



Red Hat Reference Architecture Series

Building an ITOps PaaS with Red Hat's OpenShift Enterprise PaaS Solution

Scott Collier, RHCA
Senior Principal Software Engineer

Version 1.0

June 2012





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

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

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.

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

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

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

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

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

Send feedback to refarch-feedback@redhat.com



Table of Contents

1 Executive Summary.....	1
2 Components Overview.....	4
2.1 CloudForms.....	4
2.2 JBOSS.....	5
2.2.1 JBoss Enterprise Web Server	5
2.2.2 JBoss Enterprise Application Platform.....	5
2.2.3 JBoss Operations Network	6
2.2.4 JBoss Quickstart Applications.....	7
2.3 Red Hat Enterprise Virtualization.....	7
2.4 PostgreSQL.....	8
2.5 Red Hat Enterprise Linux	8
3 Reference Architecture Environment.....	10
3.1 Hardware.....	11
3.2 Storage.....	11
3.3 Software.....	11
4 CloudForms Infrastructure Prerequisites.....	18
4.1 Preparing System / Cloud Engine Servers.....	18
4.2 Preparing the Red Hat Enterprise Virtualization Environment.....	19
5 Preparing the JBoss Infrastructure.....	21
5.1 Prepare System Templates.....	22
5.1.1 JBoss EAP / EWS Server Template	22
5.1.2 PostgreSQL Template	23
5.1.3 Configuration Server Template.....	24
5.1.4 Promote the Templates.....	25
5.2 Prepare Target Content	25
6 Configuration Server	26
6.1 Obtain Software	26
6.2 Set up repository for scripts and XML files	26
6.3 Scripts and XML files.....	29
6.3.1 Services.....	29



6.4 Deploy Configuration Server	31
6.4.1 Build and Push Configuration Server	31
6.4.2 Launch Configuration Server	36
6.4.3 Configure the Configuration Server	38
6.4.4 Add the Configuration Server to Cloud Engine.....	39
7 Deploy JBoss Infrastructure.....	42
7.1 Overview.....	42
7.2 Deploy JBoss Enterprise Web Server.....	42
7.2.1 Build and Push JBoss Enterprise Web Server	43
7.2.2 Add the XML service to the Image.....	46
7.2.3 Launch JBoss Enterprise Web Server	50
7.2.4 Test JBoss Enterprise Web Server	52
7.3 Deploy PostgreSQL Server	53
7.3.1 Build and Push PostgreSQL Server	53
7.3.2 Add the XML configuration to the Image and Launch.....	54
7.3.3 Test PostgreSQL Server	54
7.4 Deploy JBoss Enterprise Application Domain Controller	56
7.4.1 Create Application Blueprint for JBoss Enterprise Application Domain Controller ..	56
7.4.2 Add the XML configuration to the Image and Launch.....	57
7.4.3 Test Enterprise Application Domain Controller.....	58
7.5 Deploy the JBoss Enterprise Application Host Controllers.....	59
7.5.1 Launch the JBoss Enterprise Application Host Controller	59
7.5.2 Confirm the Host Controllers.....	60
8 Deploy JBoss Application.....	63
8.1 Deploy the Application	63
9 Confirm Functionality.....	68
9.1 Access the Application Directly.....	68
9.2 Access the Application via JBoss EWS.....	70
9.3 Confirm Load Balancing Functionality.....	71
10 Summary.....	72
Appendix A: Revision History.....	73
Appendix B: Troubleshooting.....	74



Appendix C: Contributors.....	77
-------------------------------	----



1 Executive Summary

Red Hat Cloud spans a broad portfolio of products in addition to supporting those from a rich ecosystem of partners. Whatever combination of the leading commercial Linux operating system, scalable and secure virtualization, distributed software-only storage, proven enterprise middleware, hybrid open cloud management, or Platform-as-a-Service you need, Red Hat can provide you with the solution.

RED HAT OPEN HYBRID CLOUD

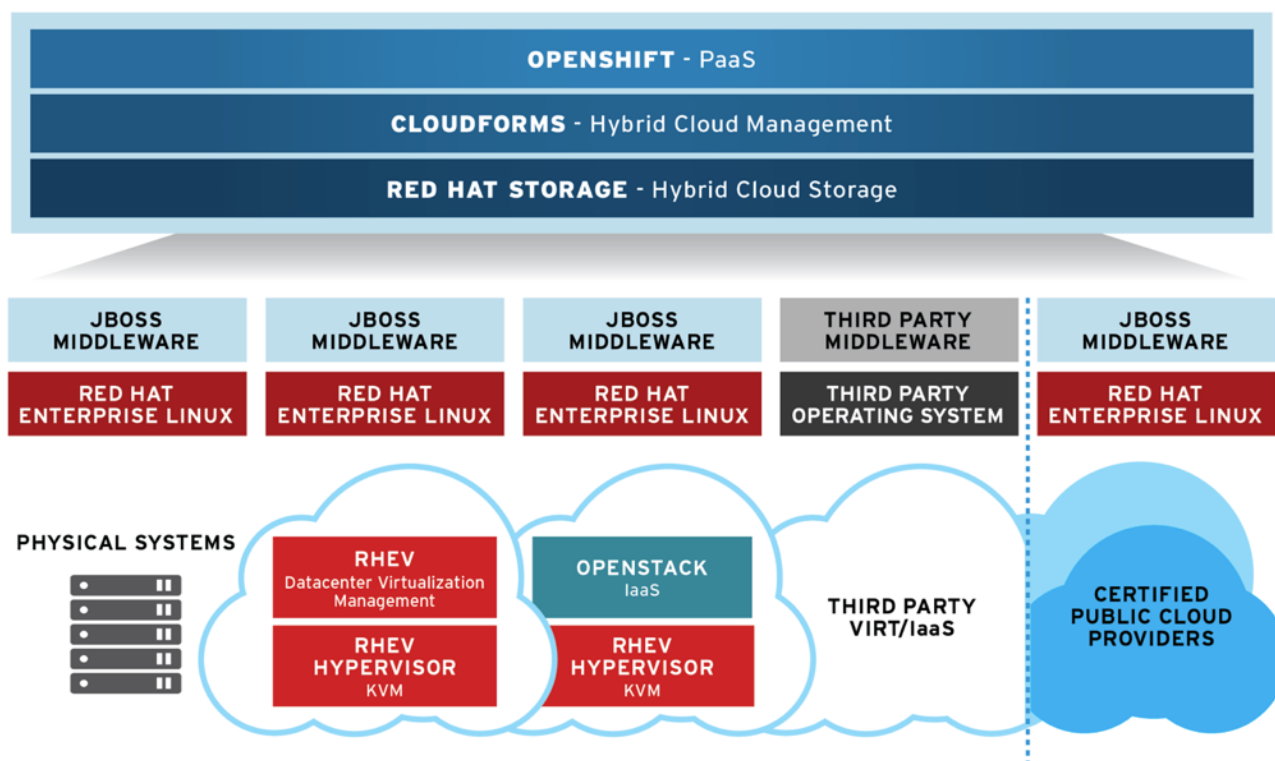


Figure 1.1: Red Hat Cloud

Red Hat CloudForms is an open hybrid Infrastructure-as-a-Service (IaaS) cloud management product. It delivers the flexibility and agility that businesses want with the control and governance that IT needs. This allows organizations to build a hybrid cloud that encompasses their heterogeneous infrastructure, thereby avoiding vendor lock-in, while also managing the applications running in that cloud. Red Hat CloudForms allows you to:

- Build and manage an open enterprise hybrid cloud, providing choice and spanning across physical servers, heterogeneous virtualization platforms, and a broad choice of public cloud providers
- Build and manage applications in that cloud, giving enterprises the level of control they need across all their classes of workloads



- Provide end users with self-service capabilities for increased speed and flexibility, while giving IT administrators the tools they need to establish policies to govern that access

Figure 1.2: CloudForms Architecture highlights CloudForms' features and benefits.

