting to
import a cert with the same issuer/serial as an existing cert,
but that is not the same cert.
```

To correct, either re-name or delete the ca.cert file:

```
# ll /etc/ipa
total 4
-rw-r--r--. 1 root root 1329 Feb 11 16:11 ca.crt
# mv /etc/ipa/ca.crt /etc/ipa/ca.crt.bad
```

and re-run the IPA client installation.

In general, should any other issues be encountered during the installation, the IdM client can simply be removed:

```
# ipa-client-install --uninstall
```

and the installation restarted as previously described above.



7.3 Integration Scenarios

7.3.1 IS1: Developer Authorization to Brokers (mandatory)

Integration Scenario 1 allows developers running the OpenShift RHC client to authenticate to OpenShift brokers using Kerberos to a centralized LDAP store - Identity Management in Red Hat Enterprise Linux 7.

Integration Scenario 1 Developer Authorization to Brokers	
Components	<ul style="list-style-type: none">• OpenShift Enterprise 2.2• Identity Management in Red Hat Enterprise Linux 7
Description	<ul style="list-style-type: none">• Allow RHC client authentication to OpenShift Enterprise brokers using Kerberos and a centralized LDAP store (IdM)
Target User	<ul style="list-style-type: none">• Developers
Configuration Files	<ul style="list-style-type: none">• <code>/etc/openshift/broker.conf</code>• <code>/var/www/openshift/broker/httpd/conf.d/http.keytab</code>• <code>/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf</code>• <code>/var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf</code>• <code>/etc/openshift/plugins.d/openshift-origin-dns-nsupdate.conf</code>• <code>/etc/dns.keytab</code>
OpenShift Services	<ul style="list-style-type: none">• <code>openshift-broker</code> (<i>restart</i>)• <code>openshift-console</code> (<i>restart</i>)
IdM Services	<ul style="list-style-type: none">• HTTP (<i>new</i>)• DNS (<i>new</i>)
Notes	<ul style="list-style-type: none">• Kerberos must be previously configured on all hosts and clients• Keytabs are created for each IdM service - HTTP, DNS

Integration Tasks - Summary

The following sequence of steps are required to configure integration scenario 1:

1. Create broker HTTP services (**idm-srv1**)
2. Create broker HTTP keytabs (**broker1**)
3. Configure broker, console to authenticate using Kerberos (**broker1, broker2, broker3**)
4. Restart broker and console services (**broker1, broker2, broker3**)
5. Configure nsupdate plugin (**broker1, broker2, broker3**)
6. Create broker DNS services (**idm-srv1**)
7. Modify DNS zone for dynamic DNS (**idm-srv1**)



8. Create broker DNS keytabs (**broker1, broker2, broker3**)
9. Restart broker services (**broker1, broker2, broker3**)
10. Configure RHC client (**rhc1**)
11. Verify RHC client (**rhc1**)
12. Verify OpenShift (**broker1, broker2, broker3, node1, node2, node3**)
13. Additional Verification and Troubleshooting Steps (**rhc1**)

Integration Tasks - Details

1. Create broker HTTP service.

On the IdM server (**idm-srv1**), create an HTTP web service for the brokers - the web service permits the OpenShift brokers to communicate to the IdM servers using Kerberos.

```
# hostname
idm-srv1.interop.example.com

# kinit admin
Password for admin@INTEROP.EXAMPLE.COM: *****
# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting          Expires                Service principal
12/23/2014 10:59:54    12/24/2014 10:59:49    krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM

# ipa service-add HTTP/broker.interop.example.com
-----
Added service "HTTP/broker.interop.example.com@INTEROP.EXAMPLE.COM"
-----
Principal: HTTP/broker.interop.example.com@INTEROP.EXAMPLE.COM
Managed by: broker.interop.example.com

# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting          Expires                Service principal
12/23/2014 11:00:05    12/24/2014 10:59:49    HTTP/idm-
srv1.interop.example.com@INTEROP.EXAMPLE.COM

12/23/2014 10:59:54    12/24/2014 10:59:49    krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM
```




The new service can be viewed using **ipa service-find**.

```
# ipa service-find HTTP/broker.interop.example.com@INTEROP.EXAMPLE.COM
-----
1 service matched
-----
Principal: HTTP/broker.interop.example.com@INTEROP.EXAMPLE.COM
Keytab: False
Managed by: broker.interop.example.com
-----
Number of entries returned 1
-----
```

Note: The previous steps can also be done from any IdM client that has the **ipa-admintools** package installed.

2. Create broker HTTP keytabs.

Keytabs provide secure access to the broker web services through the use of Kerberos principals and an encrypted copy of the principal's key.

- a) Create an HTTP keytab on the first OpenShift broker (**broker1**) host using the IdM *admin* Kerberos principal as follows.

```
# hostname
broker1.interop.example.com

# kinit admin
Password for admin@INTEROP.EXAMPLE.COM: *****

# ipa-getkeytab -s idm-srv1.interop.example.com \
  -p HTTP/broker.interop.example.com@INTEROP.EXAMPLE.COM \
  -k /var/www/openshift/broker/httpd/conf.d/http.keytab
Keytab successfully retrieved and stored in:
/var/www/openshift/broker/httpd/conf.d/http.keytab

# chown apache:apache /var/www/openshift/broker/httpd/conf.d/http.keytab
# ll /var/www/openshift/broker/httpd/conf.d/http.keytab
-rw----- . 1 apache apache 386 Dec 23 14:16
/var/www/openshift/broker/httpd/conf.d/http.keytab
```

- b) Copy the keytab to the other broker hosts (**broker2**, **broker3**).

```
# scp -p /var/www/openshift/broker/httpd/conf.d/http.keytab broker2: \
/var/www/openshift/broker/httpd/conf.d/http.keytab

# scp -p /var/www/openshift/broker/httpd/conf.d/http.keytab broker3: \
/var/www/openshift/broker/httpd/conf.d/http.keytab
```



3. Configure OpenShift broker and console to authenticate using Kerberos.

OpenShift clients communicate to OpenShift brokers through either the web interface (aka - *console*) or the `rhc` client (aka - *broker*) command line interface. The default authentication method for both uses `htpasswd` and a local file. Change both to use **Kerberos** as follows.

- a) Install the Apache `mod_auth_kerb` package on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# hostname  
broker1.interop.example.com  
  
# yum install mod_auth_kerb
```

- b) Create a backup copy of the OpenShift *broker* remote authentication user configuration file (`/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf`) on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# cp -p /var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf{,.orig}
```

This file loads authentication modules and contains the directives for **rhc** client access to the OpenShift brokers.

- c) Edit the file and replace it with the contents shown below. Ensure the Kerberos service name (**KrbServiceName**) is set to the correct OpenShift broker host name. Kerberos realm (**KrbAuthRealms**) and keytab (**Krb5KeyTab**) should match what is highlighted. For support of cross-realm Kerberos trusts (**IS 5**), also include the name of the Active Directory Kerberos realm to (**KrbAuthRealms**). Perform this step on each OpenShift broker host (**broker1**, **broker2**, **broker3**). Alternatively, the file can be edited on one host and copied to the remaining broker hosts.

```
# Provided by the mod_auth_kerb package  
LoadModule auth_basic_module modules/mod_auth_basic.so  
LoadModule authz_user_module modules/mod_authz_user.so  
LoadModule auth_kerb_module modules/mod_auth_kerb.so  
<Location /broker>  
  AuthName "OpenShift broker API"  
  AuthType Kerberos  
  KrbMethodNegotiate On  
  KrbMethodK5Passwd On  
  # The KrbLocalUserMapping enables conversion to local users, using  
  # auth_to_local rules in /etc/krb5.conf. By default it strips the  
  # @REALM part. See krb5.conf(5) for details how to set up specific rules.  
  KrbLocalUserMapping On  
  KrbServiceName HTTP/broker.interop.example.com  
  KrbAuthRealms INTEROP.EXAMPLE.COM INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM  
  Krb5KeyTab /var/www/openshift/broker/httpd/conf.d/http.keytab  
  require valid-user
```



```
# Broker handles auth tokens
SetEnvIfNoCase Authorization Bearer passthrough

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

# Old-style auth keys are POSTed as parameters. The deployment
registration
# and snapshot-save use this.
BrowserMatchNoCase ^OpenShift passthrough
# Older-style auth keys are POSTed in a header. The Jenkins cartridge
does this.
SetEnvIf broker_auth_key "[A-Za-z0-9+/=]+$" passthrough=1

<IfVersion >= 2.4>
    Require env passthrough
</IfVersion>
<IfVersion < 2.4>
    Allow from env=passthrough
</IfVersion>

<IfVersion < 2.4>
    Order Deny,Allow
    Deny from all
    Satisfy any
</IfVersion>
</Location>

# The following APIs do not require auth:
#
# /api
# /environment
# /cartridges
# /quickstarts
#
# We want to match requests in the form of:
#
# /api
# /api.json
# /api/
#
# But not:
#
# /api_with_auth
<LocationMatch ^/broker/rest/(api|environment|cartridges|quickstarts)
(\.\w+|/?|/.*)$>
    <IfVersion >= 2.4>
        Require all granted
    </IfVersion>
    <IfVersion < 2.4>
        Allow from all
```



```
</IfVersion>
</LocationMatch>
```

- d) Create a backup copy of the OpenShift *console* remote authentication user configuration file (`/var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf`) on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# cp -p /var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf{,.orig}
```

This file loads authentication modules and contains the directives for console access to the OpenShift brokers.

- e) Edit the file and replace it with the contents shown below. Ensure the Kerberos service name (**KrbServiceName**) is set to the correct OpenShift broker host name. Kerberos realm (**KrbAuthRealms**) and keytab (**Krb5KeyTab**) should match what is highlighted. Note that the same keytab created in step 2a) is used for both broker API and console. For support of cross-realm Kerberos trusts (**IS 5**), also include the name of the Active Directory Kerberos realm to (**KrbAuthRealms**). Perform this step on each OpenShift broker host (**broker1**, **broker2**, **broker3**). Alternatively, the file can be edited on one host and copied to the remaining broker hosts.

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule auth_kerb_module modules/mod_auth_kerb.so

# 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 Developer Console"
  AuthType Kerberos
  KrbMethodNegotiate On
  KrbMethodK5Passwd On
  # The KrbLocalUserMapping enables conversion to local users, using
  # auth_to_local rules in /etc/krb5.conf. By default it strips the
  # @REALM part. See krb5.conf(5) for details how to set up specific rules.
  KrbLocalUserMapping On
  KrbServiceName HTTP/broker.interop.example.com
  KrbAuthRealms INTEROP.EXAMPLE.COM INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM
  Krb5KeyTab /var/www/openshift/broker/httpd/conf.d/http.keytab
  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>
```

4. Restart the *broker* and *console* services.

On each OpenShift broker host (**broker1**, **broker2**, **broker3**) restart the *broker* and *console* services.

```
# service openshift-broker restart
Stopping openshift-broker: [ OK ]
Starting openshift-broker: [ OK ]
```

```
# service openshift-console restart
Stopping openshift-console: [ OK ]
Starting openshift-console:
```

5. Configure **nsupdate** plugin.

a) Create a backup copy of the configuration file `/etc/openshift/plugins.d/openshift-origin-dns-nsupdate.conf` on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# cp -p /etc/openshift/plugins.d/openshift-origin-dns-nsupdate.conf{,.orig}
```

The **nsupdate** plugin is a module that facilitates the updating of dynamic DNS zones without the need to edit zone files or restart the DNS server.

b) Edit the file and replace it with the contents shown below. Ensure the IP address of the first IdM server (**BIND_SERVER**), domain name (**BIND_ZONE**) and Kerberos principal (**BIND_KRB_PRINCIPAL**) are correct. The keytab (**BIND_KRB_KEYTAB**) is configured after the DNS services are created and the zones modified for dynamic DNS.

broker1

```
BIND_SERVER="10.19.140.101"
BIND_PORT=53
BIND_ZONE="interop.example.com"
BIND_KRB_PRINCIPAL="DNS/broker1.interop.example.com@INTEROP.EXAMPLE.COM"
BIND_KRB_KEYTAB="/etc/dns.keytab"
```

broker2

```
BIND_SERVER="10.19.140.101"
BIND_PORT=53
BIND_ZONE="interop.example.com"
BIND_KRB_PRINCIPAL="DNS/broker2.interop.example.com@INTEROP.EXAMPLE.COM"
BIND_KRB_KEYTAB="/etc/dns.keytab"
```



broker3

```
BIND_SERVER="10.19.140.101"
BIND_PORT=53
BIND_ZONE="interop.example.com"
BIND_KRB_PRINCIPAL="DNS/broker3.interop.example.com@INTEROP.EXAMPLE.COM"
BIND_KRB_KEYTAB="/etc/dns.keytab"
```

6. Create broker DNS services.

On the first IdM server (**idm-srv1**), create a DNS service for each OpenShift broker host (**broker1**, **broker2**, **broker3**).

```
# hostname
idm-srv1.interop.example.com

# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting      Expires            Service principal
01/09/2015 10:36:47 01/10/2015 10:36:36 HTTP/idm-srv1.interop.example.com
@INTEROP.EXAMPLE.COM
01/09/2015 10:36:40 01/10/2015 10:36:36 krbtgt/INTEROP.EXAMPLE.COM
@INTEROP.EXAMPLE.COM

# ipa service-add DNS/broker1.interop.example.com
-----
Added service "DNS/broker1.interop.example.com@INTEROP.EXAMPLE.COM"
-----
Principal: DNS/broker1.interop.example.com@INTEROP.EXAMPLE.COM
Managed by: broker1.interop.example.com

# ipa service-add DNS/broker2.interop.example.com
-----
Added service "DNS/broker2.interop.example.com@INTEROP.EXAMPLE.COM"
-----
Principal: DNS/broker2.interop.example.com@INTEROP.EXAMPLE.COM
Managed by: broker2.interop.example.com

# ipa service-add DNS/broker3.interop.example.com
-----
Added service "DNS/broker3.interop.example.com@INTEROP.EXAMPLE.COM"
-----
Principal: DNS/broker3.interop.example.com@INTEROP.EXAMPLE.COM
Managed by: broker3.interop.example.com
```

7. Modify DNS zone for dynamic DNS.

Modify the DNS zone to allow OpenShift broker hosts to dynamically register deployed applications within IdM. Perform this task on the first IdM server (**idm-srv1**).



```
# ipa dnszone-mod interop.example.com --dynamic-update=true --update-policy= \
"grant DNS\047\broker1.interop.example.com@INTEROP.EXAMPLE.COM wildcard * ANY; \
grant DNS\047\broker2.interop.example.com@INTEROP.EXAMPLE.COM wildcard * ANY; \
grant DNS\047\broker3.interop.example.com@INTEROP.EXAMPLE.COM wildcard * ANY;"
Zone name: interop.example.com
Authoritative nameserver: idm-srv1.interop.example.com.
Administrator e-mail address: hostmaster.interop.example.com.
SOA serial: 1420836997
SOA refresh: 3600
SOA retry: 900
SOA expire: 1209600
SOA minimum: 3600
BIND update policy:
grant DNS\047\broker1.interop.example.com@INTEROP.EXAMPLE.COM wildcard * ANY;
grant DNS\047\broker2.interop.example.com@INTEROP.EXAMPLE.COM wildcard * ANY;
grant DNS\047\broker3.interop.example.com@INTEROP.EXAMPLE.COM wildcard * ANY;
Active zone: TRUE
Dynamic update: TRUE
Allow query: any;
Allow transfer: none;
```

8. Create broker DNS keytabs.

Keytabs provide secure access to the broker web services through the use of Kerberos principals and an encrypted copy of the principal's key. Generate a DNS keytab using the **ipa-getkeytab** utility for each OpenShift broker host. Set the correct file ownership (**apache:apache**) and verify the keytab with Kerberos as follows.