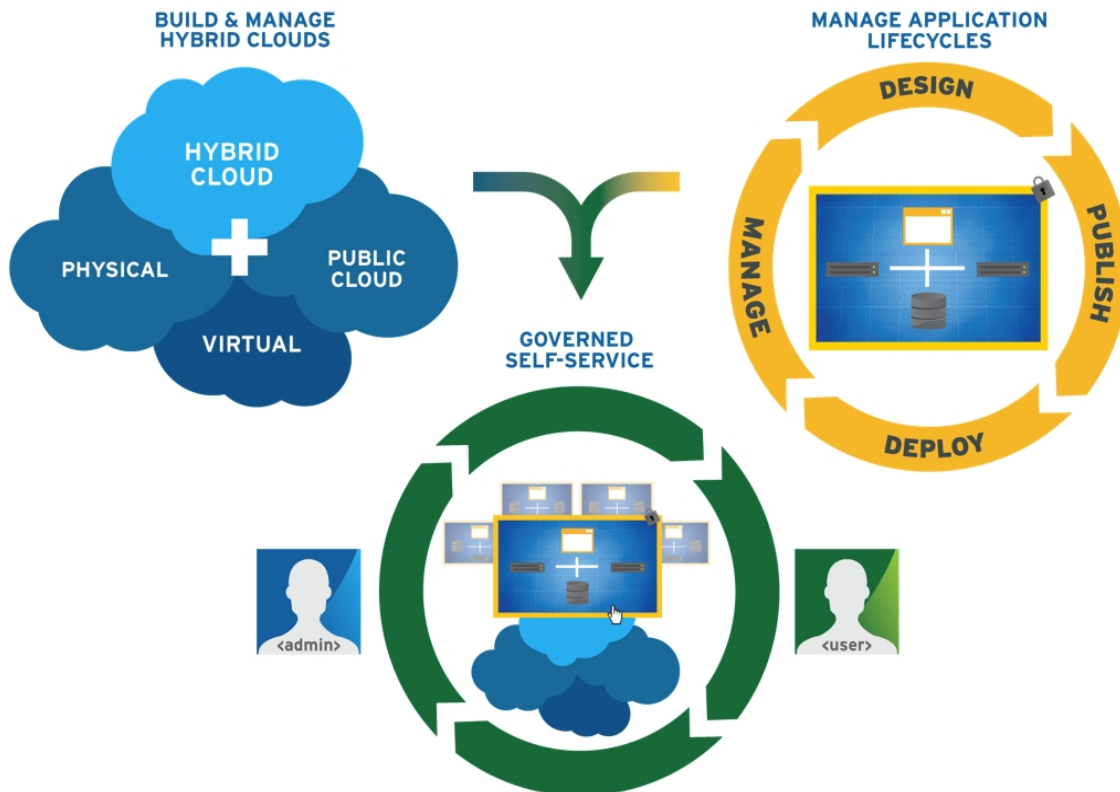


Figure 1.2: CloudForms Architecture

Red Hat's OpenShift Enterprise PaaS solution is an enterprise cloud application platform that will enable organizations to leverage the agility and efficiency of the cloud, while addressing the real-world governance and operational requirements of the enterprise. It provides a comprehensive development and execution platform for enterprise applications, delivered with a choice of cloud deployment and operational management models. OpenShift Enterprise PaaS can be deployed as both a hosted and an on-premise solution, which can support both a traditional IT Operations (ITOps) driven operational model as well as rapidly emerging DevOps models.

Platform-as-a-Service (PaaS) is a cloud service model that is revolutionizing application development, by providing developers with tremendous agility and flexibility. PaaS provides a pre-configured, auto-scaling, and self-managing application platform in the cloud that allows developers to easily develop, deploy, and run their applications. However, there are often complexities in Enterprise organizations that make implementing a PaaS a non-trivial task.



Enterprise Developers need a PaaS that is built on a technology stack that provides the required performance, security, robustness, and application portability to enable easy and sustained adoption. In addition to the technology stack, Enterprise Architects often have to worry about a variety of other influencing factors such as compliance, enterprise architecture standards (including ITIL or other methodologies), IT governance, security, data and compute locality and more. And, finally, the Enterprise IT Operations team needs to worry about where and how the PaaS will operate, i.e., in the Public Cloud, Private Cloud, or Hybrid Cloud and how they will ensure and manage SLA's to the business. For real adoption in the Enterprise, a PaaS must be able to address all these various concerns.

Red Hat's OpenShift Enterprise PaaS delivers the benefits of PaaS to the Enterprise while providing the flexibility to conform to an Enterprise's business requirements. The OpenShift PaaS platform is an enterprise-class application stack built on proven technology from Red Hat. OpenShift Enterprise PaaS is available as a Public Cloud service at OpenShift.com and can also be deployed as a Private or Hybrid Cloud solution. Red Hat plans to offer flexible management and operational models for deploying an on-premise PaaS, including a DevOps option that delivers the same flexibility and automation as OpenShift Hosted and an ITOps option that gives ITOps more control over their PaaS environment. The OpenShift Enterprise PaaS is a bundled offering for enterprise application development and execution in the Cloud. It includes the following components that you can use to build an ITOps PaaS environment today:

- Red Hat CloudForms – an on-premise cloud management framework to build open hybrid clouds, manage cloud application lifecycles and enable governed self-service deployment for end users
- JBoss Enterprise Application Platform Managed – a cloud-ready, enterprise-class, JEE6 certified application middleware platform for running enterprise Java applications
- Red Hat Enterprise Linux - a secure, scalable and reliable operating system platform for hosting cloud application environments
- Red Hat Enterprise Virtualization - an powerful virtualization solution that combines the KVM hypervisor with enterprise virtualization management

This solution enables Red Hat customers to build an on-premise ITOps PaaS environment to bring agility to application development, while addressing the needs of enterprise IT to ensure governance, security and compliance objectives. In the future this solution will be expanded to enable users to build a DevOps PaaS environment through the integration of OpenShift Origin source code that powers OpenShift.com. This Reference Architecture covers building an ITOps PaaS environment with Red Hat CloudForms.

CloudForms enables administrators to build a cloud infrastructure leveraging multiple providers including RHEV, VMWare vSphere, and Amazon EC2. They can then create rich application blueprints in CloudForms, leveraging RHEL and JBoss EAP as well as other Red Hat and 3rd party components. These application blueprints can then be added to an application catalog and deployed via a self-service portal. Administrators can control which application blueprints users have access to and also where they are allowed to run. This provides greater agility for their users while maintaining control and governance of their applications and infrastructure.



2 Components Overview

2.1 CloudForms

Red Hat CloudForms is an open hybrid Infrastructure-as-a-Service (IaaS) cloud management product. It delivers the flexibility and agility that businesses want with the control and governance that IT needs. This allows organizations to build a hybrid cloud that encompasses their heterogeneous infrastructure, thereby avoiding vendor lock-in, while also managing the applications running in that cloud. Red Hat CloudForms allows you to:

- Build and manage an open enterprise hybrid cloud, providing choice and spanning across physical servers, heterogeneous virtualization platforms, and a broad choice of public cloud providers
- Build and manage applications in that cloud, giving enterprises the level of control they need across all their classes of workloads
- Provide end users with self-service capabilities for increased speed and flexibility, while giving IT administrators the tools they need to establish policies to govern that access

Figure 2.1.1: CloudForms Architecture highlights CloudForms' features and benefits.