broker1

```
# hostname
broker1.interop.example.com

# ipa-getkeytab -s idm-srv1.interop.example.com \
               -p DNS/broker1.interop.example.com \
               -k /etc/dns.keytab
Keytab successfully retrieved and stored in: /etc/dns.keytab

# chown apache:apache /etc/dns.keytab
# kinit -kt /etc/dns.keytab -p DNS/broker1.interop.example.com
# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: DNS/broker1.interop.example.com@INTEROP.EXAMPLE.COM

Valid starting    Expires          Service principal
01/09/15 14:28:06 01/10/15 14:28:06 krbtgt/INTEROP.EXAMPLE.COM
@INTEROP.EXAMPLE.COM
renew until 01/10/15 14:28:34
```



broker2

```
# hostname
broker2.interop.example.com

# ipa-getkeytab -s idm-srv1.interop.example.com \
               -p DNS/broker2.interop.example.com \
               -k /etc/dns.keytab
Keytab successfully retrieved and stored in: /etc/dns.keytab

# chown apache:apache /etc/dns.keytab
# kinit -kt /etc/dns.keytab -p DNS/broker2.interop.example.com
# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: DNS/broker2.interop.example.com@INTEROP.EXAMPLE.COM

Valid starting      Expires              Service principal
01/09/15 14:35:06  01/10/15 14:35:06  krbtgt/INTEROP.EXAMPLE.COM
@INTEROP.EXAMPLE.COM
    renew until 01/10/15 14:35:45
```

broker3

```
# hostname
broker3.interop.example.com

# ipa-getkeytab -s idm-srv1.interop.example.com \
               -p DNS/broker3.interop.example.com \
               -k /etc/dns.keytab
Keytab successfully retrieved and stored in: /etc/dns.keytab

# chown apache:apache /etc/dns.keytab
# kinit -kt /etc/dns.keytab -p DNS/broker3.interop.example.com
# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: DNS/broker3.interop.example.com@INTEROP.EXAMPLE.COM

Valid starting      Expires              Service principal
01/09/15 14:44:06  01/10/15 14:44:06  krbtgt/INTEROP.EXAMPLE.COM
@INTEROP.EXAMPLE.COM
    renew until 01/10/15 14:44:45
```

9. Restart the *broker* services.

Enable all changes by restarting the broker service on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# service openshift-broker restart
Stopping openshift-broker: [ OK ]
Starting openshift-broker: [ OK ]
```




10. Configure RHC Client.

Dynamic DNS is now fully configured and ready for use by RHC clients. Configure the **RHC** client by running **rhc setup** and specify the round-robin DNS broker name (**broker**) for the server (**--server**) name. Perform this step on the RHC client (**rhc1**).

```
$ hostname  
rhc1.interop.example.com
```

```
$ id  
uid=889000002(ose-dev1) gid=889000002 groups=889000002,1550200007(ose-ssh-  
users) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

```
$ kinit ose-dev1  
Password for ose-dev1@INTEROP.EXAMPLE.COM: *****
```

```
$ rhc setup --server=broker.interop.example.com  
OpenShift Client Tools (RHC) Setup Wizard
```

This wizard will help you upload your SSH keys, set your application namespace, and check that other programs like Git are properly installed.

The server's certificate is self-signed, which means that a secure connection can't be established to 'broker.interop.example.com'.

You may bypass this check, but any data you send to the server could be intercepted by others.

```
Connect without checking the certificate? (yes|no): yes  
Login to broker.interop.example.com: ose-dev1  
Password: *****
```

OpenShift can create and store a token on disk which allows to you to access the server without using your password. The key is stored in your home directory and should be kept secret. You can delete the key at any time by running 'rhc logout'.

```
Generate a token now? (yes|no) yes  
Generating an authorization token for this client ... lasts about 1 day
```

```
Saving configuration to /home/ose-dev1/.openshift/express.conf ... done
```

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

```
Created: /home/ose-dev1/.ssh/id_rsa.pub
```

Your public SSH key must be uploaded to the OpenShift server to access code.
Upload now? (yes|no) yes

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

```
Checking for git ... found git version 1.7.1
```



```
Checking common problems .. done
```

```
Checking for a domain ... none
```

Applications are grouped into domains - each domain has a unique name (called a namespace) that becomes part of your public application URL. You may create your first domain here or leave it blank and use 'rhc create-domain' later. You will not be able to create an application without completing this step.

Please enter a namespace (letters and numbers only) |<none>|: **oseidmtest**
Your domain 'oseidmtest' has been successfully created

```
Checking for applications ... none
```

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

Do-It-Yourself 0.1	rhc create-app <app name> diy-0.1
JBoss Enterprise App Platform 6	rhc create-app <app name> jbosseap-6
Jenkins Server	rhc create-app <app name> jenkins-1
Node.js 0.10	rhc create-app <app name> nodejs-0.10
PHP 5.3	rhc create-app <app name> php-5.3
PHP 5.4	rhc create-app <app name> php-5.4
Perl 5.10	rhc create-app <app name> perl-5.10
Python 2.6	rhc create-app <app name> python-2.6
Python 2.7	rhc create-app <app name> python-2.7
Python 3.3	rhc create-app <app name> python-3.3
Ruby 1.8	rhc create-app <app name> ruby-1.8
Ruby 1.9	rhc create-app <app name> ruby-1.9
Tomcat 6 (JBoss EWS 1.0)	rhc create-app <app name> jbossews-1.0
Tomcat 7 (JBoss EWS 2.0)	rhc create-app <app name> jbossews-2.0

You are using 0 of 100 total gears

The following gear sizes are available to you: small

Your client tools are now configured.

```
$ rhc domain show
```

```
Domain oseidmtest (owned by ose-dev1)
```

```
-----
```

```
Created:          4:05 PM
ID:               53d2b87d34fcf03a860000008
Allowed Gear Sizes: small
```

The domain oseidmtest exists but has no applications. You can use 'rhc create-app' to create a new application.

11. Verify RHC client.

Verify the **RHC** client by checking the domain connectivity and deploying a test application.

```
$ hostname
```

```
rhc1.interop.example.com
```



```
$ rhc domain show
```

```
Domain oseidmtest (owned by ose-dev1)
```

```
-----
Created:          3:17 PM
ID:               54b0375696f47a7135000008
Allowed Gear Sizes: small
Suffix:           interop.example.com
```

The domain oseidmtest exists but has no applications. You can use 'rhc create-app' to create a new application.

```
$ rhc create-app oseidmtest php-5.4
```

```
Application Options
```

```
-----
Domain:          oseidmtest
Cartridges:      php-5.4
Gear Size:       default
Scaling:         no
```

```
Creating application 'oseidmtest' ... done
```

```
Waiting for your DNS name to be available ... done
```

```
Initialized empty Git repository in /home/ose-dev1/oseidmtest/.git/
The authenticity of host 'oseidmtest-oseidmtest.interop.example.com
(10.19.140.23)' can't be established.
RSA key fingerprint is 4a:a3:3c:fc:5e:8b:9c:55:80:51:32:8b:16:b2:71:31.
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'oseidmtest-
oseidmtest.interop.example.com,10.19.140.23' (RSA) to the list of known
hosts.
```

```
Your application 'oseidmtest' is now available.
```

```
URL:             http://oseidmtest-oseidmtest.interop.example.com/
SSH to:           54b568c796f47aea12000001@oseidmtest-
oseidmtest.interop.example.com
Git remote: ssh://54b568c796f47aea12000001@oseidmtest-
oseidmtest.interop.example.com/~/.git/oseidmtest.git/
Cloned to:        /home/ose-dev1/oseidmtest
```

```
Run 'rhc show-app oseidmtest' for more details about your app.
```

```
$ rhc apps
```

```
oseidmtest @ http://oseidmtest-oseidmtest.interop.example.com/ (uuid:
54b568c796f47aea12000001)
```

```
-----
Domain:          oseidmtest
Created:         3:49 PM
Gears:           1 (defaults to small)
Git URL:         ssh://54b568c796f47aea12000001@oseidmtest-
```



```
oseidmtest.interop.example.com/~/.git/oseidmtest.git/
SSH:      54b568c796f47aea12000001@oseidmtest-
oseidmtest.interop.example.com
Deployment: auto (on git push)

php-5.4 (PHP 5.4)
-----
Gears: 1 small
```

You have access to 1 application.

12. Verify OpenShift.

Verify the OpenShift brokers using the **oo-accept-broker** utility from any broker host. The full OpenShift environment can be verified by running the **oo-diagnostics** utility from any broker or node host.

```
# hostname
broker1.interop.example.com

# oo-accept-broker
PASS

# oo-diagnostics
WARN: test_altered_package_owned_configs
      RPM package owned configuration files have been altered:
      /etc/auto.master.rpmnew
/etc/activemq/activemq.xml.rpmsave
/etc/httpd/conf.d/000002_openshift_origin_broker_servername.conf.rpmsave
/etc/openshift/broker.conf.rpmsave
/etc/openshift/console.conf.rpmnew
/etc/openshift/console.conf.rpmsave
/etc/openshift/express.conf.rpmsave
/etc/yum/pluginconf.d/rhnplugin.conf.rpmnew
/opt/rh/ruby193/root/etc/mcollective/client.cfg.rpmsave

      Ensure any package-owned configuration files which have been
      altered are accurate. This may require a manual merge of
      your previous alterations. Once you are comfortable with the merge,
      remove the reported .rpm* configuration file (or you will continue
      to see this warning each time you run the diagnostic test).

WARN: test_broker_certificate
Using a self-signed certificate for the broker
2 WARNINGS
NO ERRORS
```



13. Additional Verification and Troubleshooting Steps.

Below are several additional checks that can be run to verify broker and console access.

a) Obtain Kerberos ticket (**rhc1**).

```
$ hostname
rhc1.interop.example.com
$ kinit ose-dev1
Password for ose-dev1@INTEROP.EXAMPLE.COM:

$ klist
Ticket cache: FILE:/tmp/krb5cc_889000002_fnQShv
Default principal: ose-dev1@INTEROP.EXAMPLE.COM

Valid starting      Expires              Service principal
07/25/14 20:54:57   07/26/14 20:54:56   krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM
```

b) Test broker access to all broker hosts using Kerberos (**rhc1**).

```
$ curl -Ik --negotiate -u : https://broker.interop.example.com/broker/rest/domains
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:43:10 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift broker API"
Content-Type: text/html; charset=iso-8859-1
Connection: close

HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 15:43:10 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: Phusion Passenger (mod_rails/mod_rack) 3.0.21
X-OpenShift-Identity: ose-dev1
X-OpenShift-Identity-Id: 54b0367796f47a7135000001
X-OAuth-Scopes: session
X-UA-Compatible: IE=Edge,chrome=1
ETag: "e31c8c398954ba82752fc57b0769cf59"
Cache-Control: must-revalidate, private, max-age=0
X-Request-Id: 7ec295e802e2204c454ab7d913a36a09
X-Runtime: 0.017453
X-Rack-Cache: miss
Status: 200
Content-Type: application/json; charset=utf-8
Connection: close

$ curl -Ik --negotiate -u : https://broker1.interop.example.com/broker/rest/domains
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:45:45 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift broker API"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```



```
$ curl -Ik --negotiate -u : https://broker2.interop.example.com/broker/rest/domains
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:46:23 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift broker API"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

```
$ curl -Ik --negotiate -u : https://broker3.interop.example.com/broker/rest/domains
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:46:28 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift broker API"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

Notes:

- Must use the **--negotiate** flag to test if Kerberos/Apache is setup correctly.
- The **"-u :"** picks up the cached ticket after **kinit**.
- When using **--negotiate** two responses are returned if **broker** is setup correctly:
 1. **401**
 2. **200** or **302** ([console](#) link)

and only a **401** code should be returned for **broker1**, **broker2**, **broker3**

- c) Test [console](#) access to all broker hosts using Kerberos Ticket (**rhc1**).

```
$ curl -Ik --negotiate -u : https://broker.interop.example.com/console
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:52:34 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift Developer Console"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

```
HTTP/1.1 302 Found
Date: Thu, 15 Jan 2015 15:52:34 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: Phusion Passenger (mod_rails/mod_rack) 3.0.21
X-Frame-Options: SAMEORIGIN
X-UA-Compatible: IE=Edge,chrome=1
Cache-Control: no-cache, private
X-Request-Id: 19850c03d115c42b926a1b5bad0edef0
X-Runtime: 0.025662
X-Rack-Cache: miss
Location: https://broker.interop.example.com/console/applications
Status: 302
Content-Type: text/html; charset=utf-8
```



Connection: close

```
$ curl -Ik --negotiate -u : https://broker1.interop.example.com/console
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:52:50 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift Developer Console"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

```
$ curl -Ik --negotiate -u : https://broker2.interop.example.com/console
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:52:53 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift Developer Console"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

```
$ curl -Ik --negotiate -u : https://broker3.interop.example.com/console
HTTP/1.1 401 Authorization Required
Date: Thu, 15 Jan 2015 15:52:55 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate
WWW-Authenticate: Basic realm="OpenShift Developer Console"
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

This completes the tasks for integration scenario 1.



7.3.2 IS2: Centralized Management of SSH Keys (optional)

Integration Scenario 2 allows administrators to centrally manage all IdM client SSH keys within IdM.

Integration Scenario 2 Centralized Management of SSH Keys	
Components	<ul style="list-style-type: none">• OpenShift Enterprise 2.2• Identity Management in Red Hat Enterprise Linux 7
Description	<ul style="list-style-type: none">• Centrally manage any IdM client SSH keys within IdM - includes all OpenShift hosts, clients
Target User	<ul style="list-style-type: none">• Administrators, Developers, Operators
Use Case	<ul style="list-style-type: none">• All users interested in letting IdM centrally manage SSH keys
Configuration Files	<ul style="list-style-type: none">• n/a
OpenShift Services	<ul style="list-style-type: none">• n/a
IdM Services	<ul style="list-style-type: none">• LDAP
Notes	<ul style="list-style-type: none">• Kerberos must be previously configured on IdM client• Multiple keys can be uploaded by separating each key with a comma• Alternatively, keys can be loaded directly from the IdM console: https://idm-srv1.interop.example.com/ipa/ui/ https://idm-srv2.interop.example.com/ipa/ui/

Integration Tasks - Summary

The following sequence of steps are required to configure integration scenario 2:

1. Generate SSH key pair (**admin1**)
2. Upload SSH public key to IdM server (**admin1**)
3. Verify SSH key (**admin1**)



Integration Tasks - Details

Repeat steps 1-3 for each OpenShift host requiring centralized key management within IdM. Once the host key pair has been uploaded to IdM, the key can be used from all hosts within the IdM domain.

1. Generate SSH key pair.

As the user requiring the key (**ose-admin1**), obtain a Kerberos ticket then create an **ssh** key pair for that user (**ose-admin1**) by running the **ssh-keygen** utility.

```
$ hostname
admin1.interop.example.com

# su - ose-admin1
$ kinit ose-admin1
Password for ose-admin1@INTEROP.EXAMPLE.COM: *****

$ id
uid=889000001(ose-admin1) gid=889000001(ose-admins) groups=889000001(ose-admins)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ ssh-keygen -t rsa -C ose-admin1@interop.example.com
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ose-admin1/.ssh/id_rsa):
Created directory '/home/ose-admin1/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ose-admin1/.ssh/id_rsa.
Your public key has been saved in /home/ose-admin1/.ssh/id_rsa.pub.
14:cd:97:fd:0f:e0:c7:d4:f2:05:80:7c:c1:30:98:5c ose-admin1@interop.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      ..BE+0=.o |
|      +.=.* + o |
|      . + + +. |
|      . . + o |
|      S . . |
|      . |
|      |
|      |
+-----+

$ ls -la /home/ose-admin1/.ssh/id_rsa*
-rw-----. 1 ose-admin1 889000001 1679 Jan 15 10:57 /home/ose-admin1/.ssh/id_rsa
-rw-r--r--. 1 ose-admin1 889000001 412 Jan 15 10:57 /home/ose-admin1/.ssh/id_rsa.pub
```

The **ssh** key pair can be re-created at any time by re-running **ssh-keygen**.



2. Upload the SSH public key to IdM. Copy and paste the contents of the *user* public key file as input to the **ipa user-mod** utility.

```
$ cat /home/ose-admin1/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDAQDKCIBSmdDoAS7amQunscFygNA3E0809YxhVU1MsRqvfbw80Y85S
NYwY4TvlV3iD7y5WMD1cvgyV0buFcWhXA2rabGmAhmJm5fHAZGABPQ3QS49jGTAH1ErzkLG+BZuskrhHt
+hGpUpFsrqcBDNvwKQVoHUwAIGhQ6ZpY+/bp2bEahkMDoHX9G6B2yM4k0mbLMihCoM4wHi/uZxKVpAM6k
Q5y4r0XBIuyIiZ1KUNBss0lc9UHzw56Po2yY0ZgQZc0CHkrzEKD6PRsLcyGukWB8kdYAM8B0HEWALrir5
6ppaUyF3KETVJmbZ0bh6vvVEe6f8tQ0n8CY3NBj5DKBck5G5 ose-admin1@interop.example.com

$ ipa user-mod ose-admin1 --sshpubkey="ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDA00u60SrKfFvPTfEcjUoXXN0Wyg0w/nZAE0JiUdwqJPFdJm+
Dxk73k7EDUmZQfeAjJzJwJPWApfr1f8XLD+k0+SAFwe65lCV60mgecGYNBs2rS47fUKN0AYdwhpZv+kid
u+hfaIEECx+7vWN1TotQI8ChEkfL0DfbpVIt31VlUChPMCS9Sr2N2MF0Sf8dmRjHtU5Ki6jY3fyiImnH0
Vqz0qKXgrxgZtBJGW/lJ1voBdR9Nf0eWwPUo3pKNBYditMiRhJLyx0I05K7Jnxd2pskFbdJGeQXe0ZMX0
Muas7A1iSRpTp4ZHhk10aQkgX6BIEx9igvvJnfpGD4Z1XnWZ ose-admin1@interop.example.com"
-----
Modified user "ose-admin1"
-----
User login: ose-admin1
First name: OSE
Last name: Admin 1
Home directory: /home/ose-admin1
Login shell: /bin/bash
Email address: ose-admin1@interop.example.com
UID: 889000001
GID: 889000001
Account disabled: False
SSH public key: ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQDAQDKCIBSmdDoAS7amQunscFygNA3E0809YxhVU1MsRqvfbw80Y85S
NYwY4TvlV3iD7y5WMD1cvgyV0buFcWhXA2rabGmAhmJm5fHAZGABPQ3QS49jGTAH1ErzkLG+BZuskrhHt
+hGpUpFsrqcBDNvwKQVoHUwAIGhQ6ZpY+/bp2bEahkMDoHX9G6B2yM4k0mbLMihCoM4wHi/uZxKVpAM6k
Q5y4r0XBIuyIiZ1KUNBss0lc9UHzw56Po2yY0ZgQZc0CHkrzEKD6PRsLcyGukWB8kdYAM8B0HEWALrir5
6ppaUyF3KETVJmbZ0bh6vvVEe6f8tQ0n8CY3NBj5DKBck5G5
ose-admin1@interop.example.com

Password: True
Member of groups: ipausers, idm-ssh-users
Kerberos keys available: True
SSH public key fingerprint: 14:CD:97:FD:0F:E0:C7:D4:F2:05:80:7C:C1:30:98:5C
ose-admin1@interop.example.com (ssh-rsa)
```

Should the the key be incorrect or the wrong key uploaded, remove the existing key from IdM.

```
$ ipa user-mod ose-admin1 --sshpubkey=""
```

```
-----
Modified user "ose-admin1"
-----
```

```
User login: ose-admin1
First name: OSE
Last name: Admin 1
Home directory: /home/ose-admin1
```



```
Login shell: /bin/bash
Email address: ose-admin1@interop.example.com
UID: 889000001
GID: 889000001
Account disabled: False
Password: True
Member of groups: ipausers
Kerberos keys available: True
```

then re-run **ipa user-mod** utility with the correct public key.

3. Verify user SSH key.

From admin1 to broker1 (user = ose-admin1)

```
$ hostname
admin1.interop.example.com

$ id
uid=889000001(ose-admin1) gid=889000001(ose-admins) groups=889000001(ose-
admins) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ ssh broker1.interop.example.com
Creating home directory for ose-admin1.
Kickstarted on 2014-09-09

$ hostname
broker1.interop.example.com

$ id
uid=889000001(ose-admin1) gid=889000001(ose-admins) groups=889000001(ose-
admins) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ pwd
/home/ose-admin1

$ logout
Connection to broker1.interop.example.com closed.
```

From admin1 to node1 (user = ose-admin1)

```
$ hostname
admin1.interop.example.com

$ id
uid=889000001(ose-admin1) gid=889000001(ose-admins) groups=889000001(ose-
admins) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ ssh node1.interop.example.com
Creating home directory for ose-admin1.
Kickstarted on 2014-09-09

$ hostname
node1.interop.example.com
```



```
$ id
uid=889000001(ose-admin1) gid=889000001(ose-admins) groups=889000001(ose-
admins) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ pwd
/home/ose-admin1

$ logout
Connection to node1.interop.example.com closed.
```

This completes the tasks for integration scenario 2.



7.3.3 IS3: Centralized Management of UID's (optional)

Integration Scenario 3 allows administrators to centrally manage all IdM user account UID's within IdM.

Integration Scenario 3 Centralized Management of user ID's (UID)	
Components	<ul style="list-style-type: none">• OpenShift Enterprise 2.2• Identity Management in Red Hat Enterprise Linux 7
Description	<ul style="list-style-type: none">• Create new groups, user accounts - all are centrally managed within IdM
Target User	<ul style="list-style-type: none">• Administrators
Use Case	<ul style="list-style-type: none">• Centralized management of user accounts with a Linux domain
Configuration Files	<ul style="list-style-type: none">• n/a
OpenShift Services	<ul style="list-style-type: none">• n/a
IdM Services	<ul style="list-style-type: none">• LDAP• Kerberos
Notes	<ul style="list-style-type: none">• Kerberos must be previously configured on IdM hosts and clients• Steps are similar to those used during the deployment of the first IdM Server

Integration Tasks - Summary

The following sequence of steps are required to configure integration scenario 3:

1. Obtain Kerberos ticket (**idm-srv1**)
2. Create IdM group (**idm-srv1**)
3. Create IdM user (**idm-srv1**)
4. View IdM group and user (**idm-srv1**)
5. Verify IdM group and user (**admin1**)
6. Additional Verification and Troubleshooting Tips (**All hosts**)



Integration Tasks – Detail

Unless otherwise indicated, all steps can be performed from either **idm-srv1**, **idm-srv2** or **admin1**. For convenience, **idm-srv1** is shown here.

1. Obtain Kerberos ticket.

```
# hostname
idm-srv1.interop.example.com

# kinit admin
Password for admin@INTEROP.EXAMPLE.COM:

# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM

Valid starting          Expires              Service principal
01/15/2015 11:11:40    01/16/2015 11:11:18  HTTP/idm-srv1.interop.example.com
@INTEROP.EXAMPLE.COM
01/15/2015 11:11:21    01/16/2015 11:11:18  krbtgt/INTEROP.EXAMPLE.COM
@INTEROP.EXAMPLE.COM
```

2. Create an IdM group named *ose-operators*.

```
# ipa group-add ose-operators --desc="OpenShift Operators" --gid=889000004
-----
Added group "ose-operators"
-----
Group name: ose-operators
Description: OpenShift Operators
GID: 889000004
```

3. Create an IdM a user named *ose-oper1* and add it to the *ose-operators* group.

```
# ipa user-add ose-oper1 --first="OSE" --last="Oper 1"
--displayname="OpenShift Operator 1" --homedir="/home/ose-oper1"
--shell="/bin/bash" --uid=889000004 --gidnumber=889000004 --password
Password: *****
Enter Password again to verify: *****
-----
Added user "ose-oper1"
-----
User login: ose-oper1
First name: OSE
Last name: Oper 1
Full name: OSE Oper 1
Display name: OpenShift Operator 1
Initials: 00
Home directory: /home/ose-oper1
GECOS: OSE Oper 1
Login shell: /bin/bash
```



```
Kerberos principal: ose-oper1@INTEROP.EXAMPLE.COM
Email address: ose-oper1@interop.example.com
UID: 8890000004
GID: 8890000004
Password: True
Member of groups: ipausers
Kerberos keys available: True
```

Note: By default OpenShift node hosts use UID's in the range 1000-6999 for gears. For most environments this should not present an issue as the default ranges used by IdM are well outside this range. The default ranges can be set during server installation, using the **--idstart** and **--idmax** options.

If no range is set when the first IdM server is installed, a range of 200,000 IDs is randomly selected. There are 10,000 possible ranges. Selecting a random range from that number provides a high probability of non-conflicting IDs if two separate IdM domains are ever merged in the future.

For more information, consult the *Managing Unique UID and GID Number Assignments* chapter of the [Red Hat Enterprise Linux 7 - Linux Domain, Identity, Authentication and Policy Guide](#) on the Red Hat customer portal.

4. View the new IdM group and user.

```
# ipa group-show ose-operators
Group name: ose-operators
Description: OpenShift Operators
GID: 8890000004

# ipa user-show ose-oper1
User login: ose-oper1
First name: OSE
Last name: Oper 1
Home directory: /home/ose-oper1
Login shell: /bin/bash
Email address: ose-oper1@interop.example.com
UID: 8890000004
GID: 8890000004
Account disabled: False
Password: True
Member of groups: ipausers
Kerberos keys available: True
```

5. Verify the new IdM group and user.

```
$ hostname
admin1.interop.example.com

# id ose-oper1
```



```
uid=8890000004(ose-oper1) gid=8890000004 groups=8890000004

# su - ose-oper1
Creating home directory for ose-oper1.

$ id ose-oper1
uid=8890000004(ose-oper1) gid=8890000004(ose-operators) groups=8890000004(ose-operators) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

6. Additional Verification and Troubleshooting Tips (All hosts).

- a) If either **ssh** or **su** complain about the home directory:

```
su: warning: cannot change directory to /home/ose-oper1:
No such file or directory
```

then make sure that **oddjob-mkhomedir** has been installed and the **oddjob** service is running and enabled (on boot) for the target host:

Red Hat Enterprise Linux 6

```
# yum install oddjob-mkhomedir
# service oddjobd restart
Shutting down oddjobd: [FAILED]
Starting oddjobd: [ OK ]
# chkconfig oddjobd on
# chkconfig oddjobd --list
# oddjobd 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Red Hat Enterprise Linux 7

```
# yum install oddjob-mkhomedir
# systemctl restart oddjobd
# systemctl enable oddjobd
# systemctl status oddjobd
oddjobd.service - privileged operations for unprivileged applications
  Loaded: loaded (/usr/lib/systemd/system/oddjobd.service; enabled)
  Active: active (running) since Mon 2014-07-28 12:36:52 EDT; 26s ago
  Main PID: 23329 (oddjobd)
  CGroup: /system.slice/oddjobd.service
          └─23329 /usr/sbin/oddjobd -n -p /var/run/oddjobd.pid -t 300
```

- b) If the initial **ipa-client-install** was not done with **--enablemkhomedir** then it can be enabled later using:

```
# authconfig --enablemkhomedir --update
Stopping sssd: [ OK ]
Starting oddjobd: [ OK ]
```

and then restarting SSSD:



Red Hat Enterprise Linux 6

```
# service sssd restart
Shutting down oddjobd: [ OK ]
Starting oddjobd: [ OK ]
```

Red Hat Enterprise Linux 7

```
# systemctl restart sssd
```

c) If either **ssh** or **su** complain about the group:

```
# su - ose-oper1
id: cannot find name for group ID 889000001
```

then make sure the **SSSD** service is running and enabled to start on boot:

Red Hat Enterprise Linux 6

```
# service sssd restart
Stopping sssd: [ FAILED ]
Starting sssd: [ OK ]

# chkconfig --list sssd
sssd                                0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Red Hat Enterprise Linux 7

```
# systemctl restart sssd

# systemctl status sssd
sssd.service - System Security Services Daemon
  Loaded: loaded (/usr/lib/systemd/system/sss.service; enabled)
  Active: active (running) since Wed 2014-07-30 13:22:11 EDT; 15s ago
  Process: 17920 ExecStart=/usr/sbin/sss -D -f (code=exited,
status=0/SUCCESS)
  Main PID: 17921 (sss)
# systemctl enable sssd
ln -s '/usr/lib/systemd/system/sss.service' '/etc/systemd/system/multi-
user.target.wants/sss.service'
```

This completes the tasks for integration scenario 3.



7.3.4 IS4: IS4: Kerberized Access to Gears (optional)

Integration Scenario 4 provides an alternative, more secure method to using public ssh keys to access gears when RHC clients have previously been configured to use Kerberos.

Integration Scenario 4 Kerberized Access to Gears	
Components	<ul style="list-style-type: none">OpenShift Enterprise 2.2Identity Management in Red Hat Enterprise Linux 7
Description	<ul style="list-style-type: none">Enable access to applications via the rhc client and Kerberos
Target User	<ul style="list-style-type: none">Developers
Use Case	<ul style="list-style-type: none">Developers looking for an alternative, more secure method to using ssh public keys to access the gear that contains an application.
Configuration Files	<ul style="list-style-type: none"><code>\$HOME/.k5login</code> (of gear user on node running the application)
OpenShift Services	<ul style="list-style-type: none">n/a
IdM Services	<ul style="list-style-type: none">KerberosLDAP
Notes	<ul style="list-style-type: none">Kerberos must be previously configured on node hosts and the rhc client

Integration Tasks - Summary

The following sequence of steps are required to configure integration scenario 4:

1. Create Kerberos Principal SSH Key (**rhc1**)
2. Verify the Key (**rhc1**)
3. Obtain Kerberos Ticket (**rhc1**)
4. Verify access to gear (**rhc1**)



Integration Tasks - Details

1. Create Kerberos Principal SSH Key.

As the developer (**ose-dev1**) on the **RHC** client (**rhc1**), add a new Kerberos Principal ssh key type by specifying a name for the new key (**ose-dev1_KerbKey**), the sshkey type (**--type krb5-principal**) and Kerberos principal (**ose-dev1@INTEROP.EXAMPLE.COM**).

```
$ hostname
rhc1.interop.example.com

$ id
uid=889000002(ose-dev1) gid=889000002(ose-developers) groups=889000002(ose-developers) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

$ rhc sshkey add ose-dev1_KerbKey --type krb5-principal --content ose-dev1@INTEROP.EXAMPLE.COM
RESULT:
SSH key 'ose-dev1_KerbKey' has been added
```

Note: If the key should need to be removed, run the following from the **RHC** client:

```
$ rhc sshkey remove ose-dev1_KerbKey
Removing the key 'ose-dev1_KerbKey ... removed
```

2. Verify the Key.

Confirm the new key using **rhc sshkey list**.

```
$ rhc sshkey list
default (type: ssh-rsa)
-----
Fingerprint: 06:be:93:01:6b:90:e8:66:21:f2:a6:94:78:ca:d6:8f

ose-dev1_KerbKey (type: krb5-principal)
-----
Principal: ose-dev1@INTEROP.EXAMPLE.COM

You have 2 SSH keys associated with your account.
```

The first key (**default**) is the ssh public key created during the initial '**rhc setup**'. Remove this key to ensure the new Kerberos principal SSH key is used:

```
$ rhc sshkey remove default
Removing the key 'default ... removed

$ rhc sshkey list
ose-dev1_KerbKey (type: krb5-principal)
-----
Principal: ose-dev1@INTEROP.EXAMPLE.COM

You have 1 SSH keys associated with your account.
```



3. Obtain Kerberos Ticket.

```
$ kinit ose-dev1
Password for ose-dev1@INTEROP.EXAMPLE.COM: *****

$ klist
Ticket cache: FILE:/tmp/krb5cc_889000002
Default principal: ose-dev1@INTEROP.EXAMPLE.COM

Valid starting      Expires            Service principal
01/15/15 11:29:46  01/16/15 11:29:46  krbtgt/INTEROP.EXAMPLE.COM
@INTEROP.EXAMPLE.COM
        renew until 01/16/15 11:30:07
```

4. Verify Access to Gear.

As the developer (**ose-dev1**) on the **RHC** client (**rhc1**), verify access to the gear by running '**rhc ssh <AppName>**' as follows:

```
$ rhc ssh oseidmtest
Connecting to 54b568c796f47aea12000001@oseidmtest-
oseidmtest.interop.example.com ...
Kickstarted on 2014-09-09

*****
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://www.openshift.com/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 delete"
and recreate it
!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!

Type "help" for more info.

[oseidmtest.interop.example.com 54b568c796f47aea12000001]\> ls -la .k5login
-rw-r--r--. 1 root 54b568c796f47aea12000001 77 Jan 15 11:28 .k5login
[oseidmtest.interop.example.com 54b568c796f47aea12000001]\> cat .k5login

# id: 54b0367796f47a7135000001-ose-dev1_KerbKey
ose-dev1@INTEROP.EXAMPLE.COM
[oseidmtest.interop.example.com 54b568c796f47aea12000001]\>
```

This completes the tasks for integration scenario 4.