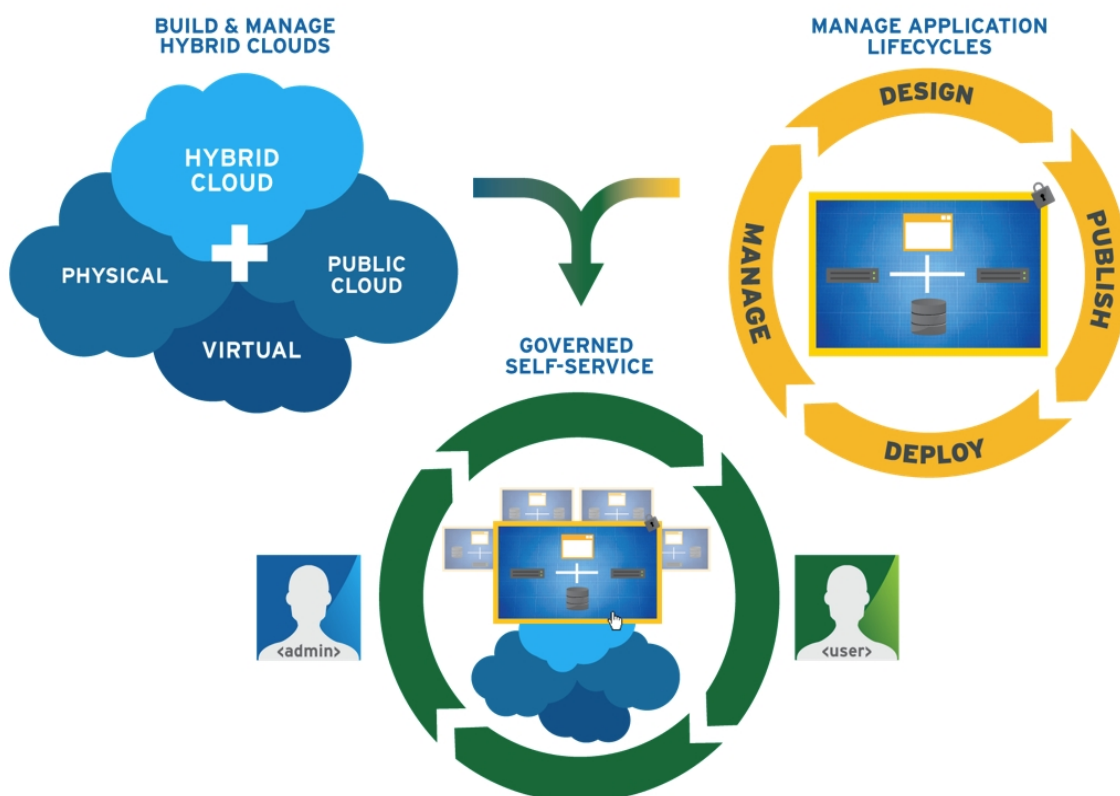


Figure 2.1.1: CloudForms Architecture



Building a cloud is a highly strategic IT decision. In fact, it's perhaps the most important single decision CIOs will make this decade. But not all approaches are created equal. The open cloud approach taken by CloudForms provides important benefits to both users and builders of clouds. It lets them deploy on their choice of private or public infrastructure. It lets them bring both new and existing IT assets into the cloud and manage them together. It lets them evolve to the cloud, gaining incremental value at each step along the way. It ensures that their applications and data can be moved across clouds. It keeps them in charge of their technology roadmap and the future of their IT.

Red Hat CloudForms delivers on open cloud, bringing together cloud computing with Red Hat's rich heritage as the open source leader. This heritage isn't just about open source code but the communities and business models that go hand-in-hand. Red Hat is the vendor best prepared to help you build an open cloud using CloudForms. At the same time, CloudForms ensures that certification and governance can be baked into cloud infrastructure. It does so by adding rich policy controls to user self-service, including application lifecycle management designed for the cloud, and by offering the choice of Red Hat's proven stack and ecosystem to better deliver enterprise-class Service Level Agreements (SLA) in the cloud. Combined with the flexibility, portability, and choice provided by an open cloud, CloudForms provides management that truly delivers the best of both worlds: the speed and agility of cloud with the control needed by the business.

2.2 JBOSS

JBoss Enterprise Middleware provides a comprehensive middleware portfolio. It includes tools and platforms for developing, deploying and managing applications and services, integrating applications, services and data, managing business rules, and enabling rich user interaction and collaboration – either on premises or in the cloud.

2.2.1 JBoss Enterprise Web Server

JBoss Enterprise Web Server is an enterprise-class web server solution for large-scale websites and lightweight web applications. It combines certified, production-ready versions of the world's most deployed web server (Apache) and top servlet engine (Tomcat) with Red Hat's industry-leading middleware support.

JBoss EWS includes:

- Apache Tomcat, the market-leading servlet container
- Apache Web Server, the world's most popular web server
- All the common modules and connectors for security and load balancing

2.2.2 JBoss Enterprise Application Platform

JBoss Enterprise Application Platform (EAP) is the industry leading platform for next-generation enterprise Java applications. It provides a stable, open source foundation for highly transactional Java applications and services. By combining market-leading technologies into a single, simple, and flexible solution, JBoss EAP makes it easy to develop, deploy, and manage Java Enterprise Edition (EE) applications.



JBoss EAP includes the following components:

- JBoss Application Server - the most widely used Java EE-certified application server on the market
- JBoss Hibernate® - the leading technology for object-relational mapping (ORM) and persistence
- JBoss Seam - a powerful application framework that simplifies building web 2.0 applications
- JBoss Web Framework Kit - providing enterprise support for popular open source frameworks

JBoss Enterprise Application Platform 6 is the newest EAP release which combines a cloud ready architecture with world-class developer productivity. JBoss EAP6 was designed and architected for use everywhere - on premise physical and virtual infrastructure, in public clouds, and in private clouds. It features an updated cloud-ready modular architecture, a highly efficient container with extremely low memory footprint with blazing fast startup times, and flexible manageability. JBoss EAP 6 also provides world-class developer productivity including support for Java EE6 and other popular frameworks like Spring and Struts. It also includes support for modern development tools like Maven and Jenkins, updates to Hibernate and new components like Arquillian. This makes Java EAP 6 the best platform for building and running enterprise applications on-premise or in the cloud. JBoss EAP 6 can be run in one of two ways.

- Standalone Server
- Managed Domain

A standalone server instance is a independent process that can be launched via the *standalone.sh*. In standalone server mode, if multi-server management is desired it is the responsibility of the administrator to coordinate management across the servers. Managed domain mode can be launched via *domain.sh*. In a managed domain environment a collection of servers are referred to as members of a “domain” with a single domain controller. The domain controller acts a central management control point. All JBoss EAP 6 instances in the domain share a common management policy and the domain controller ensures that each server is configured according to that policy. This Reference Architecture uses the Managed Domain mode.

2.2.3 JBoss Operations Network

Red Hat's JBoss Operations Network, a key component of Red Hat's JBoss managed solutions, provides a single point of control to deploy, manage and monitor your JBoss Enterprise Middleware, applications and services.

JBoss Operations Network enables organizations to:

- Simplify application release management with support for application provisioning and patching
- Support IT governance objectives with enterprise-grade configuration management and run-time SOA governance
- Ensure application service levels with performance and availability monitoring of



JBoss Operations Network

JBoss Operations Network (JBoss ON) automatically discovers application resources, including platforms, servers and services, running on physical, virtual or cloud-based environments. These can then be added to JBoss ON's managed resource inventory so users can manage all of their applications from a single dashboard. JBoss ON can then be used to monitor the performance and availability of applications and underlying infrastructure, generate alerts when issues occur, modify application configurations and even deploy new applications and updates.

2.2.4 JBoss Quickstart Applications

To get started quickly with JBoss, Red Hat provides some quickstart applications. These applications can be built and deployed quickly to confirm environment functionality. This reference architecture uses the kitchensink quickstart application for demonstration purposes.

2.3 Red Hat Enterprise Virtualization

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



2.4 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl and ODBC. This reference architecture uses PostgreSQL as the backend database for the kitchensink quickstart application.

2.5 Red Hat Enterprise Linux

Red Hat Enterprise Linux 6, the latest release of Red Hat's trusted data center platform, delivers advances in application performance, scalability, and security. Red Hat Enterprise Linux 6 can deploy physical, virtual, and cloud computing within the data center, reducing complexity, increasing efficiency, and minimizing administration overhead while leveraging existing technical skills and operational know-how. Red Hat Enterprise Linux 6 is an ideal platform to translate current and future technology innovations into the best value and scale for IT solutions.

Red Hat Enterprise Linux 6 builds on the previous Red Hat Enterprise Linux releases by introducing significant improvements in the areas of:

Area	Featured Components
<i>Efficiency, scalability and reliability</i>	Scalability, scheduling, reliability, availability and serviceability (RAS), file systems, high availability, power management
<i>Resource Management</i>	System resource allocation, storage, networking
<i>Security</i>	Access control, enforcement and verification of security policies, identity management (IDM)
<i>Application Development and Production</i>	Web infrastructure, Java, development tools/utilities, application tuning, databases, system API/ABI stability
<i>Virtualization</i>	Kernel-based virtualization, kernel features, guest acceleration, security, Microsoft Windows (WHQL) drivers
<i>Enterprise Manageability</i>	Software deployments (installation, updates), task delegation, printing, Windows interoperability

Table 2.5-1: Red Hat Enterprise Linux



Red Hat Enterprise Linux 6 improves Windows Server 2008 R2 interoperability with Samba updates including support for trust relationships, Windows cross-forest, transitive and one-way domain trusts.



3 Reference Architecture Environment

This reference architecture environment uses IBM blades and EMC Celerra storage for the hardware. The software consists of CloudForms V1 and JBoss EAP 6 Beta1 suite consisting of the JBoss Enterprise Web Server (EWS), JBoss Operations Network Server (JON), JBoss Enterprise Application Server (EAP), JON plugin pack and PostgreSQL for the back-end database. All of the images deployed are running Red Hat Enterprise Linux 6.2 x86_64.

Figure 3.1: Reference Architecture Infrastructure depicts how the environment is set up. The goal is to have traffic come into the EWS and then load balance traffic to the JBoss EAP host controller nodes which store data in the back-end PostgreSQL database. The supporting infrastructure for this is an existing CloudForms environment consisting of a System Engine and Cloud Engine server. This reference architecture assumes that the JON server has been pre-configured and is accepting registration requests. Finally, for post boot configuration there is a CloudForms Configuration Server that executes all the scripts for this reference architecture. All of the virtual machines are hosted on a Red Hat Enterprise Virtualization 3.0 environment which has been configured to use a floppy injector hook to pass the user data from the configuration server to the images.

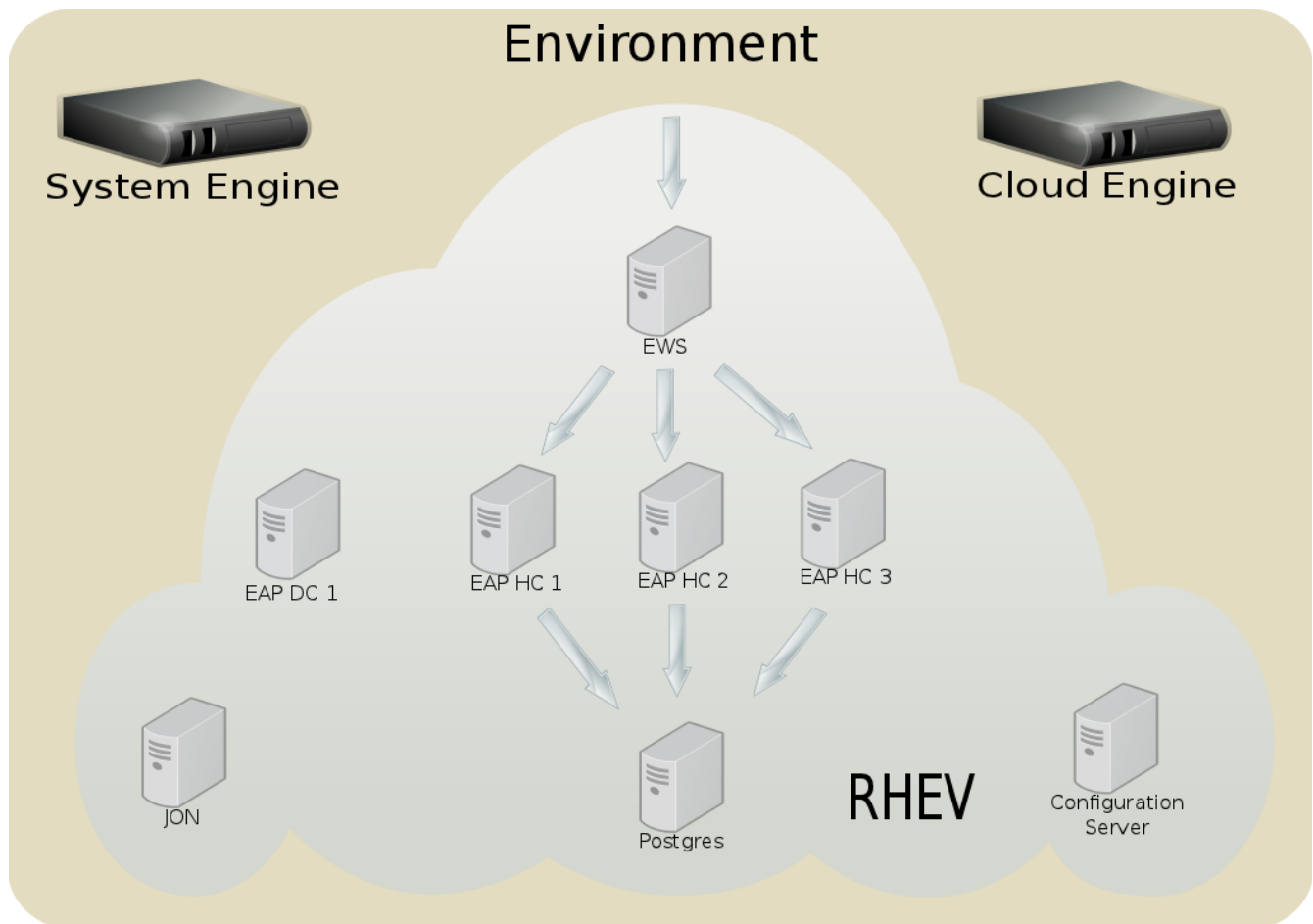


Figure 3.1: Reference Architecture Infrastructure



3.1 Hardware

All physical systems used the same hardware platform type:

Component	Description
Blade Chassis	IBM BladeCenter H - 8852HC1
Blade Server	IBM BladeServer – HS22 - 70870
CPU	2 x Intel Xeon X5680 (6 core @3.33 GHz)
Memory	52 GB
Network	2 x Broadcom Corporation NetXtreme II BCM5709S Gigabit Ethernet
Disk	2 x 146 GB SAS

Table 3.1-1: Physical Systems

All Virtual Machines were configure with the same settings (except disk sizes) from the same server.

Component	Description
CPU	2
Memory	4096 MB
Network	1 bridged virtIO
Disk	System Engine – 15 GB + 100 GB iSCSI RHEVM - 6/25 GB Window Client – 30 GB

Table 3.1-2: Virtual Machines

3.2 Storage

Non-local storage was provided by a EMC Celerra NS-120.

System	Disk Usage
RHEV Hypervisor	200 GB
System Engine	100 GB

Table 3.2-1: Storage

3.3 Software

The following table lists the software for the CloudForms components.



System	Software Version
System Engine	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Katello & major components: katello-0.1.311-1.el6_2.noarch candlepin-0.5.26-1.el6.noarch pulp-1.0.4-1.el6.noarch
Cloud Engine	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Aeolus Conductor and major components: aeolus-conductor-0.8.13-1.el6_2.noarch imagefactory-1.0.0rc11-1.el6.noarch iwhd-1.5-2.el6.x86_64 deltacloud-core-0.5.0-8.el6.noarch
Configuration Server	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Aeolus Configuration Server major component: aeolus-configserver-0.4.8-1.el6_2.noarch

Table 3.3-1: CloudForms Infrastructure

The following table lists the PostgreSQL components.