7.3.5 IS5: Cross-realm Kerberos Trust (optional)

Integration Scenario 5 creates multi-master cross-realm Kerberos trust between IdM and Active Directory to utilize existing user/developer accounts within Active Directory.

Integration Scenario 5 Cross-realm Kerberos Trust	
Components	<ul style="list-style-type: none">• OpenShift Enterprise 2.2• Identity Management in Red Hat Enterprise Linux 7
Description	<ul style="list-style-type: none">• Configure cross-realm Kerberos trust between IdM and Active Directory. Multi-master IdM server configurations (server, replica) provide the highest level of IdM service availability.
Target User	<ul style="list-style-type: none">• OpenShift developers, users with existing Active Directory accounts.
Use Case	<ul style="list-style-type: none">• Environments looking to use existing Active Directory accounts on OpenShift hosts for <u>non-RHC</u> operations
Configuration Files	<ul style="list-style-type: none">• <i>/etc/krb5.conf</i>
OpenShift Services	<ul style="list-style-type: none">• n/a
IdM Services	<ul style="list-style-type: none">• Kerberos• LDAP
Notes	<ul style="list-style-type: none">• OpenShift RHC client has limited Kerberos support

Integration Tasks - Summary

The following sequence of steps are required to configure integration scenario 5:

1. Install **ipa-server-trust-ad** package (**idm-srv1, idm-srv2**)
2. Run **ipa-adtrust-install** (**idm-srv1, idm-srv2**)
3. Re-configure Firewall Ports for AD Trust (**idm-srv1, idm-srv2**)
4. Add a DNS forwarder for the Active Directory domain (**idm-srv1**)
5. Modify Kerberos domain and realm mappings (**idm-srv1, idm-srv2**)
6. Restart KDC (**idm-srv1, idm-srv2**)
7. Run **ipa trust-add** (**idm-srv1**)
8. Create IdM external and IdM POSIX groups (**idm-srv1**)
9. Add AD group to IdM external group (**idm-srv1**)
10. Add IdM external group to IdM POSIX group (**idm-srv1**)
11. Verify access from both IdM servers (**idm-srv1, idm-srv2**)



12. Disable Kerberos local user mappings (**broker1, broker2, broker3**)
13. Run RHC setup using Active Directory user account (**rhc1**)
14. Verify RHC client (**rhc1**)

Integration Tasks - Details

1. Install the **ipa-server-trust-ad** package on both the IdM server and replica (**idm-srv1, idm-srv2**).

```
# yum install ipa-server-trust-ad
Loaded plugins: langpacks, product-id, subscription-manager
Installed:
  ipa-server-trust-ad.x86_64 0:3.3.3-28.el7

...output abbreviated...

Dependency Installed:
  iniparser.x86_64 0:3.1-5.el7  libsss_nss_idmap-python.x86_64 0:1.11.2-
68.el7_0.5 pyldb.x86_64 0:1.1.16-4.el7 python-tdb.x86_64
0:1.2.12-3.el7 python-tevent.x86_64 0:0.9.18-6.el7
  samba.x86_64 0:4.1.1-35.el7_0 samba-python.x86_64 0:4.1.1-35.el7_0
samba-winbind.x86_64 0:4.1.1-35.el7_0 samba-winbind-modules.x86_64 0:4.1.1-
35.el7_0

Complete!
```

2. Run the **ipa-adtrust-install** utility on both IdM servers (**idm-srv1, idm-srv2**).

```
# ipa-adtrust-install
The log file for this installation can be found in /var/log/ipaserver-
install.log
=====
This program will setup components needed to establish trust to AD domains
for the IPA Server.

This includes:
  * Configure Samba
  * Add trust related objects to IPA LDAP server

To accept the default shown in brackets, press the Enter key.

WARNING: The smb.conf already exists. Running ipa-adtrust-install will
break your existing samba configuration.

Do you wish to continue? [no]: yes
Do you want to enable support for trusted domains in Schema Compatibility
plugin?
This will allow clients older than SSSD 1.9 and non-Linux clients to work
with trusted users.

Enable trusted domains support in slapi-nis? [no]: yes
```



Configuring cross-realm trusts for IPA server requires password for user 'admin'.

This user is a regular system account used for IPA server administration.

admin password: *****

Enter the NetBIOS name for the IPA domain.

Only up to 15 uppercase ASCII letters and digits are allowed.

Example: EXAMPLE.

NetBIOS domain name [INTEROP]: <...**accept default**...>

WARNING: 11 existing users or groups do not have a SID identifier assigned. Installer can run a task to have ipa-sidgen Directory Server plugin generate the SID identifier for all these users. Please note, the in case of a high number of users and groups, the operation might lead to high replication traffic and performance degradation. Refer to ipa-adtrust-install(1) man page for details.

Do you want to run the ipa-sidgen task? [no]: <...**accept default**...>

The following operations may take some minutes to complete.

Please wait until the prompt is returned.

Configuring CIFS

[1/20]: stopping smbd

[2/20]: creating samba domain object

[3/20]: creating samba config registry

[4/20]: writing samba config file

[5/20]: adding cifs Kerberos principal

[6/20]: check for cifs services defined on other replicas

[7/20]: adding cifs principal to S4U2Proxy targets

[8/20]: adding admin(group) SIDs

[9/20]: adding RID bases

[10/20]: updating Kerberos config

'dns_lookup_kdc' already set to 'true', nothing to do.

[11/20]: activating CLDAP plugin

[12/20]: activating sidgen plugin and task

[13/20]: activating extdom plugin

[14/20]: configuring smbd to start on boot

[15/20]: adding special DNS service records

[16/20]: enabling trusted domains support for older clients via Schema

Compatibility plugin

[17/20]: restarting Directory Server to take MS PAC and LDAP plugins changes into account

[18/20]: adding fallback group

[19/20]: setting SELinux booleans

[20/20]: starting CIFS services

Done configuring CIFS.

=====
Setup complete

You must make sure these network ports are open:



```
TCP Ports:
* 138: netbios-dgm
* 139: netbios-ssn
* 445: microsoft-ds
UDP Ports:
* 138: netbios-dgm
* 139: netbios-ssn
* 389: (C)LDAP
* 445: microsoft-ds
```

Additionally you have to make sure the IPA LDAP server is not reachable by any domain controller in the Active Directory domain by closing down the following ports for these servers:

```
TCP Ports:
* 389, 636: LDAP/LDAPS
```

You may want to choose to REJECT the network packets instead of DROPing them to avoid timeouts on the AD domain controllers.

```
=====
```

3. Re-configure Firewall Ports for AD Trust

The firewall ports must be opened on both IdM servers (**idm-srv1**, **idm-srv2**) for a select set of network services and ports from the Active Directory server as shown in **Table 7.3.5: AD Trust - Ports**.

Host	Port	Protocol	Service/Description
idm-srv1 idm-srv2 (ipa-ad-trust-chain)	138	TCP, UDP	netbios-dgm (NetBIOS Datagram)
	139	TCP, UDP	netbios-ssn (NetBIOS)
	445	TCP, UDP	microsoft-ds (Directory Server)
	389	UDP	LDAP

Table 7.3.5: AD Trust - Ports

- a) Create a new firewall chain (**ipa-ad-trust-chain**) with the appropriate firewall rules as follows:

```
# firewall-cmd --permanent --direct --add-chain ipv4 filter ipa-ad-trust-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT 0 -m conntrack
--ctstate NEW -j ipa-ad-trust-chain
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto tcp --destination-port 138 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto udp --destination-port 138 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto tcp --destination-port 139 --jump ACCEPT
```




```
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto udp --destination-port 139 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto tcp --destination-port 445 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto udp --destination-port 445 --jump ACCEPT
success
# firewall-cmd --permanent --direct --add-rule ipv4 filter ipa-ad-trust-chain 0
--proto udp --destination-port 389 --jump ACCEPT
success

# firewall-cmd --reload
success

# firewall-cmd --permanent --direct --get-all-rules
ipv4 filter INPUT 0 -m conntrack --ctstate NEW -j ipa-server-chain
ipv4 filter INPUT 0 -m conntrack --ctstate NEW -j ipa-ad-trust-chain
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 80 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 443 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 389 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 636 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 88 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 464 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 53 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto udp --destination-port 123 --jump ACCEPT
ipv4 filter ipa-server-chain 0 --proto tcp --destination-port 7389 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto tcp --destination-port 138 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto udp --destination-port 138 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto tcp --destination-port 139 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto udp --destination-port 139 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto tcp --destination-port 445 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto udp --destination-port 445 --jump
ACCEPT
ipv4 filter ipa-ad-trust-chain 0 --proto udp --destination-port 389 --jump
ACCEPT
```

4. Add a DNS conditional forwarder for the Active Directory domain to IdM. DNS lookup requests for hosts in the AD domain (**interop-ad.cloud.lab.eng.bos.redhat.com**) are forwarded to the AD server (**10.19.142.101**). Run this from one IdM server (**idm-srv1**).

```
# kinit admin
```



```
# ipa dnszone-add interop-ad.cloud.lab.eng.bos.redhat.com --name-  
server=win2012-srv1.interop-ad.cloud.lab.eng.bos.redhat.com --admin-  
email='hostmaster@interop-ad.cloud.lab.eng.bos.redhat.com' --force  
--forwarder=10.19.142.101 --forward-policy=only --ip-address=10.19.142.101  
Zone name: interop-ad.cloud.lab.eng.bos.redhat.com  
Authoritative nameserver: win2012-srv1.interop-  
ad.cloud.lab.eng.bos.redhat.com  
Administrator e-mail address: hostmaster.interop-  
ad.cloud.lab.eng.bos.redhat.com.  
SOA serial: 1406846543  
SOA refresh: 3600  
SOA retry: 900  
SOA expire: 1209600  
SOA minimum: 3600  
BIND update policy: grant INTEROP.EXAMPLE.COM krb5-self * A; grant  
INTEROP.EXAMPLE.COM krb5-self * AAAA; grant INTEROP.EXAMPLE.COM krb5-self *  
SSHFP;  
Active zone: TRUE  
Dynamic update: FALSE  
Allow query: any;  
Allow transfer: none;  
Zone forwarders: 10.19.142.101  
Forward policy: only
```

Verify the Windows server and domain are accessible:

```
# ping -c3 win2012-srv1  
PING win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101) 56(84) bytes  
of data.  
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):  
icmp_seq=1 ttl=128 time=0.678 ms  
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):  
icmp_seq=2 ttl=128 time=0.346 ms  
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):  
icmp_seq=3 ttl=128 time=0.485 ms  
  
--- win2012-srv1.cloud.lab.eng.bos.redhat.com ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 0.346/0.503/0.678/0.136 ms  
# ping -c3 interop-ad  
PING interop-ad.cloud.lab.eng.bos.redhat.com (10.19.142.101) 56(84) bytes  
of data.  
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):  
icmp_seq=1 ttl=128 time=0.355 ms  
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):  
icmp_seq=2 ttl=128 time=0.805 ms  
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):  
icmp_seq=3 ttl=128 time=0.606 ms  
  
--- interop-ad.cloud.lab.eng.bos.redhat.com ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 0.355/0.588/0.805/0.186 ms  
# ping -c3 interop-ad.cloud.lab.eng.bos.redhat.com
```



```
PING interop-ad.cloud.lab.eng.bos.redhat.com (10.19.142.101) 56(84) bytes
of data.
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):
icmp_seq=1 ttl=128 time=0.627 ms
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):
icmp_seq=2 ttl=128 time=0.716 ms
64 bytes from win2012-srv1.cloud.lab.eng.bos.redhat.com (10.19.142.101):
icmp_seq=3 ttl=128 time=0.596 ms

--- interop-ad.cloud.lab.eng.bos.redhat.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.596/0.646/0.716/0.054 ms
```

5. Modify the Kerberos domain and realm mappings.

- a) On the first IdM server (**idm-srv1**), edit the `[realms]` section of the Kerberos configuration file (`/etc/krb5.conf`) as highlighted in **bold** below. The ***auth_to_local*** rules converts the case differences between the Kerberos principal names (uppercase) and the POSIX (lowercase) equivalents. A new entry is also created for the Active Directory realm (***INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM***):

```
includedir /var/lib/sss/pubconf/krb5.include.d/

[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = INTEROP.EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = true
rdns = false
ticket_lifetime = 24h
forwardable = yes
default_ccache_name = KEYRING:persistent:%{uid}

[realms]
INTEROP.EXAMPLE.COM = {
    kdc = idm-srv1.interop.example.com:88
    master_kdc = idm-srv1.interop.example.com:88
    admin_server = idm-srv1.interop.example.com:749
    default_domain = interop.example.com
    pkinit_anchors = FILE:/etc/ipa/ca.crt
    auth_to_local = RULE:[1:$1@$0](^.*@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM$)
s/@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM/
@interop-ad.cloud.lab.eng.bos.redhat.com/
    auth_to_local = DEFAULT
}

INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM = {
    kdc = win2012-srv1.cloud.lab.eng.bos.redhat.com
    master_kdc = win2012-srv1.cloud.lab.eng.bos.redhat.com
    admin_server = win2012-srv1.cloud.lab.eng.bos.redhat.com
```



```
}  
  
[domain_realm]  
    .interop.example.com = INTEROP.EXAMPLE.COM  
    interop.example.com = INTEROP.EXAMPLE.COM  
  
    .interop-ad.cloud.lab.eng.bos.redhat.com = INTEROP-AD.CLOUD.LAB.ENG.BOS.  
REDHAT.COM  
    interop-ad.cloud.lab.eng.bos.redhat.com = INTEROP-AD.CLOUD.LAB.ENG.BOS.  
REDHAT.COM  
  
[dbmodules]  
    INTEROP.EXAMPLE.COM = {  
        db_library = ipadb.so  
    }
```

- b) On the IdM replica server (**idm-srv2**), the equivalent sections should be adjusted accordingly as highlighted in **bold**:

```
includedir /var/lib/sss/pubconf/krb5.include.d/  
  
[logging]  
    default = FILE:/var/log/krb5libs.log  
    kdc = FILE:/var/log/krb5kdc.log  
    admin_server = FILE:/var/log/kadmind.log  
  
[libdefaults]  
    default_realm = INTEROP.EXAMPLE.COM  
    dns_lookup_realm = false  
    dns_lookup_kdc = true  
    rdns = false  
    ticket_lifetime = 24h  
    forwardable = yes  
    default_ccache_name = KEYRING:persistent:%{uid}  
  
[realms]  
    INTEROP.EXAMPLE.COM = {  
        kdc = idm-srv2.interop.example.com:88  
        master_kdc = idm-srv2.interop.example.com:88  
        admin_server = idm-srv2.interop.example.com:749  
        default_domain = interop.example.com  
        pkinit_anchors = FILE:/etc/ipa/ca.crt  
        auth_to_local = RULE:[1:$1@$0](^.*@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM$)  
s/@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM/  
@interop-ad.cloud.lab.eng.bos.redhat.com/  
        auth_to_local = DEFAULT  
    }  
  
INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM = {  
    kdc = win2012-srv1.cloud.lab.eng.bos.redhat.com  
    master_kdc = win2012-srv1.cloud.lab.eng.bos.redhat.com  
    admin_server = win2012-srv1.cloud.lab.eng.bos.redhat.com  
    }
```



```
[domain_realm]
.interop.example.com = INTEROP.EXAMPLE.COM
interop.example.com = INTEROP.EXAMPLE.COM

.interop-ad.cloud.lab.eng.bos.redhat.com = INTEROP-AD.CLOUD.LAB.ENG.BOS.
REDHAT.COM
    interop-ad.cloud.lab.eng.bos.redhat.com = INTEROP-AD.CLOUD.LAB.ENG.BOS.
REDHAT.COM

[dbmodules]
INTEROP.EXAMPLE.COM = {
    db_library = ipadb.so
}
```

On all clients (e.g. **rhc1**) requiring Kerberos single sign-on (SSSO) with an Active Directory user account, the ***auth_to_local*** rules should also be added to the **[realms]** section of the Kerberos configuration file:

```
[realms]
INTEROP.EXAMPLE.COM = {
    pkinit_anchors = FILE:/etc/ipa/ca.crt
    auth_to_local = RULE:[1:$1@$0](^.*@INTEROP-
AD.CLOUD.LAB.ENG.BOS.REDHAT.COM$)s/@INTEROP-
AD.CLOUD.LAB.ENG.BOS.REDHAT.COM/@interop-ad.cloud.lab.eng.bos.redhat.com/
    auth_to_local = DEFAULT
}
```

No other sections of the client Kerberos configuration file need to be changed.