System	Software Version
PostgreSQL Server	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 PostgreSQL postgresql-server-8.4.9-1.el6_1.1.x86_64

Table 3.3-2: PostgreSQL

The following table lists the JBoss software components. This reference architecture uses the JBoss Beta 1 software stack.



System	Software Version
JBoss Operations Network (JON) Server	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Katello & major Components: katello-0.1.311-1.el6_2.noarch candlepin-0.5.26-1.el6.noarch pulp-1.0.4-1.el6.noarch
JBoss Enterprise Web Server	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Aeolus Conductor and major Components: aeolus-conductor-0.8.13-1.el6_2.noarch imagefactory-1.0.0rc11-1.el6.noarch iwhd-1.5-2.el6.x86_64 deltacloud-core-0.5.0-8.el6.noarch
JBoss Enterprise Application Platform Domain Controller	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Apache Maven Components: apache-maven-3.0.4 JBoss EAP Components: jboss-eap-6.0.0.Beta1 jboss-as-quickstarts-7.1.1.CR1-dist.zip jon-plugin-pack-tech-preview-3.0.1.GA jboss-eap-6.0.0.Beta1-maven-repository postgresql-9.1-901.jdbc3.jar
JBoss Enterprise Application Platform Host Controllers	Red Hat Enterprise Linux 6.2: 2.6.32-220.17.1.el6.x86_64 Apache Maven Components: apache-maven-3.0.4 JBoss EAP Components: jboss-eap-6.0.0.Beta1

Table 3.3-3: JBoss Software



The following table lists the Red Hat Enterprise Virtualization environment.

Cloud Provider	Configuration	Software Versions
Red Hat Enterprise Virtualization	<ul style="list-style-type: none">• One RHEL 6.2 Hypervisor• RHEL 6.2 VM (2 vCPU, 4 GB) for RHEVM• One Microsoft Windows 7 VM for access to RHEV Management Portal	RHEV Manager: 3.0.1_0001-4.el6 Red Hat Enterprise Virtualization Hypervisor Hosts: 6.2.0.3.el6 VDSM Version: 3.0.112.1

Table 3.3-4: RHEV Environment

The following table lists the scripts that are included in this Reference Architecture. These scripts are available on the Red Hat Customer Portal¹. All of the scripts place log files in the `/var/audrey/tooling` directory. More details on the scripts are provided inline as comments in the scripts themselves. There are several ways to combine the scripts below to accomplish the goals and these are simply just one proposal.



Script	System	Actions
<i>install_postgres.sh</i>	<ul style="list-style-type: none">PostgreSQL Server	<ul style="list-style-type: none">Initializes the postgresql databaseTells postgres to trust any host. Change this depending on the security requirementsTells postgres to listen on all interfaces, again, change depending on security requirementsEnables postgres on boot and starts serviceCreates database ksink
<i>register_se.sh</i>	<ul style="list-style-type: none">JBoss EWS ServerPostgreSQL ServerJBoss EAP Domain ControllerJBoss EAP Host Controllers	<ul style="list-style-type: none">Registers the system with the System Engine serverImports the Red Hat GPG key
<i>jboss-as</i>	<ul style="list-style-type: none">JBoss EAP Domain / Host Controllers	<ul style="list-style-type: none">Provides startup script for JBoss
<i>jboss_dc_config.sh</i>	<ul style="list-style-type: none">JBoss EAP Domain Controller	<ul style="list-style-type: none">Backs up the domain.xml fileInstructs the Domain Controller to listen on the public interface, not localhostConfigures the web service to assign unique identifiers so mod_cluster functionsInstructs maven to point to a local repository that is compatible with EAP Beta 1Configures the PostgreSQL datasourceInstalls and configures the PostgreSQL Drivers and module fileBuilds the kitchen sink application



<i>jboss_hc_config.sh</i>	<ul style="list-style-type: none">• JBoss EAP Host Controllers	<ul style="list-style-type: none">• Instructs the host controllers to run in host controller mode instead of domain controller mode• Exposes the management interface on the host controllers• Add base64 encoded password to the host.xml file• Initiates connection to the domain controller to add itself to the JBoss cluster
<i>jboss_eap_maven_jon_install.sh</i>	<ul style="list-style-type: none">• JBoss EAP Host Controllers• JBoss EAP Domain Controller	<ul style="list-style-type: none">• Pulls over all the software from the repository• Creates the directory structure for the software• Extracts the software into those directories• Configures maven environment variables• Sets up SSH keys• Deploys and starts the JON agent
<i>jboss_ews_deploy.sh</i>	<ul style="list-style-type: none">• JBoss EWS server	<ul style="list-style-type: none">• Pulls over EWS software and extracts it• Set up EWS user• Configures mod_cluster in Jboss_HTTP.conf• Creates service startup script for Apache and starts the service
<i>start_jboss.sh</i>	<ul style="list-style-type: none">• JBoss EAP Host Controllers• JBoss EAP Domain Controller	<ul style="list-style-type: none">• Pulls over the jboss-as startup script• Enables JBoss service to start every boot and then starts the service

Table 3.3-5: Configuration Scripts



The following table lists the XML services files that are included in this reference architecture. The following XML files are available on the Red Hat Customer Portal².

Services File	Scripts Called	Systems Affected
jboss_eap_dc_hc_node_files.xml	<ul style="list-style-type: none">• register_sh.sh• jboss_eap_maven_jon_install.sh• jboss_dc_config.sh• jboss_hc_config.sh• jboss_jon_agent_config.sh• start_jboss.sh	<ul style="list-style-type: none">• JBoss EAP Host Controllers• JBoss EAP Domain Controller
jboss_ews_nodes_files.xml	<ul style="list-style-type: none">• jboss_ews_deploy.sh• register_se.sh• jboss_ews_install.sh	<ul style="list-style-type: none">• JBoss EWS server
jboss_postgres_files.xml	<ul style="list-style-type: none">• register_se.sh• install_postgres.sh	<ul style="list-style-type: none">• PostgreSQL Server

Table 3.3-6: Configuration Server Services Files



4 CloudForms Infrastructure Prerequisites

This section describes the prerequisites that must be met for the environment. It is assumed that there is an existing System Engine, Cloud Engine and JON server. Any changes from the default install are listed in this section. This section sets the foundation for the rest of the deployment in that it makes sure that the proper infrastructure is set up before proceeding. The checklist below lists deployment actions that are required to take place, in the order that they take place.

Pre-Requisites:

- ☐ JON Server
- ☐ System Engine Serve
- ☐ Cloud Engine Server

4.1 Preparing System / Cloud Engine Servers

This reference architecture assumes that an existing CloudForms infrastructure has been setup complete with a System Engine and a Cloud Engine server. For assistance getting this environment up and running, refer to *Red Hat CloudForms v1 Infrastructure and Application Deployment Fundamentals*³. The configuration of the CloudForms environment for this reference architecture is as follows:

System Engine:

- Create *refarch* organization
- Create *dev / test / prod* environments
- Select and Synchronize the Red Hat CloudForms Cloud Engine Channel
- Create and download a Manifest that
 - Contains CloudForms Channel (Synchronize the Repository)
 - Contains CloudForms Tools Repository
 - Contains CloudForms Cloud Engine Repository
 - Contains Red Hat Enterprise Linux Channel (Synchronize the Repository)

Cloud Engine:

- Configure a Red Hat Enterprise Virtualization provider and the associated accounts
- Create a **JBoss** Cloud and associate the RHEV account to this cloud
- Create a **JBossCRZ** Cloud Resource Zone
- Create a **JBoss** catalog and associated it with the JbossCRZ cloud resource zone



- Create a *target_content.xml* that installs the *rhev-agent*, *aeolus-audrey-agent* and the *katello-agent*
- Create a larger Cloud Resource Profile with 4GB of memory and 2 processors for the JBoss EAP domain / host controllers.

4.2 Preparing the Red Hat Enterprise Virtualization Environment

The Red Hat Enterprise Virtualization environment requires floppy injector support to pass user-data from the configuration server to the instances being deployed on the RHEV environment. The requirements for this are a floppy injector package and a RHEL/KVM based RHEV hypervisor to support the necessary hook. To deploy the floppy injector hook, follow these instructions.

Red Hat Enterprise Virtualization Hypervisor:

1. Add the CloudForms tools channel

```
# rhn-channel --add --channel=rhel-x86_64-server-6-cf-tools
```

2. Install the *vdsm-hook-floppyinject* package

```
# yum install vdsm-hook-floppyinject
```

Red Hat Enterprise Virtualization Manager:

1. Update the Red Hat Enterprise Virtualization Manager database

```
# rhevm-config -s UserDefinedVMProperties='floppyinject=^.*:.*$' --cver=3.0
```

2. Restart the *jbossas* service

```
# service jbossas restart
```

Confirm Functionality:

1. Use **curl** to confirm via the API that the floppy injector is installed.

```
# curl -k -H "Accept: application/xml" -u admin@internal https://cf-rhevm.cloud.lab.eng.bos.redhat.com:8443/api/capabilities > floppy_injector.out
Enter host password for user 'admin@internal':
% Total    % Received % Xferd  Average Speed   Time    Time     Time
Current                                  Dload  Upload   Total   Spent    Left
Speed
100 28508    0 28508    0    0  24984      0 --:--:--  0:00:01 --:--:--
31816
```

2. Check the *<custom_properities>* XML element and look for the *floppyinject* entry. If it is not there, it was not installed properly.

```
<custom_properities>
  <custom_property regexp="^(true|false)$" name="sap_agent"/>
  <custom_property regexp="^[0-9]+$" name="sndbuf"/>
```




```
<custom_property regexp="^(([a-zA-Z0-9_]*):(true|false))((,(([a-zA-Z0-9_]*):(true|false)))*)*$" name="vhost"/>
<custom_property regexp="^(none|writeback|writethrough)$"
name="viodiskcache"/>
<custom_property regexp="^.*:.*$" name="floppyinject"/>
</custom_properties>
```



5 Preparing the JBoss Infrastructure

This section describes how to prepare the JBoss infrastructure once the CloudForms infrastructure has been configured. The first thing to do when preparing the JBoss infrastructure is to create the appropriate system templates for the systems to be deployed. The system templates are outlines of what software is to be installed on the systems. This section assumes that the content has already been promoted as per section **Preparing System / Cloud Engine Servers**. Secondly, the *target_content.xml* file is configured to match systems that are being deployed. The *target_content.xml* is used to ensure that the *katello-agent*, *rhev-agent* and *aeolus-audrey-agent* are installed on each image that gets built. For more information about the *target_content.xml* setup refer to the *Red Hat CloudForms v1 Infrastructure and Application Deployment Fundamentals*⁴. Unless otherwise noted, perform the actions in this Reference Architecture as the admin user for both System Engine and Cloud Engine. The below checklist provides the steps that are necessary to proceed.

Deployment ToDo:

- ☐ Create Templates
- ☐ Download Scripts
- ☐ Extract Scripts
- ☐ Download JBoss Software
- ☐ Run `replace_hostname.sh`
- ☐ Deploy EWS Server
- ☐ Deploy PostgreSQL Server
- ☐ Deploy JBoss Host Controller
- ☐ Deploy JBoss Domain Controllers
- ☐ Deploy Application
- ☐ Test Application



5.1 Prepare System Templates

The system templates are prepared in the System Engine web based user interface. The prerequisites for configuring system templates are that the Red Hat Enterprise Linux and CloudForms content has been synchronized and there are environments to which the content and system templates can be promoted. This Reference Architecture is only concerned with the *Library* and *Dev* environments. There are two options to promote the system templates, the first is to create the templates and promote one at a time. The second way is to create all the templates and then create one *changeset* and promote all the templates at one time. The latter is the preferred method and the one used in this Reference Architecture. The system templates in **Table 5.1-1: System Templates** are to be created.

System Template	Description
JBoss_Config_Server	Template for the configuration server, contains aeolus-configserver
JBoss_EWS_EAP_Server	Template for EWS and EAP host / domain controllers, contains openjdk
JBoss_PostgreSQL_Server	Template for the PostgreSQL server, contains postgres-server

Table 5.1-1: System Templates

5.1.1 JBoss EAP / EWS Server Template

The first system template to be created is used to build the image for both JBoss EWS and the JBoss EAP servers. In addition, the **jboss_eap_ews_template** can be used as a outline for the JBoss EAP Server Domain Controller and Host Controllers. This template contains the *Red Hat Enterprise Linux 6.2 x86_64* product, *java-1.6.0-openjdk* and *java-1.6.0-openjdk-devel* packages. The rhev-agent and the aeolus-audrey-agent are installed at build time with the *target_content.xml*. In summary, the **jboss_eap_ews_template** is responsible for three images: JBoss EWS, EAP (DC and HC's).

Create the **jboss_eap_ews_template** template by following these steps:

1. Log into the System Engine user interface: <https://cf-se2/katello>
2. Click on the **Content Management** tab
3. Click on the **System Templates**
4. Click **+New Template**
5. On the right hand navigation pane, provide a name and a description. The example name for the EWS template is: *jboss_ews_eap_template*
6. Click **Save**
7. On the left hand navigation pane, in the **Eligible Content** section, select **Red Hat Enterprise Linux Server x86_64**
8. Select **Repositories**
9. Click **+ Add** for **Red Hat Enterprise Linux 6 Server RPMs x86_64 6Server** and **Red**



Hat CloudForms Tools for RHEL 6 RPMs x86_64 6Server

10. Click **Packages** in the right navigation pane
11. In the **Search available packages** field type:
 - **java-1.6.0-openjdk**
 - **java-1.6.0-openjdk-devel**
 - **httpd**

For the openjdk-devel package full name search may be required.
12. Click **+ Add** for all
13. Click the **jboss_ews_eap_template** in the upper navigation section of the **System Templates** pane
14. Click **Selected Distribution** in the **System Templates** navigation pane
15. Select **ks-Red Hat Enterprise Linux-Server-6.2-x86_64**
16. Click **Save**

5.1.2 PostgreSQL Template

The second system template to be created is the PostgreSQL template. This template is used to create the image that will host the database that the JBoss Host Controllers send their data.. This template contains Red Hat Enterprise Linux 6.2 x86_64 product and the *postgresql-server* package. The rhev-agent and the aeolus-audrey-agent are installed at build time with the *target_content.xml*.

Create the **postgresql_template** template by following these steps:

1. Log into the System Engine user interface: <https://cf-se2/katello>
2. Click on the **Content Management** tab
3. Click on the **System Templates** breadcrumb
4. Click **+New Template**
5. On the right hand navigation pane, provide a name and a description. This paper calls the PostgreSQL template: **postgresql_template**
6. Click **Save**
7. On the left hand navigation pane, in the **Eligible Content** section, select **Red Hat Enterprise Linux Server x86_64**
8. Select **Repositories**
9. Click **+ Add** for **Red Hat Enterprise Linux 6 Server RPMs x86_64 6Server**
10. Click **Packages** in the right navigation pane
11. In the **Search available packages** field, type in *postgresql-server*
12. Click **+ Add**



13. Click the **postgresql_template** in the upper navigation section of the System Templates pane
14. Click **Selected Distribution** in the **System Templates** navigation pane
15. Select **ks-Red Hat Enterprise Linux-Server-6.2-x86_64**
16. Click **Save**

5.1.3 Configuration Server Template

The third system template to be created is the Audrey configuration server template. This template contains the *Red Hat Enterprise Linux 6.2 x86_64* product, *Red Hat CloudForms Cloud Engine Beta RPMs x86_64 6Server* and the *aeolus-configserver* package. This is also the first server deployed into the environment as all the other nodes require its functionality for post boot configuration. In order to create this template follow these steps.