- Restart the KDC server on both IdM servers (**idm-srv1**, **idm-srv2**).

```
# systemctl restart krb5kdc
# systemctl restart sssd
```

- Run **ipa trust-add** on the first IdM server (**idm-srv1**).

Specify the trust type as Active Directory (**--type=ad**) followed by the name of the AD domain (**INTEROP-AD**), the AD Administrator account (**--admin Administrator**) and password (**--passwd**). This step should only be run on the first IdM server (**idm-srv1**).

```
# kinit admin
Password for admin@INTEROP.EXAMPLE.COM:

# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INTEROP.EXAMPLE.COM
```

```
Valid starting      Expires            Service principal
```



```
07/31/2014 19:52:03 08/01/2014 19:51:58
krbtgt/INTEROP.EXAMPLE.COM@INTEROP.EXAMPLE.COM
```

```
# ipa trust-add --type=ad interop-ad.cloud.lab.eng.bos.redhat.com --admin
Administrator --password
```

```
-----
Added Active Directory trust for realm "interop-ad.cloud.lab.eng.bos.
redhat.com"
```

```
-----
Realm name: interop-ad.cloud.lab.eng.bos.redhat.com
Domain NetBIOS name: INTEROP-AD
Domain Security Identifier: S-1-5-21-1146061590-2662362510-3882198490
SID blacklist incoming: S-1-0, S-1-1, S-1-2, S-1-3, S-1-5-1, S-1-5-2, S-
1-5-3, S-1-5-4, S-1-5-5, S-1-5-6, S-1-5-7, S-1-5-8, S-1-5-9, S-1-5-10, S-1-
5-11, S-1-5-12,
                        S-1-5-13, S-1-5-14, S-1-5-15, S-1-5-16, S-1-5-17,
S-1-5-18, S-1-5-19, S-1-5-20
SID blacklist outgoing: S-1-0, S-1-1, S-1-2, S-1-3, S-1-5-1, S-1-5-2, S-
1-5-3, S-1-5-4, S-1-5-5, S-1-5-6, S-1-5-7, S-1-5-8, S-1-5-9, S-1-5-10, S-1-
5-11, S-1-5-12,
                        S-1-5-13, S-1-5-14, S-1-5-15, S-1-5-16, S-1-5-17,
S-1-5-18, S-1-5-19, S-1-5-20
Trust direction: Two-way trust
Trust type: Active Directory domain
Trust status: Established and verified
```

Verify the new trusted domain:

```
# ipa trustdomain-find interop-ad.cloud.lab.eng.bos.redhat.com
Domain name: interop-ad.cloud.lab.eng.bos.redhat.com
Domain NetBIOS name: INTEROP-AD
Domain Security Identifier: S-1-5-21-1146061590-2662362510-3882198490
Domain enabled: True
-----
Number of entries returned 1
-----
```

8. Create IdM external and IdM POSIX groups on the first IdM server (**idm-srv1**).

a) The IdM external group (***ad-ose-users-external1***) is used for mapping OpenShift users in the Active Directory domain:

```
# ipa group-add --desc='AD OpenShift users - external group' ad-ose-users-
external --external
-----
Added group "ad-ose-users-external"
-----
Group name: ad-ose-users-external
Description: AD OpenShift users - external group
```

b) The IdM POSIX group (***ad-ose-users-posix***) is used by IdM to manage the policies of the OpenShift users in the Active Directory domain:



```
# ipa group-add --desc='AD OpenShift users - POSIX group' ad-ose-users-
posix
-----
Added group "ad-ose-users-posix"
-----
Group name: ad-ose-users-posix
Description: AD OpenShift users - POSIX group
GID: 1550200005
```

c) Verify both groups:

```
# ipa group-show ad-ose-users-external
Group name: ad-ose-users-external
Description: AD OpenShift users - external group

# ipa group-show ad-ose-users-posix
Group name: ad-ose-users-posix
Description: AD OpenShift users - POSIX group
GID: 1550200005
```

d) Verify the AD domain group is accessible:

```
# getent group 'INTEROP-AD\Domain Users'
domain users@interop-ad.cloud.lab.eng.bos.redhat.com:*:645800513:
```

9. Add the AD domain group to the IdM external group on the first IdM server (**idm-srv1**).

```
# ipa group-add-member ad-ose-users-external --external 'INTEROP-AD\Domain Users'
[member user]:
[member group]:
Group name: ad-ose-users-external
Description: AD OpenShift users - external group
External member: S-1-5-21-1146061590-2662362510-3882198490-513
-----
Number of members added 1
-----
```

10. Add the IdM external group to the IdM POSIX group on the first IdM server (**idm-srv1**).

```
# ipa group-add-member ad-ose-users-posix --groups ad-ose-users-external
Group name: ad-ose-users-posix
Description: AD OpenShift users - POSIX group
GID: 1550200005
Member groups: ad-ose-users-external
-----
Number of members added 1
-----
```

11. Verify access from both IdM servers using an existing AD user account (**ad-user1**).



Specifying the **-K** parameter is optional but enables the delegation of Kerberos tickets to allow the ticket to be re-used on the remote host. The POSIX name is required (**interop-ad.cloud.lab.eng.bos.redhat.com**) for login using the AD account.

IdM Server to RHC Client

```
# hostname
idm-srv1.interop.example.com

# ssh -l ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com rhc1
ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com's password: *****
Creating home directory for ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com.
Kickstarted on 2014-09-25

[ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com@rhc1 ~]$ pwd
/home/interop-ad.cloud.lab.eng.bos.redhat.com/ad-user1

[ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com@rhc1 ~]$ hostname
rhc1.interop.example.com
```

IdM Replica to Broker Host

```
# hostname
idm-srv2.interop.example.com

# ssh -K -l ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com broker1
ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com's password: *****
Creating home directory for ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com.
Kickstarted on 2014-09-09

[ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com@rhc1 ~]$ pwd
/home/interop-ad.cloud.lab.eng.bos.redhat.com/ad-user1

[ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com@rhc1 ~]$ hostname
broker1.interop.example.com
```

Note: Since User and Group ID's are POSIX based they can be lengthy depending on the size of the domain names in use:

```
$ id
uid=645801103(ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com)
gid=645801103(ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com)
groups=645801103(ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com),
645800513(domain users@interop-ad.cloud.lab.eng.bos.redhat.com),645801104(idm-
ose-users@interop-ad.cloud.lab.eng.bos.redhat.com),1550200005(ad-ose-users-
posix) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Verification of ssh access from the Active Directory server to the IdM servers requires the use of an ssh client such as PuTTY. Download PuTTY from here:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>



In order to use Kerberos, GSSAPI must be enabled within PuTTY.

12. Disable Kerberos local user mappings

OpenShift clients communicate to OpenShift brokers through either the web interface (aka - *console*) or the rhc client (aka - *broker*) command line interface. By default, when using Kerberos authentication, local users are implied resulting in the removal of the **@REALM** portion of the user mapping. To permit authentication across a cross-realm trust to Active Directory, the default behavior must be disabled in both the OpenShift *broker* and *console* remote authentication files as follows.

- a) Create a backup copy of the OpenShift *broker* remote authentication user configuration file (`/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf`) on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# cp -p /var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf{,.orig}
```

- b) Edit the file and disable Kerberos local user mapping (***KrbLocalUserMapping***) as highlighted below. Perform this step on each OpenShift broker host (**broker1**, **broker2**, **broker3**). Alternatively, the file can be edited on one host and copied to the remaining broker hosts.

```
# Provided by the mod_auth_kerb package
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule auth_kerb_module modules/mod_auth_kerb.so
<Location /broker>
  AuthName "OpenShift broker API"
  AuthType Kerberos
  KrbMethodNegotiate On
  KrbMethodK5Passwd On
  # The KrbLocalUserMapping enables conversion to local users, using
  # auth_to_local rules in /etc/krb5.conf. By default it strips the
  # @REALM part. See krb5.conf(5) for details how to set up specific rules.
  KrbLocalUserMapping Off
  KrbServiceName HTTP/broker.interop.example.com
  KrbAuthRealms INTEROP.EXAMPLE.COM INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM
  Krb5KeyTab /var/www/openshift/broker/httpd/conf.d/http.keytab
  require valid-user
```

- c) Create a backup copy of the OpenShift *console* remote authentication user configuration file (`/var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf`) on all broker hosts (**broker1**, **broker2**, **broker3**).

```
# cp -p /var/www/openshift/console/httpd/conf.d/openshift-origin-auth-remote-user.conf{,.orig}
```

- d) Edit the file and disable Kerberos local user mapping (***KrbLocalUserMapping***) as highlighted below. Perform this step on each OpenShift broker host (**broker1**, **broker2**, **broker3**). Alternatively, the file can be edited on one host and copied to the remaining



broker hosts.

```
LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule auth_kerb_module modules/mod_auth_kerb.so

# 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 Developer Console"
    AuthType Kerberos
    KrbMethodNegotiate On
    KrbMethodK5Passwd On
    # The KrbLocalUserMapping enables conversion to local users, using
    # auth_to_local rules in /etc/krb5.conf. By default it strips the
    # @REALM part. See krb5.conf(5) for details how to set up specific rules.
    KrbLocalUserMapping Off
    KrbServiceName HTTP/broker.interop.example.com
    KrbAuthRealms INTEROP.EXAMPLE.COM INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM
    Krb5KeyTab /var/www/openshift/broker/httpd/conf.d/http.keytab
    require valid-user
```

e) Restart the *broker* and *console* services.

On each OpenShift broker host (**broker1**, **broker2**, **broker3**) restart the *broker* and *console* services.

```
# service openshift-broker restart
```

```
Stopping openshift-broker: [ OK ]
Starting openshift-broker: [ OK ]
```

```
# service openshift-console restart
```

```
Stopping openshift-console: [ OK ]
Starting openshift-console:
```

13. Run RHC setup using Active Directory user account

Configure the **RHC** client by running **rhc setup** and specify the round-robin DNS broker name (**broker**) for the server (**- -server**) name. Perform this step on the RHC client (**rhc1**).

```
$ hostname
```

```
rhc1.interop.example.com
```

```
$ id
```

```
uid=645801103(ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com)
```

```
gid=645801103(ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com)
```



```
groups=645801103(ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com),
645800513(domain users@interop-ad.cloud.lab.eng.bos.redhat.com),
645801104(idm-ose-users@interop-ad.cloud.lab.eng.bos.redhat.com),
1550200005(ad-ose-users-posix)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

```
$ kinit ad-user1@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM
```

```
Password for ad-user1@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM: *****
```

```
$ klist
```

```
Ticket cache: FILE:/tmp/krb5cc_645801103
```

```
Default principal: ad-user1@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM
```

```
Valid starting      Expires            Service principal
01/16/15 10:56:42   01/16/15 20:56:34  krbtgt/INTEROP-AD.CLOUD.LAB.ENG.BOS.
REDHAT.COM@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM
        renew until 01/17/15 10:56:42
```

```
$ rhc setup --server=broker.interop.example.com
```

```
OpenShift Client Tools (RHC) Setup Wizard
```

This wizard will help you upload your SSH keys, set your application namespace, and check that other programs like Git are properly installed.

The server's certificate is self-signed, which means that a secure connection can't be established to 'broker.interop.example.com'.

You may bypass this check, but any data you send to the server could be intercepted by others.

```
Connect without checking the certificate? (yes|no): yes
```

```
Login to broker.interop.example.com: ad-user1@INTEROP-AD.CLOUD.LAB.ENG.BOS.
REDHAT.COM
```

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

OpenShift can create and store a token on disk which allows to you to access the server without using your password. The key is stored in your home directory and should be kept secret. You can delete the key at any time by running 'rhc logout'.

```
Generate a token now? (yes|no) yes
```

```
Generating an authorization token for this client ... lasts about 1 day
```

```
Saving configuration to /home/interop-ad.cloud.lab.eng.bos.redhat.com/ad-
user1/.openshift/express.conf ... done
```

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

```
Created: /home/interop-ad.cloud.lab.eng.bos.redhat.com/ad-user1/.ssh/
id_rsa.pub
```

Your public SSH key must be uploaded to the OpenShift server to access code.
Upload now? (yes|no) yes



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

Uploading key 'default' ... done

Checking for git ... found git version 1.7.1

Checking common problems .. done

Checking for a domain ... none

Applications are grouped into domains - each domain has a unique name (called a namespace) that becomes part of your public application URL. You may create your first domain here or leave it blank and use 'rhc create-domain' later. You will not be able to create an application without completing this step.

Please enter a namespace (letters and numbers only) |<none>|: **adtrusttest**
Your domain 'adtrusttest' has been successfully created

Checking for applications ... none

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

Do-It-Yourself 0.1	rhc create-app <app name> diy-0.1
Jenkins Server	rhc create-app <app name> jenkins-1
Node.js 0.10	rhc create-app <app name> nodejs-0.10
PHP 5.3	rhc create-app <app name> php-5.3
PHP 5.4	rhc create-app <app name> php-5.4
Perl 5.10	rhc create-app <app name> perl-5.10
Python 2.6	rhc create-app <app name> python-2.6
Python 2.7	rhc create-app <app name> python-2.7
Python 3.3	rhc create-app <app name> python-3.3
Ruby 1.8	rhc create-app <app name> ruby-1.8
Ruby 1.9	rhc create-app <app name> ruby-1.9
Tomcat 6 (JBoss EWS 1.0)	rhc create-app <app name> jbossews-1.0
Tomcat 7 (JBoss EWS 2.0)	rhc create-app <app name> jbossews-2.0

You are using 0 of 100 total gears

The following gear sizes are available to you: small

Your client tools are now configured.

\$ rhc domain show

Domain adtrusttest (owned by ad-user1@INTEROP-AD.CLOUD.LAB.ENG.BOS.REDHAT.COM)

```
-----
Created:          12:04 PM
ID:               54b9448b96f47ab044000008
Allowed Gear Sizes: small
Suffix:           interop.example.com
```

The domain adtrusttest exists but has no applications. You can use 'rhc create-app' to create a new application.



14. Verify RHC client

Verify the **RHC** client by checking the domain connectivity and deploying a test application.