Create the **config_server_template** template by following these steps:

1. Log into the System Engine user interface: <https://cf-se2/katello>
2. Click on the **Content Management** tab
3. Click on the **System Templates** breadcrumb
4. Click **+New Template**
5. On the right hand navigation pane, provide a name and a description. This paper calls the Configuration Server template: **config_server_template**
6. Click **Save**
7. On the right hand navigation pane, select the **config_server_template**
8. Enable Repositories
 1. On the left hand navigation pane, in the **Eligible Content** section, select **Red Hat Enterprise Linux Server x86_64**
 2. Select **Repositories**
 3. Click **+ Add** for **Red Hat Enterprise Linux 6 Server RPMs x86_64 6Server**
 1. Repeat this process for the following repository: **Red Hat CloudForms Cloud Engine Beta RPMs x86_64 6Server**
9. Click **Packages** in the right navigation pane
10. In the **Search available packages** field, type in *aeolus-configserver*
11. Click **+ Add**
12. Click the **config_server_template** in the upper navigation breadcrumb section of the System Templates pane
13. Click **Selected Distribution** in the **System Templates** navigation pane
14. Select **ks-Red Hat Enterprise Linux-Server-6.2-x86_64**
15. Click **Save**



5.1.4 Promote the Templates

Promote the **jboss_eap_ews_template**, **postgresql_template** and the **config_server_template** by following these steps:

1. Click on the **promotions** link in the upper navigation menu
2. Click **+ New Promotion Changeset**
 1. The promotion takes place from the *Library* environment to the *dev* environment
3. Provide a name and a description. This example uses **template_promotion** as the name
4. Click **Save Changeset**
5. Click **System Templates** in the left navigation pane titled Library Content
6. Select **+ Add** for the **jboss_eap_ews_template**, **postgresql_template** and the **config_server_template**
7. Click **Review** and then **Promote**

After the promotion is complete, download the templates by following these steps:

1. Click **System Templates** in the upper navigation menu
2. Select the **jboss_eap_ews_template** in the right navigation pane titled System Templates
3. Click **Download**
4. Change the environment to *dev* and click **Download**
5. Save the file to a location that is accessible by the Cloud Engine server
 - Repeat this process for the remaining two templates.
6. Once the template is downloaded, it can be formatted with the following command

```
$ xmllint -format template_name_here.xml -o template_name_here.xml
```

5.2 Prepare Target Content

Place a *target_content.xml* file in the */etc/imagfactory* directory on the Cloud Engine server. For information on how to create a *target_content.xml* file, please refer to *Red Hat CloudForms v1 Infrastructure and Application Deployment Fundamentals*⁵.



6 Configuration Server

The **Audrey Configuration Server** provides post boot configuration for AppForms that are deployed to providers. This Reference Architecture provides scripts and *services.xml* files that the configuration server uses to do this post boot configuration. In addition to the scripts and *services.xml* files the JBoss software and quickstart applications should be downloaded and organized as per the following section. The following section makes suggestions as to how and where to host the content to take advantage of this functionality.

6.1 Obtain Software

The software can be downloaded using the links in **Table 6.1-1: Software and Scripts**. The following section describes how to organize the software. The software should be placed on the System Engine server.

JBoss Software
JBoss Enterprise Web Server ⁶
JBoss Plugin Pack Tech Preview ⁷
JBoss Application Server Quick Starts ⁸
JBoss Enterprise Application Server 6.0 Beta 1 ⁹
JBoss Maven Repository ¹⁰
Apache
Apache Maven ¹¹
PostgreSQL
PostgreSQL JDBC3 Driver ¹²
Scripts
Reference Architecture Scripts ^{13 1}

Table 6.1-1: Software and Scripts

6.2 Set up repository for scripts and XML files

The System Engine server hosts all the scripts and XML files that the Configuration Server uses when deploying the instances.

Create a repository on the System Engine server to host the scripts. Download the scripts that are associated with this Reference Architecture from the Red Hat customer portal¹⁴.

1. On the System Engine server, unzip the files.

```
# cd /var/www/html/pub/  
# tar xzvf csrepo_v1.tgz
```

- Note, create new SSH keys to place in the `domain_controller_ssh_keys` directory.

¹ Contains an EXAMPLE directory with a complete copy of every file that is touched by the scripts.



These keys are used to automate the host controller / domain controller actions. To create a new set of keys use the following command on the System Engine server once the directory structure is intact:

```
# ssh-keygen -t rsa -f
/var/www/html/pub/csrepo/domain_controller_ssh_keys/id_rsa -P ''
```

Now create a *config* file and a *authorized_hosts* file.

```
# cat /var/www/html/pub/csrepo/domain_controller_ssh_keys/id_rsa.pub >
authorized_keys
# cat > /var/www/html/pub/csrepo/domain_controller_ssh_keys/config << EOF
> StrictHostKeyChecking=no
> UserKnownHostsFile=/dev/null
> EOF
```

2. This is what the directory structure should look like when populated. The scripts and content directories contain scripts that can be obtained from <http://access.redhat.com>¹⁵

```
# ls -R /var/www/html/pub/csrepo/
/var/www/html/pub/csrepo/:
content  domain_controller_ssh_keys  scripts  xml

/var/www/html/pub/csrepo/content:
jboss_beta1

/var/www/html/pub/csrepo/content/jboss_beta1:
apache-maven-3.0.4-bin.tar.gz          jboss-ews-1.0.2-RHEL6-x86_64.zip
maven_repo
jboss-as-quickstarts-7.1.1.CR1-dist.zip  jon_agents
postgres
jboss-eap-6.0.0.Beta1.zip              jon-plugin-pack-tech-preview-
3.0.1.GA.zip

/var/www/html/pub/csrepo/content/jboss_beta1/jon_agents:
rhq-enterprise-agent-3.0.1.GA.jar

/var/www/html/pub/csrepo/content/jboss_beta1/maven_repo:
jboss-eap-6.0.0.Beta1-maven-repository.zip

/var/www/html/pub/csrepo/content/jboss_beta1/postgres:
module

/var/www/html/pub/csrepo/content/jboss_beta1/postgres/module:
module.xml  postgresql-9.1-901.jdbc3.jar

/var/www/html/pub/csrepo/domain_controller_ssh_keys:
authorized_keys  config  id_rsa  id_rsa.pub

/var/www/html/pub/csrepo/scripts:
domain_datasource.xml  install_postgres.sh  jboss_dc_config.sh
jboss_ews_deploy.sh    register_se.sh
domain_driver.xml      jboss-as              jboss_eap_maven_jon_install.sh
```




```
jboss_hc_config.sh    start_jboss.sh

/var/www/html/pub/csrepo/xml:
jboss_eap_dc_hc_node_files.xml  jboss_ews_nodes_files.xml
jboss_postgres_files.xml
```

```
# chown -Rv apache.apache /var/www/html/pub/csrepo/
changed ownership of `/var/www/html/pub/csrepo/xml' to apache:apache
changed ownership of `/var/www/html/pub/csrepo/scripts' to apache:apache
changed ownership of `/var/www/html/pub/csrepo/' to apache:apache
. . . < Output Truncated > . . .
```

3. Ensure that the following files exist in the `/var/www/html/pub/csrepo/scripts` directory

```
# ll /var/www/html/pub/csrepo/scripts/ | awk '{ print $9 }'

install_postgres.sh
jboss-as
jboss_dc_config.sh
jboss_eap_maven_jon_install.sh
jboss_ews_deploy.sh
jboss_hc_config.sh
register_se.sh
start_jboss.sh
```