```
$ hostname
rhc1.interop.example.com

$ rhc domain show
Domain adtrusttest (owned by ad-user1@INTEROP-
AD.CLOUD.LAB.ENG.BOS.REDHAT.COM)
-----
--
Created:          12:04 PM
ID:              54b9448b96f47ab044000008
Allowed Gear Sizes: small
Suffix:          interop.example.com

The domain adtrusttest exists but has no applications. You can use 'rhc
create-app' to create a new application.

$ rhc create-app adtrustapp php-5.4
Application Options
-----
Domain:          adtrusttest
Cartridges:      php-5.4
Gear Size:       default
Scaling:         no

Creating application 'adtrustapp' ... done

Waiting for your DNS name to be available ... done

Initialized empty Git repository in /home/interop-
ad.cloud.lab.eng.bos.redhat.com/ad-user1/adtrustapp/.git/
The authenticity of host 'adtrustapp-adtrusttest.interop.example.com
(10.19.140.23)' can't be established.
RSA key fingerprint is 4a:a3:3c:fc:5e:8b:9c:55:80:51:32:8b:16:b2:71:31.
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'adtrustapp-
adtrusttest.interop.example.com,10.19.140.23' (RSA) to the list of known
hosts.

Your application 'adtrustapp' is now available.

URL:             http://adtrustapp-adtrusttest.interop.example.com/
SSH to:          54b946de96f47ab044000009@adtrustapp-
adtrusttest.interop.example.com
Git remote:      ssh://54b946de96f47ab044000009@adtrustapp-
adtrusttest.interop.example.com/~/.git/adtrustapp.git/
Cloned to:       /home/interop-ad.cloud.lab.eng.bos.redhat.com/ad-
user1/adtrustapp

Run 'rhc show-app adtrustapp' for more details about your app.
```



```
$ rhc show-app adtrustapp
adtrustapp @ http://adtrustapp-adtrusttest.interop.example.com/ (uuid:
54b946de96f47ab044000009)
-----
Domain:      adtrusttest
Created:     12:14 PM
Gears:       1 (defaults to small)
Git URL:     ssh://54b946de96f47ab044000009@adtrustapp-
adtrusttest.interop.example.com/~/.git/adtrustapp.git/
SSH:         54b946de96f47ab044000009@adtrustapp-
adtrusttest.interop.example.com
Deployment:  auto (on git push)

php-5.4 (PHP 5.4)
-----
Gears: 1 small

$ rhc domain show
Domain adtrusttest (owned by ad-user1@INTEROP-AD.CLOUD.LAB.ENG.BOS.
REDHAT.COM)
-----
--
Created:      12:04 PM
ID:           54b9448b96f47ab044000008
Allowed Gear Sizes: small
Suffix:       interop.example.com

adtrustapp @ http://adtrustapp-adtrusttest.interop.example.com/ (uuid:
54b946de96f47ab044000009)
-----
Domain:      adtrusttest
Created:     12:14 PM
Gears:       1 (defaults to small)
Git URL:     ssh://54b946de96f47ab044000009@adtrustapp-
adtrusttest.interop.example.com/~/.git/adtrustapp.git/
SSH:         54b946de96f47ab044000009@adtrustapp-
adtrusttest.interop.example.com
Deployment:  auto (on git push)

php-5.4 (PHP 5.4)
-----
Gears: 1 small

You have 1 application in your domain.
```

This completes the tasks for integration scenario 5.



8 Conclusion

This reference architecture demonstrated how to deploy and integrate Identity Management in Red Hat Enterprise Linux and OpenShift Enterprise into a cohesive, tightly coupled solution that can be deployed into new or existing OpenShift environments.

Identity Management (IdM) in Red Hat Enterprise Linux provides a simplified, centralized and flexible solution to securely managing user identities, policies and authorizations in a native Linux-based domain. The IdM framework is ideal for large scale Linux and Unix deployments in both traditional and cloud based enterprise environments such OpenShift Enterprise.

For improved availability of IdM services, an IdM replica server was configured. In turn, OpenShift Enterprise was deployed in a distributed configuration with the messaging (ActiveMQ) and database (MongoDB) services shared across multiple broker hosts. The configuration of IdM cross-realm Kerberos trusts was also presented for the authorization and authentication of existing Active Directory domain users.



Acknowledgements

Developing a Red Hat reference architecture is often times a long, arduous journey involving endless hours of coordination and collaboration in the true spirit of Red Hat. This reference architecture is no exception and would not have been possible without the valued input, creativity and guidance provided by the following teams and individuals:

The OpenShift Team: Matt Hicks for the initial concept and inspiration; Luke Meyer for being my OpenShift “wingman”; Jason DeTiberus and Jordan Liggitt for Kerberos sanity checking. John Keck for supporting the project and providing access to the team.

The Identity Management Team: Dmitri Pal for IdM air support; Jan Pazdziora for sanity checking of initial integration scenarios; Jakub Hrozek, Petr Spacek and Alexander Bokovoy for ground support of IdM, DNS and Trusts respectively.

Solutions Engineering Team: Vijay Trehan, Daniel Riek for supporting the project and providing air cover. Scott Collier and Aaron Weitekamp for initial ramp-up and sanity-checking on all things OpenShift.

Thank you to each and every one of these individuals for their participation, contributions and feedback during the development and design of this reference architecture.

Mark Heslin – January, 2015



Appendix A: References

Red Hat Enterprise Linux 6, 7

1. Red Hat Enterprise Linux 6 Installation Guide
Installing Red Hat Enterprise Linux 6 for all architectures
Edition 1.0
http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/pdf/Installation_Guide/Red_Hat_Enterprise_Linux-6-Installation_Guide-en-US.pdf
2. Red Hat Enterprise Linux 6 Deployment Guide
Deployment, Configuration and Administration of Red Hat Enterprise Linux 6
Edition 1.0
http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/pdf/Deployment_Guide/Red_Hat_Enterprise_Linux-6-Deployment_Guide-en-US.pdf
3. Red Hat Enterprise Linux 7 Installation Guide
Installing Red Hat Enterprise Linux 7 for all architectures
Edition 1.0
http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/7/pdf/Installation_Guide/Red_Hat_Enterprise_Linux-7-Installation_Guide-en-US.pdf

Red Hat Enterprise Virtualization

4. Red Hat Enterprise Linux 6 Virtualization Getting Started Guide
Virtualization Documentation
Edition 0.2
http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/pdf/Virtualization_Getting_Started_Guide/Red_Hat_Enterprise_Linux-6-Virtualization_Getting_Started_Guide-en-US.pdf
5. Red Hat Enterprise Linux 6 Virtualization Administration Guide
Virtualization Documentation
Edition 0.1
http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/pdf/Virtualization_Administration_Guide/Red_Hat_Enterprise_Linux-6-Virtualization_Administration_Guide-en-US.pdf

Identity Management in Red Hat Enterprise Linux

6. Identity Management Guide
Managing Identity and Authorization Policies for Linux-Based Infrastructures
Edition 1.0
https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/Linux_Domain_Identity_Authentication_and_Policy_Guide/Red_Hat_Enterprise_Linux-7-Linux_Domain_Identity_Authentication_and_Policy_Guide-en-US.pdf



OpenShift Enterprise 2

7. Deployment Guide
Installing and Configuring OpenShift Enterprise
Edition 1.0
https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/pdf/Deployment_Guide/OpenShift_Enterprise-2-Deployment_Guide-en-US.pdf
8. Administration Guide
A Guide to OpenShift Enterprise Operation and Administration
Edition 1.0
https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/pdf/Administration_Guide/OpenShift_Enterprise-2-Administration_Guide-en-US.pdf
9. Troubleshooting Guide
Troubleshooting Guide for OpenShift Enterprise
Edition 1.0
https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/pdf/Troubleshooting_Guide/OpenShift_Enterprise-2-Troubleshooting_Guide-en-US.pdf
10. User Guide
Managing Applications in the Cloud for OpenShift Enterprise
Edition 1.0
https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/pdf/User_Guide/OpenShift_Enterprise-2-User_Guide-en-US.pdf
11. 2.1 Release Notes
Release Notes for OpenShift Enterprise 2.2
Edition 1.0
https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/2/pdf/2.0_Release_Notes/OpenShift_Enterprise-2-2.0_Release_Notes-en-US.pdf

Microsoft Windows Server 2012 R2

12. Install and Deploy Windows Server
April 23, 2012
<http://technet.microsoft.com/en-us/library/dn645472.aspx>

Active Directory Domain Services

13. Active Directory
August 21, 2013
<http://technet.microsoft.com/en-us/library/dn283324.aspx>
14. Active Directory Lightweight Directory Services Overview
February 29, 2012
<http://technet.microsoft.com/en-us/library/hh831593.aspx>



Appendix B: Glossary

A ACL (Access Control List)

A method for controlling access to files, directories on a file system.

Access Rights

In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.

ActiveMQ

An open source messaging service that supports a wide variety of programming languages. ActiveMQ is maintained by the Apache foundation.

AD (Active Directory)

A suite of directory services developed by Microsoft and based on Novell eDirectory. Active Directory utilizes a number of standardized protocols (*DNS, LDAP, Kerberos*). Active Directory provides a scalable, centralized database infrastructure for securely managing objects (*users, systems, groups, printers, applications*). Directory objects are stored in a hierarchy consisting of *nodes, trees, forests and domains*.

AD DS (Active Directory Domain Services)

An update to Active Directory (AD) introduced in Windows Server 2008 R2. Active Directory Domain Services is included in Windows Server 2008 R2 and is activated as a Server Role.

A Record (Address Record)

A DNS record used to point domain and host names to a static IP address.

authentication

The process of proving the identity of a user or client in order to grant access to a server resource.

B Broker

The OpenShift server(s) responsible for all application management activities - authentication of users, DNS updates, application state, etc. The primary interface to brokers is via RESTful API calls but the web console, CLI and Eclipse JBoss Developer Studio may also be used.

Builtin

One of the default containers defined when an Active Directory domain is created. The Builtin container defines user groups that are within the local scope of the domain - Account Operators, Administrators, Backup Operators, Guests, Network Configuration Operators, etc.

C CA (Certificate Authority)

Company or organization that sells and issues authentication certificates that are known, trusted.

Cartridges

Simplify and automate the configuration and management of the underlying software that



applications within a gear utilize. OpenShift provides a set of pre-defined cartridges for common languages, middleware and databases. Custom cartridges can also be created.

Cgroups

Kernel-level constructs that provide an efficient method of isolating applications and their resources from each other. Cgroups are the underlying mechanism used by OpenShift nodes to manage cpu, disk and memory resources within a gear.

Cross-realm Kerberos Trust

A trusted relationship between two, separate Kerberos realms that allows the sharing of machines or services by users through the granting of a valid Kerberos ticket. IdM Cross-realm Kerberos trusts allow Active Directory users to access Linux resources. All Kerberos communication between Active Directory and IdM Cross-realm trusts use GSS-API.

D daemon

A UNIX/Linux program that runs without human intervention to perform a given task. For example, **smbd** is the Samba server daemon that provides file sharing and print services to clients.

dcdiag (Domain Controller Diagnostics)

A command line diagnostics tool that provides a framework for running a series of tests to verify various components of a server in an Active Directory domain. The dcdiag tool is included with Windows Server 2008 R2 and is available for earlier versions of Windows Server.

Directory Service

A database application that provides a structure for organizing and managing common network objects into collections of name-value mappings. Commonly used directory services include Active Directory (AD), Domain Name System (DNS), Lightweight Directory Access protocol (LDAP), Network Information Service (NIS), OpenLDAP and Red Hat Directory Server (RDS).

Distinguished Name (DN)

The String representation of an entry's name and location in an LDAP directory.

DNS (Domain Name System)

A hierarchical, distributed naming system for managing the mappings of human-friendly domain, host and service names to IP addresses. DNS also defines the protocol for communication exchanges in DNS as part of the Internet Protocol (IP) suite.

Domain

A human-friendly name for an IP address representing a collection of computer and network IP addresses.

Domain Realm

The name of the Active Directory domain.

E Enumeration

Enumeration is the process of listing the users, groups in an Active Directory domain.

F Forest

A forest is a collection of trees that share a common global catalog, directory schema, logical



structure, and directory configuration. The forest represents the security boundary within which users, computers, groups, and other objects are accessible.

Forward Lookup Zone

A FQDN mapped to an IP Address under Active Directory Domain Services.

FQDN (Fully Qualified Domain Name)

A domain name specifying an exact location in a DNS hierarchical tree. For example, a host named *rhel-srv1* that resides in the *bos.redhat.com* domain, has the fully qualified domain name (FQDN) of *rhel-srv1.bos.redhat.com*.

G Gear

Gears are containers that run the applications a user has deployed on a node. A gear is created for each application that is deployed within OpenShift. Management of all resources (cpu, disk, memory) is handled through the use of Control groups (Cgroups).

GID (Group ID)

A numeric value assigned to represent a group of UNIX/Linux users. Groups identify user access to system resources and membership is managed through entries in the file */etc/group*.

GSS-API (Generic Security Services - Application Programming Interface)

The generic access protocol that is the native way for UNIX-based systems to access and authenticate Kerberos services; also supports session encryption.

GSS-TSIG (Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS)

An extension to TSIG that provides an efficient protocol (RFC2136) for authenticating and securing messaging between systems. Like TSIG, GSS-TSIG is primarily used to authenticate Dynamic DNS (DDNS) updates.

H Hostname

The name for a host in the form *host.domain.com*, which is translated into an IP address. For example, *rhel-srv1.interop.example.com* is the machine *rhel-srv1* in the subdomain *interop* and the *example.com* domain.

I IdM (Identity Management)

IdM is Red Hat's implementation of IPA. IdM provides a way to create identity stores, centralized authentication, domain control for Kerberos and DNS services, and authorization policies on Linux systems. Identity Management provides a unifying skin for standards-defined, common network services, including PAM, LDAP, Kerberos, DNS, NTP, and certificate services, and it allows Red Hat Enterprise Linux systems to serve as the domain controllers. Identity Management defines a domain, with servers and clients who share centrally-managed services, like Kerberos and DNS.

ID Mappings

The mapping of UID, GID on the Red Hat Enterprise Linux 6 system to SID in a Windows Active Directory domain.

IMU (Identity Management for UNIX)



An additional role service that enables Red Hat Enterprise Linux 6 system to integrate with Active Directory. IMU is standard on Windows Server 2008 R2 and replaced the optional Services For UNIX (SFU) product on earlier Windows Server versions.

IPA (Identity, Policy and Management)

An open source project for providing centralized, secure user identity management and authorization policies.

IP Address (Internet Protocol Address)

A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10).

K KDC (Key Distribution Center)

The Kerberos database server that manages the secure database of secret keys used for Kerberos trusted authentication. Kerberos clients request a “ticket” from the KDC that has a configurable expiration date and must be renewed on a regular basis.

Kerberos

A network authentication protocol developed at the Massachusetts Institute of Technology (MIT). Kerberos uses strong cryptography to provide highly secure Single Sign-On (SSO) capabilities between client and server applications.

LDAP (Lightweight Directory Access Protocol)

- L** A hierarchical directory service and an application protocol for performing lookups and updates to a remote directory service. LDAP data is transmitted securely over networks via SSL, TLS or SASL. Common LDAP implementations include Active Directory, Apache Directory Server, OpenLDAP, Oracle Internet Directory and Red Hat Directory Server.

LDIF (LDAP Data Interchange Format)

Format used to represent LDAP server entries in text form.

M MongoDB

A NoSQL database used by OpenShift to provide fast, scalable access to internal data (users, gears, metadata.) for deployed applications.

N Node

The OpenShift server(s) responsible for running the applications that a user has deployed. Each application is run within a dedicated gear on the node to facilitate the management of resources.

NIS (Network Information Service)

A client-server directory service protocol developed by Sun Microsystems. NIS provides a method to distribute and share configuration data such as users, hosts and files between networked systems. Due to scalability and security concerns, the function of NIS has been largely replaced by LDAP.

ns-slapd

Red Hat's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server



NSS (Name Service Switch)

A UNIX/Linux facility that manages a variety of common configuration databases and name resolution sources - */etc/passwd*, */etc/group*, */etc/hosts*, *DNS*, *NIS*, *LDAP*, etc.

NTP (Network Time Protocol)

An Internet protocol used to synchronize computer system clocks to a time reference. On Red Hat Enterprise Linux the *ntpd* daemon handles the actual synchronization.

O Objects

An Active Directory object represents a single entity—whether a user, a computer, a printer, or a group—and its attributes. Objects fall into two general categories: resources (e.g., printers) and [security principals](#) (user or computer accounts and groups). Security principals are assigned unique [security identifiers](#) (SIDs).

P PAM (Pluggable Authentication Modules)

A set of libraries that handle the authentication tasks of applications.

PAAS (Platform-as-a-Service)

A cloud application platform in which the application configuration, build, deployment, hosting and administration is automated within an elastic cloud environment that also includes all the supported application stacks.

Permissions

In the context of access control, permissions define whether access to the directory information is granted or denied and the level of access that is granted or denied.

Principal

A user or computer in a Kerberos realm. Principals are stored in the Kerberos authentication database.

Protocol

A set of rules that describes how devices on a network exchange information.

R Realm

A collection of Kerberos principals. In the Kerberos configuration file, the realm includes the name of the KDC and the administration server.

Replication

Act of copying data between servers.

Reverse Lookup Zone

An IP Address mapped to a FQDN under Active Directory Domain Services.

RHDS (Red Hat Directory Server)

An LDAP-compliant server that centralizes user identity and application information. RHDS provides an operating system-independent, network-based registry for storing application settings, user profiles, group data, policies and access control information.

**RID (Relative Identifier)**

The portion of the security identifier (SID) that identifies a user or group in relation to the authority that issued the SID.

RFC (Request For Comments)

Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.

Root User

The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.

S Samba

A suite of programs that provide seamless file and print services for Windows clients in Linux environments. Like CIFS, Samba uses the SMB protocol for client to server communications.

SASL (Simple Authentication and Security Layer)

An authentication framework for clients as they attempt to bind to a directory.

Schema

Definitions describing what types of information can be stored as entries in a database or directory service. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

SELinux (Security-Enhanced Linux)

A flexible, mandatory access control architecture that provides support for the enforcement of access control policies.

Service

A background process on a machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SID (Security Identifier)

A data structure for identifying user, group and system accounts in Windows operating system environments.

slapd

LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication.

SSL (Secure Sockets Layer)

A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP.

SSSD (System Security Services Daemon)

SSSD contains a set of daemons to manage access to remote directories and authentication



mechanisms. It provides PAM (authentication) and NSS (name resolution) modules, a pluggable backend to connect to multiple different account sources and a D-BUS based interface. It is also the basis to provide client auditing and policy services for IdM.

T TCP/IP (Transmission Control Protocol/Internet Protocol)

A standard set of communications protocols organized into four layers – *Application*, *Transport*, *Internet* and *Link*. TCP/IP is the most commonly deployed protocols for computer and network communications.

TLS (Transport Layer Security)

The standard for secure socket layers (SSL); a public key based protocol.

Tree

A tree is a collection of one or more Active Directory domains and domain trees in a contiguous namespace, linked in a transitive trust hierarchy.

Trust

A method to allow users in one Active Directory domain to access the resources in another domain. Under Active Directory a variety of trusts types can be configured:

One-way: One domain allows access to users on another domain, but the other domain does not allow access to users on the first domain.

Two-way: Two domains allow access to users on both domains.

Transitive: A trust that can extend beyond two domains to other trusted domains in the forest.

Intransitive: A one way trust that does not extend beyond two domains.

Explicit: A trust that an admin creates. It is not transitive and is one way only.

Cross-link: An explicit trust between domains in different trees or in the same tree when a descendant/ancestor (child/parent) relationship does not exist between the two domains.

TSIG (Transactional Signature)

A networking protocol (RFC2845) used to authenticate updates to a Dynamic DNS (DDNS) database.

U UID (User ID)

A numeric value assigned to represent a UNIX/Linux user. A UID identifies user access to system resources and is managed through an entry in the file */etc/passwd*.

W Winsync

Winsync is a utility for synchronizing user data and passwords between Red Hat Enterprise Linux 6 hosts running RHDS or IdM and Windows servers. For most environments, the use of IdM Cross-realm Kerberos trusts supercedes and negates the need to use Winsync.



Appendix C: Active Directory - Deployment and Configuration

This summary is provide as a guide to the installation and configuration of Active Directory Domain Services on Windows Server 2012 R2 Datacenter edition.

Prerequisites

The following are required before Active Directory can be configured on a Windows Server 2012 R2 server:

- Administrator account access
- Properly configured NIC (Static IP)
- NTFS partition with 250mb free space for Active Directory
- Functional DNS server (can be installed on the AD server itself or point to an existing DNS server)
- Domain name to use

Note: In this reference architecture, two separate DNS domains are used:

- **interop.example.com** (10.19.140) - OpenShift and IdM hosts
- **refarch-ad.cloud.lab.eng.bos.redhat.com** (10.19.143) - Windows Active Directory

DNS forward and reverse lookup zones are created for both domains.

Installation Summary

Refer to the following Microsoft TechNet article for the most current and comprehensive details:

<http://technet.microsoft.com/en-us/library/dn645472.aspx>¹⁴

An Active Directory installation involves the following series of steps on a Windows Server 2012 R2 server:

1. Install Active Directory Domain Services Role
2. Configure Windows Time Service
3. Create DNS Forward Lookup Zone(s)
4. Create DNS Reverse Lookup Zone(s)
5. Restart DNS Service
6. Verify Active Directory Domain Services
7. Verify Active Directory Domain Services
8. Create User Accounts

A set of Supplemental Tasks is also included. Details on each of these steps are provided in the sections that follow.



Installation Details

1. Install Active Directory Domain Services Role

- Open *Server Manager*
- From the toolbar select **Manage** then **Add Roles and Features** to start the *Add Roles Wizard*.
- On the **Before you begin** page, select **Next**.
- On the **Select installation type** page, select **Role-based or feature-based installation** and then select **Next**.
- On the **Select destination server** page, select **Select a server from the server pool**, select the name of the server (**win2012-srv1**) and then select **Next**.
- On the **Select server roles** page, select **Active Directory Domain Services**, then select **Next**.
- On the **Select features** page, accept the defaults then select **Next**.
- On the **Active Directory Domain Services** page, review the information then select **Next**.
- On the **Confirm installation selections** page, select **Install**.
- On the **Results** page, verify that the installation succeeded, and select **Promote this server to a domain controller** to start the Active Directory Domain Services Configuration Wizard.
- On the **Deployment Configuration** page, select **Add a new forest**. Enter the name of the root domain (**interop-ad.cloud.lab.eng.bos.redhat.com**) then select **Next**.
- On the **Domain Controller Options** page, select the following options:
 - Forest functional level: **Windows Server 2012 R2**
 - Domain functional levels: **Windows Server 2012 R2**

Select **Domain Name System (DNS) server**, specify the DSRM password, then select **Next**.

- On the **DNS Options** page, select **Next**.
- On the **Additional Options** page, verify the default NetBIOS name of the domain, and then select **Next**.
- On the **Paths** page, verify the locations (or accept the defaults) for:
 - Database folder: C:\Windows\NTDS
 - Log files folder: C:\Windows\NTDS
 - SYSVOL folder: C:\Windows\SYSVOLand select **Next**.



- On the **Review Options** page, confirm your selections then select **Next**.
- On the **Prerequisites Check** page, confirm that all prerequisite checks passed successfully, then select **Install**.
- On the **Results** page, verify that the server was successfully configured as a domain controller. The server will be restarted automatically to complete the AD DS installation.

Note: From this point forward, the AD domain name (e.g. - *interop-ad*) must be specified for all user logins

2. Configure Windows Time Service

- From the Windows PowerShell (aka *Command Prompt*) run:

```
PS C:\Users\Administrator.WIN2012-SRV1> w32tm /config \  
/manualpeerlist:"ns1.bos.redhat.com" \  
/syncfromflags:manual /update  
The command completed successfully.
```

Note: Use the time server most appropriate to your environment.

- To verify, enter:

```
C:\WIN-SRV1> w32tm /query /status  
PS C:\Users\Administrator.WIN2012-SRV1> w32tm /query /status  
Leap Indicator: 0(no warning)  
Stratum: 1 (primary reference - syncd by radio clock)  
Precision: -6 (15.625ms per tick)  
Root Delay: 0.0000000s  
Root Dispersion: 10.0000000s  
ReferenceId: 0x4C4F434C (source name: "LOCL")  
Last Successful Sync Time: 5/15/2014 3:05:04 PM  
Source: Local CMOS Clock  
Poll Interval: 6 (64s)
```

3. Create DNS Forward Lookup Zone

- Open *Server Manager*
- Select DNS, right click on the server (**WIN2012-SRV1**) and select **DNS Manager** from the drop down menu
- Right click **Forward Lookup Zones** and select **New Zone** from the drop-down
- The *New Zone Wizard* opens – select **Next**
- Select **Secondary zone**
- Select **Next**
- Enter Zone name: **cloud.lab.eng.bos.redhat.com**
- Select **Next**
- Enter the IP Address of the Master Server: **10.19.143.247**
- Select **Next**

Repeat this for the **interop.example.com** zone with Master Server IP **10.19.140.101**



4. Create DNS Reverse Lookup Zone

- Open *Server Manager*
- Select DNS, right click on the server (**WIN2012-SRV1**) and select **DNS Manager** from the drop down menu
- Right click **Reverse Lookup Zones** and select **New Zone** from the drop-down
- The *New Zone Wizard* opens – select **Next**
- Select **Secondary zone**
- Select **Next**
- Select **IPv4 Reverse Lookup Zone**
- Select **Next**
- Enter Network ID: **10.19.143**
- Select **Next**
- Enter the IP Address of the Master Server: **10.19.143.247**
- Select **Next**
- Select **Finish**

Repeat this for the Network ID: **10.19.140** with Master Server IP **10.19.140.101**

5. Restart DNS Service

- Open *Server Manager*
- Select **DNS**
- Right click **WIN2012-SRV1** then select **Computer Management**
- Double-click **Services and Applications** then select **Services**
- In the list of Services select **DNS Server**
- Right click and select **Restart the service**
- Verify DNS is forwarding lookups. Open a Power Shell window and run:

```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator.WIN2012-SRV1> nslookup www.redhat.com
Server: localhost
Address: 127.0.0.1

Non-authoritative answer:
Name:      e1890.b.akamaiedge.net
Address:  23.196.23.214
Aliases:   www.redhat.com
           wildcard.redhat.com.edgekey.net
           wildcard.redhat.com.edgekey.net.globalredir.akadns.net

PS C:\Users\Administrator.WIN2012-SRV1> ipconfig /all
```



Windows IP Configuration

```
Host Name . . . . . : win2012-srv1
Primary Dns Suffix . . . . . : interop-ad.cloud.lab.eng.
bos.redhat.com
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : interop-ad.cloud.lab.eng.
bos.redhat.com
                                     cloud.lab.eng.bos.redhat.com
```

Ethernet adapter Ethernet:

```
Connection-specific DNS Suffix . :
cloud.lab.eng.bos.redhat.com
Description . . . . . : Red Hat VirtIO Ethernet
Adapter
Physical Address. . . . . : 0A-0A-0A-00-8E-65
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
IPv4 Address. . . . . : 10.19.142.101(Preferred)
Subnet Mask . . . . . : 255.255.248.0
Default Gateway . . . . . : 10.19.143.254
DNS Servers . . . . . : 127.0.0.1
NetBIOS over Tcpip. . . . . : Enabled
```

Tunnel adapter isatap.cloud.lab.eng.bos.redhat.com:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
cloud.lab.eng.bos.redhat.com
Description . . . . . : Microsoft ISATAP Adapter
Physical Address. . . . . : 00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
```

Tunnel adapter Local Area Connection* 12:

```
Connection-specific DNS Suffix . :
Description . . . . . : Teredo Tunneling Pseudo-
Interface
Physical Address. . . . . : 00-00-00-00-00-00-E0
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . :
2001:0:9d38:90d7:2411:3cf2:f5ef:719a(Preferred)
Link-local IPv6 Address . . . . . : fe80::2411:3cf2:f5ef:719a
%14(Preferred)
Default Gateway . . . . . : ::
DHCPv6 IAID . . . . . : 385875968
```



```
DHCPv6 Client DUID. . . . . : 00-01-00-01-1A-F4-3B-7B-0A-0A-0A-00-8E-65
NetBIOS over Tcpi. . . . . : Disabled#
```

7. Verify Active Directory Domain Services

- Open *Server Manager*
- Select **AD DS**
- Scroll down to the *Best Practices Analyzer*
- Under the **TASKS** drop-down menu, select **Start BPA Scan**
- Ensure the server (**win2012-srv1.interop-ad.cloud.lab.eng.bos.redhat.com**) is selected then select **Start Scan**

Review the results and correct any errors or warnings.

8. Create User Accounts

- Open *Server Manager*
- Select **AD DS**
- Right click on the server name (**WIN2012-SRV1**) then select **Active Directory Users and Computers**
- Open **interop-ad.cloud.lab.eng.bos.redhat.com** (Domain)
- Right click on **Users**, select **New**, then **User** and enter:

First name: **AD** Initials: **AD**

Last Name: **User 1**

Full name: **AD AD. User 1**

User logon name:

ad-user1@interop-ad.cloud.lab.eng.bos.redhat.com

- Select **Next**
- Enter Password: *********
Confirm password: *********
- Optionally uncheck the option **User must change password at next logon**
- Select **Password never expires**
- Select **Next**
- Select **Finish**

For more detail on Windows password policy requirements, see the following Microsoft TechNet article:

<http://technet.microsoft.com/en-us/library/cc736605.aspx>



Supplemental Tasks

1. Add Red Hat Enterprise Linux 6 IdM Server DNS A Record

In most environments it is necessary to add a DNS A (Address) records:

- Open *Server Manager*
- Select **DNS**
- Right click on the server name (**WIN2012-SRV1**) then select **DNS Manager**
- Expand **Forward Lookup Zones**
- Right click on the Active Directory domain (**interop-ad.cloud.lab.eng.bos.redhat.com**)
- Select **New Host** (A or AAAA)...
- Enter Name: **idm-srv1**
- Enter IP address: **10.19.140.101**
- Select **Add Host**

Repeat this for each Red Hat Enterprise Linux 6 IdM server.

2. Install Identity Management for UNIX Role¹⁶ (*optional but recommended*)

- Open a Power Shell window as Administrator and run:

```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator.WIN2012-SRV1> Dism.exe /online /enable-feature
/featurename:adminui /all
Deployment Image Servicing and Management tool
Version: 6.3.9600.17031
Image Version: 6.3.9600.17031
Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
Restart Windows to complete this operation.
Do you want to restart the computer now? (Y/N) n
PS C:\Users\Administrator.WIN2012-SRV1> Dism.exe /online /enable-feature
/featurename:nis /all

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Image Version: 6.3.9600.17031

Enabling feature(s)
[=====100.0%=====]
The operation completed successfully.
Restart Windows to complete this operation.
Do you want to restart the computer now? (Y/N) n
PS C:\Users\Administrator.WIN2012-SRV1> Dism.exe /online /enable-feature
```

¹⁶ http://technet.microsoft.com/en-us/library/cc731178.aspx#BKMK_command



```
/featurename:psync /all
```

```
Deployment Image Servicing and Management tool  
Version: 6.3.9600.17031
```

```
Image Version: 6.3.9600.17031
```

```
Enabling feature(s)
```

```
[=====100.0%=====]
```

```
The operation completed successfully.
```

```
Restart Windows to complete this operation.
```

```
Do you want to restart the computer now? (Y/N) Y
```

- Restart the server to activate

3. Configure Group for OpenShift Enterprise Users

- Open *Active Directory Users and Computers* (Start -> *Administrative Tools*)
- Right click **Users** and select **New**, then select **Group**
- Under the *New Object – Group* screen enter the following fields:
 - Group Name: ***idm-ose-users***
- Select **OK**
- Add each of the OpenShift users by right-clicking each user and selecting **Properties**
- Under the *Member Of* tab, select **Add**
- Enter ***idm-ose-users*** for the object name
- Select **Set Primary Group**
- Select **OK**

Repeat this for each group as needed – ***idm-ose-developers***, ***idm-ose-admins***. etc.



4. Configure User Accounts for RFC2307 Support (*optional*)

- Open *Active Directory Users and Computers* (Start -> *Administrative Tools*)
- Right click a user (**e.g. OSE IdM. Admin 1**) and select **Properties**
- Under the *UNIX Attributes* tab set the following fields:
 - NIS Domain: ***interop-ad***
 - UID: ***12340001***
 - Login Shell: ***/bin/bash***
 - Home Directory: ***/home/ose-admin1***
 - Primary group name/GID: ***ose-admins***
- Select **OK**

Repeat this for each user account as needed - ***ose-dev1***, ***ose-user1***, *etc.*



Appendix D: OpenShift Deployment Script

```
#!/bin/bash

#-----#
# File: deploy-ose.sh                                     #
# Date: 2015-01-15                                       #
# Desc: Script to simplify deployment of OpenShift Enterprise across a distributed #
#       (e.g. 3 broker, 3 node) environment. Script uses a small set of environment #
#       variables (described below) that are passed to openshift.sh.               #
# To use:                                                #
# 1. Copy/save this script to the first broker host (primary)                         #
# 2. Modify the following environment variables as appropriate:                      #
#    - DOMAIN                                           #
#    - BROKER1, BROKER2, BROKER3 (add/delete to match number of brokers)           #
#    - NODE1, NODE2, NODE3 (add/delete to match number of nodes)                  #
#    - CONF_NAMED_IP_ADDR=ip_address_of_IdM_server    #
#    - CONF_ACTIVEMQ_REPLICANTS (add/delete to match number of brokers)           #
#    - CONF_DATASTORE_REPLICANTS (add/delete to match number of brokers)          #
# 3. For Node hosts only:                             #
#    If a load balancer is in use for broker hosts, then configure 'broker'       #
#    to that name:                                     #
#        export BROKER_HOSTNAME="broker.${DOMAIN}"    #
#    otherwise, set to any valid broker host name:    #
#        export BROKER_HOSTNAME="broker1.${DOMAIN}"   #
#    Only set this value at line 137 in the script.  #
# 4. Set the script execute permissions (e.g. chmod 755, chmod u+x, etc.)          #
# 5. Copy this script to each of the remaining broker hosts, node hosts           #
# 6. Run the script as root on each host according to type, in this order:        #
#    - Secondary broker hosts:                                                       #
#        ./deploy-ose.sh secondary (e.g. - broker2, broker3, etc.)                #
#    - Primary broker host:                                                         #
#        ./deploy-ose.sh primary (e.g. - broker1)                                  #
#    - All node hosts:                                                             #
#        ./deploy-ose.sh node (e.g. - node1, node2, etc.)                         #
# 7. Post deployment, run the script as root on the Primary broker host:          #
#    - Primary broker host only:                                                    #
#        ./deploy-ose.sh post-deploy                                               #
# Note: Script is provided as a supplement to the Red Hat Reference Architecture:  #
#       "Integrating OpenShift Enterprise with Identity Management (IdM)           #
#       in Red Hat Enterprise Linux". For more details see:                       #
#       http://www.redhat.com/resourcelibrary/reference-architectures/ #
#-----#
#
# Add/delete/modify according to your environment:
#
DOMAIN="interop.example.com"
BROKER1="broker1.${DOMAIN}"
BROKER2="broker2.${DOMAIN}"
BROKER3="broker3.${DOMAIN}"
NODE1="node1.${DOMAIN}"
NODE2="node2.${DOMAIN}"
NODE3="node3.${DOMAIN}"
```



```
#
# Applies to all host types - do not modify:
#
export CONF_INSTALL_METHOD="none"
export CONF_DOMAIN=${DOMAIN}
export CONF_KEEP_HOSTNAME="true"

#-----#
# Broker details: #
#-----#

# Set CONF_NAMED_IP_ADDRESS to use IdM server for lookups
export CONF_NAMED_IP_ADDR="10.19.140.101"
export BROKER_HOSTNAME=`/bin/hostname`
export CONF_BROKER_HOSTNAME="${BROKER_HOSTNAME}"
export CONF_ACTIVEMQ_HOSTNAME=`/bin/hostname`
export CONF_DATASTORE_HOSTNAME=`/bin/hostname`

# Add/delete according to the number of brokers in your environment:
export CONF_ACTIVEMQ_REPLICANTS="${BROKER1}, ${BROKER2}, ${BROKER3}"
export CONF_DATASTORE_REPLICANTS="${BROKER1}:27017, ${BROKER2}:27017, ${BROKER3}:27017"

#
# Set username/passwords according to your environment security policies:
#
export CONF_OPENSIFT_USER1="user1"
export CONF_OPENSIFT_PASSWORD1="password"
export CONF_MONGODB_BROKER_USER="openshift"
export CONF_MONGODB_BROKER_PASSWORD="mongopass"
export CONF_MONGODB_ADMIN_USER="admin1"
export CONF_MONGODB_ADMIN_PASSWORD="mongopass"
export CONF_MONGODB_REPLSET="ose"
export CONF_MONGODB_KEY="OSEnterprise"
export CONF_MCOLLECTIVE_USER="mcollective"
export CONF_MCOLLECTIVE_PASSWORD="mcollective"
export CONF_ACTIVEMQ_ADMIN_PASSWORD="password"
export CONF_ACTIVEMQ_AMQ_USER_PASSWORD="password"

#-----#
# Main #
#-----#

function usage {
    printf "\n\tusage:  $0 primary|secondary|node|post-deploy\n\n"
    exit 1
}

if [ "`whoami`" != "root" ]
then
    printf "\n\t*** Must be root to run ***\n\n"
    exit 2
fi

if [ $# != 1 ]
```



```
then
    usage
fi

#
# Set installation target type
#
case $1 in
    primary)
        TARGET="the *Primary* Broker"
        # secondary brokers must be fully running before replica set is initiated
        export CONF_INSTALL_COMPONENTS="broker,activemq,datastore"
        export CONF_ACTIONS="do_all_actions,configure_datastore_add_replicants"
        ;;

    secondary)
        TARGET="a *Secondary* Broker"
        export CONF_INSTALL_COMPONENTS="broker,activemq,datastore"
        export CONF_ACTIONS="do_all_actions"
        ;;

    node)
        TARGET="a *Node*"
        export CONF_INSTALL_COMPONENTS="node"
        export CONF_ACTIONS="do_all_actions"
        NODE="/bin/hostname"
        export CONF_NODE_HOSTNAME="${NODE}"
        # Adjust accordingly to match whether or not load balancer in use:
        export BROKER_HOSTNAME="broker.${DOMAIN}"
        ;;

    post-deploy)
        TARGET=""
        TYPE=$1
        ;;

    *)
        usage
        ;;
esac

#
#-----#
# Download and run openshift.sh using the previously set variables: #
#-----#
#
printf "\nDeploying OpenShift Enterprise in a distributed environment\n"
printf "\n\t...Installation type set to *${1}*...\n"
sleep 2
if [ $1 != "post-deploy" ]
then
    printf "\n*** Control-C now if this is not ${TARGET} or the wrong host ***\n"
    sleep 8
    printf "\n\t...Continuing with installation...\n"
else
```



```
    printf "\n\t...Continuing with post deployment tasks...\n"
fi

printf "\n\t...Downloading openshift.sh script...\n\n"
sleep 5

curl https://raw.githubusercontent.com/openshift/openshift-extras/enterprise-2.2/enterprise/install-scripts/generic/openshift.sh -o openshift.sh
if [ $? != 0 ]
then
    printf "\n\t*** Can not complete download ***\n"
    printf "\n\tThe openshift.sh script (or site) is currently not accessible.\n"
    exit 3
fi

printf "\n\t* Script download complete *\n"
chmod 755 openshift.sh 2>&1 | tee /tmp/openshift.out

if [ $1 != "post-deploy" ]
then
    printf "\n\t...Beginning installation - `date` ...\n\n"
    ./openshift.sh
    printf "\n\t* Installation completed - `date` * \n\n"
else
    printf "\n\t...Beginning post deployment tasks - `date` ...\n"
    ./openshift.sh "actions=post_deploy"
    printf "\n\t* Post deployment tasks completed - `date` * \n\n"
fi

exit 0
```



Appendix E: Deployment Checklist

Task	Task Description	Location	Details
Infrastructure Staging Tasks			
1	Red Hat Enterprise Virtualization	RHEV Hosts	Section 4.1
2	Install Red Hat Enterprise Linux	All hosts	Section 4.2
3	Configure Network	All hosts	Section 4.3
4	Configure Domain Name System (DNS)	All hosts	Section 4.4
5	Configure Network Time Protocol (NTP)	All hosts	Section 4.5
6	Configure SELinux	All hosts	Section 4.6
7	Update Hosts File	All hosts	Section 4.7
8	Register Hosts and Run Updates	All hosts	Section 4.8
Deployment Tasks - IdM			
Deploy IdM Master Server			
1	Configure Firewall Ports	idm-srv1	Section 5.1.1
2	Install Packages	idm-srv1	Section 5.1.2
3	Install/configure IdM Server	idm-srv1	Section 5.1.3
4	Verify Server	idm-srv1	Section 5.1.4
5	Create Groups and Users	idm-srv1	Section 5.1.5
6	Verify Groups and Users	idm-srv1	Section 5.1.6
Deploy IdM Admin Client			
1	Configure Firewall Ports	admin1	Section 5.2.1
2	Install Packages	admin1	Section 5.2.2
3	Configure DNS	admin1	Section 5.2.3
4	Install/configure Admin Client	admin1	Section 5.2.4
5	Verify IdM Admin Client	admin1	Section 5.2.5
6	Install Admin Tools (optional)	admin1	Section 5.2.6
Deploy IdM Replica Server			
1	Configure Firewall Ports	idm-srv2	Section 5.3.1
2	Install Packages	idm-srv2	Section 5.3.2
3	Run Replica Prepare	idm-srv1	Section 5.3.3
4	Copy Replica Information File	idm-srv1	Section 5.3.4
5	Run Replica Installation	idm-srv2	Section 5.3.5
6	Verify DNS	idm-srv2	Section 5.3.6
7	Verify Replication	idm-srv2	Section 5.3.7



Appendix E: Deployment Checklist (continued)

Task	Task Description	Location	Details
Deployment Tasks - OpenShift Enterprise			
Configure Deployment Script			
1	Configure Deployment Script	All broker, node hosts	Section 6.1
Deploy Brokers			
1	Configure Entitlements	broker1, broker2, broker3	Section 6.2.1
2	Configure Yum	broker1, broker2, broker3	Section 6.2.2
3	Install OpenShift	broker1, broker2, broker3	Section 6.2.3
4	Re-configure Firewall Ports (optional)	broker1, broker2, broker3	Section 6.2.4
Deploy Nodes			
1	Configure Entitlements	node1, node2, node3	Section 6.3.1
2	Configure Yum	node1, node2, node3	Section 6.3.2
3	Install OpenShift	node1, node2, node3	Section 6.3.3
4	Re-configure Firewall Ports (optional)	node1, node2, node3	Section 6.3.4
5	Verify Communications	node1, node2, node3	Section 6.3.5
Post Deployment Tasks			
1	Run Post_Deploy	broker1	Section 6.4.1
2	Copy public rsync keys	node1, node2, node3	Section 6.4.2
3	Copy ssh host keys, httpd keys/certs	node1, node2, node3	Section 6.4.3
4	Copy Auth Keys across Brokers	broker1, broker2, broker3	Section 6.4.4
5	Configure Broker DNS request balancing (optional)	idm-srv1 (or idm-srv2)	Section 6.4.5
6	Update DNS Entries	idm-srv1 (or idm-srv2)	Section 6.4.6
Deploy RHC Client			
1	Configure Entitlements	rhc1	Section 6.5.1
2	Install Package	rhc1	Section 6.5.2
Verify OpenShift Enterprise Environment			
1	Reboot Hosts	All broker, node hosts	Section 6.6.1
2	Run Diagnostics	All broker, node hosts	Section 6.6.2



Appendix F: Integration Checklist

Task	Task Description	Location	Details
Integration Prerequisites			
1	Configure Firewall Ports	All broker, node hosts; rhc client	Section 7.2.1
2	Verify DNS	All broker, node hosts; rhc client	Section 7.2.2
3	Configure IdM Clients	All broker, node hosts; rhc client	Section 7.3.3
IS 1 - Developer Authorization to Brokers (mandatory)			Section 7.3.1
1	Create broker HTTP services	idm-srv1	Step 1
2	Create broker HTTP keytabs	broker1	Step 2
3	Configure broker, console to use Kerberos	broker1, broker2, broker3	Step 3
4	Restart broker and console services	broker1, broker2, broker3	Step 4
5	Configure nsupdate plugin	broker1, broker2, broker3	Step 5
6	Create broker DNS services	idm-srv1	Step 6
7	Modify DNS zone for dynamic DNS	idm-srv1	Step 7
8	Create broker DNS keytabs	broker1, broker2, broker3	Step 8
9	Restart broker services	broker1, broker2, broker3	Step 9
10	Configure RHC client	rhc1	Step 10
11	Verify RHC client	rhc1	Step 11
12	Verify OpenShift	All broker, node hosts	Step 12
13	Additional Verification and Troubleshooting	rhc1	Step 13
IS 2 - Centralized Management of SSH Keys (optional)			Section 7.3.2
1	Generate SSH key pair	admin1 (or any IdM client)	Step 1
2	Upload SSH public key to IdM server	admin1 (or any IdM client)	Step 2
3	Verify SSH key	admin1 (or any IdM client)	Step 3
IS 3 - Centralized Management of UID's (optional)			Section 7.3.3
1	Obtain Kerberos ticket	idm-srv1/2 (or admin1)	Step 1
2	Create IdM group	idm-srv1/2 (or admin1)	Step 2
3	Create IdM user	idm-srv1/2 (or admin1)	Step 3
4	Verify IdM group and user	idm-srv1/2 (or admin1)	Step 4
5	Verify IdM group and user	admin1	Step 5
6	Additional verification and troubleshooting	idm-srv1/2 (or admin1)	Step 6



Appendix F: Integration Checklist (continued)

Task	Task Description	Location	Details
IS4 - Kerberized Access to Gears (optional)			Section 7.3.4
1	Create Kerberos Principal SSH key	rhc1	Step 1
2	Verify the key	rhc1	Step 2
3	Obtain Kerberos ticket	rhc1	Step 3
4	Verify access to gear	rhc1	Step 4
IS 5 - Cross-realm Kerberos Trust (optional)			Section 7.3.5
1	Install ipa-server-trust-ad package	idm-srv1, idm-srv2	Step 1
2	Run ipa-adtrust-install	idm-srv1, idm-srv2	Step 2
3	Re-configure firewall ports for AD trust	idm-srv1, idm-srv2	Step 3
4	Add DNS forwarder	Idm-srv1	Step 4
5	Modify Kerberos domain and realms	idm-srv1, idm-srv2	Step 5
6	Run ipa trust-add	Idm-srv1	Step 6
7	Restart KDC	idm-srv1, idm-srv2	Step 7
8	Create IdM external and IdM POSIX groups	idm-srv1	Step 8
9	Add AD group to IdM external group	idm-srv1	Step 9
10	Add IdM external group to IdM POSIX group	idm-srv1	Step 10
11	Verify access	idm-srv1, idm-srv2	Step 11



Appendix G: Revision History

Revision 1.0

July, 2014

Mark Heslin

Initial Release

Revision 1.1

January, 2015

Mark Heslin

Solution validated for OpenShift Enterprise 2.2 on Red Hat Enterprise Linux 6.6.
Revised to include deployment of OpenShift applications with Active Directory accounts across cross-realm trust; streamlining of Kerberos keytabs/service principals for IdM web (HTTP) services.