4. Place the following XML files in the `/var/www/html/pub/csrepo/xml` directory

```
# ll /var/www/html/pub/csrepo/xml/ | awk '{ print $9 }'

jboss_eap_dc_hc_node_files.xml
jboss_ews_nodes_files.xml
jboss_postgres_files.xml
```

5. Each services XML file has an executable URL element that contains a path to the scripts. This is important to change and match the environment that this Reference Architecture is being deployed in. If the configuration server cannot pull the packages and scripts the deployment of the JBoss, PostgreSQL and EWS servers does not work.

```
<executable
url="http://your.server.here/pub/csrepo/scripts/jboss_hc_config.sh"/>
```

6. Place the following files in the `content` directory

```
# ls -R /var/www/html/pub/csrepo/content/
/var/www/html/pub/csrepo/content/:
jboss_beta1

/var/www/html/pub/csrepo/content/jboss_beta1:
apache-maven-3.0.4-bin.tar.gz          jboss-ews-1.0.2-RHEL6-x86_64.zip
maven_repo
jboss-as-quickstarts-7.1.1.CR1-dist.zip  jon_agents
```



```

postgres
jboss-eap-6.0.0.Beta1.zip          jon-plugin-pack-tech-preview-
3.0.1.GA.zip

/var/www/html/pub/csrepo/content/jboss_beta1/jon_agents:
rhq-enterprise-agent-3.0.1.GA.jar

/var/www/html/pub/csrepo/content/jboss_beta1/maven_repo:
jboss-eap-6.0.0.Beta1-maven-repository.zip

/var/www/html/pub/csrepo/content/jboss_beta1/postgres:
module

/var/www/html/pub/csrepo/content/jboss_beta1/postgres/module:
module.xml  postgresql-9.1-901.jdbc3.jar

```

6.3 Scripts and XML files

The scripts and XML files are used by the Configuration Server to do post boot configuration of the instances. The launch process in Conductor will scan the Application Blueprint for any `<services>` element. If the `<services>` element is included in the XML the “Configure Launch Time Parameters” page is enabled and parameters can be passed through the XML to the AppForms. The following section describes this process in greater detail.

6.3.1 Services

1. The conductor launch process reads the Application Blueprint which contains the XML that has elements listed in the **Table 6.3.1-1: Services File XML**.

An example **services.xml** file...

```

<services>
  <service name="Service_Name_Here">
    <executable url="http://path/to/url/script_goes_here.sh"/>
      <parameters>
        <parameter name="password_parameter" type="password">
          <value><![CDATA[default_password_here]]></value>
        </parameter>
        <parameter name="non_password_parameter" type="scalar">
          <value><![CDATA[default_value]]></value>
        </parameter>
      </parameters>
    </service>
    <service name="inline_script">
      <executable>
        <contents><![CDATA[
inline script content goes here
]]></contents>
      </executable>
      <files>
        <file>
          <contents filename="file_to_create"><![CDATA[
file contents go here
]]></contents>

```



```
</file>
</files>
</service>
</services>
```

XML Element	Description
services	Parent element for services XML file.
service name	The name of the service. There can be many services in a services XML file. Each service can contain executable URL, files, etc...
executable url	Path to script located on an accessible system. Optionally, the executable code can be placed in-line here.
parameters	Variables that can be passed from configuration server to the scripts when run. There are two types of parameters: password and scalar. Password parameters are masked, scalar parameters are not.
parameter name	The name and type of parameter – password or scalar
value	Optional, can provide a default value for a parameter. For example, this is helpful if you register to the same System Engine server for every deployment, but want the option to register to a different one.
files	Parent element for the file element
file	Allows creation of a file on a file system. Place inline content here to create a file on the Appform that is being deployed. Optionally, the file can be accessed via URL.

Table 6.3.1-1: Services File XML



The “Configure Launch Time Parameters” screen accepts parameters for each service, that are passed as variables to the instances in the following format:

\$AUDREY_VAR_serviceName_parameterName

Variable Section	Description
AUDREY_VAR	Required at the beginning of every variable
serviceName	The name of the service associated with the parameterName
parameterName	The name of the parameter with the data passed via the “Configure Launch Time Parameters” screen in the Conductor UI.



6.4 Deploy Configuration Server

The CloudForms Configuration Server provides post boot configuration for deployed instances. There are two components to the configuration server. The first is the configuration server itself. The configuration server is deployed as a virtual machine to one of the providers. After the configuration server is deployed it needs to be configured. To configure the configuration server, login and run **aeolus-configserver-setup**. After the configuration server is configured the conductor auth key and secret need to be recorded as these keys are used to add the configuration server to the CloudForms Cloud Engine server later. The second component of the configuration server is the *Audrey Agent*. The *Audrey Agent* is deployed to each instance and invoked on boot up. The agent obtains the user data one of three ways, depending on the provider that the virtual machine is running on.

Provider	User Data Pass-through Method
RHEV	Floppy Injection
VMware	ISO
EC2	URL

Table 6.4-1: Configuration Server User Data

One of the features of configuration server include passing parameters between instances in a multi-appform environment. This feature is still in tech-preview and is not used in this reference architecture. More information is provided on deploying configuration server in section **Deploy Configuration Server**.

The configuration server is used to provide post boot configuration for the instances that are deployed.

6.4.1 Build and Push Configuration Server

1. Log into the Cloud Engine interface
2. Click on **Clouds**
3. Click on **New Image** for the JBoss Cloud

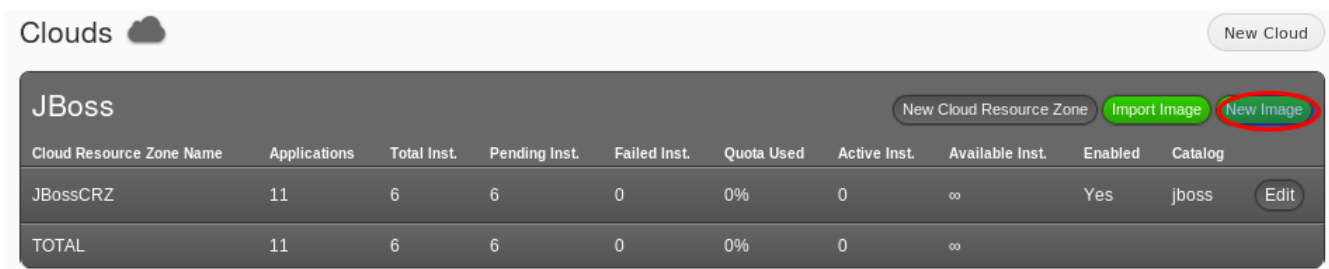


Figure 6.4.1: New Image



4. Provide a name and component outline. This reference architecture uses the name of **configuration_server**.
 1. Browse to the *config_server_template-dev-export.xml* component outline and click **Continue**

New Image

Choose one of the following options to upload or import a component outline into "JBoss Cloud".

Upload From URL

Name:

Choose a component outline file: [Browse...](#)

Edit this file before saving: ☒

[Cancel](#) [Continue](#)

Figure 6.4.2: Choose Component Outline

5. Verify the XML and click **Save and Continue**

configuration_server [Return to Cloud](#)

Edit

```
<?xml version="1.0" encoding="UTF-8"?>
<template>
  <name>configuration_server</name>
  <os>
    <name>RHEL-6</name>
    <version>2</version>
    <arch>x86_64</arch>
    <install type="url">
      <url>http://cf-se2.cloud.lab.eng.bos.redhat.com/pulp/ks/refsarch/dev/content/dist/rhel/server/6/6Server/x86_64/os/</url>
    </install>
    <rootpw>redhat</rootpw>
  </os>
  <description/>
  <packages>
    <package name="rhev-agent"/>
    <package name="katello-agent"/>
    <package name="aeolus-configserver"/>
  </packages>
  <repositories>
```

[Cancel](#) [Save and Continue](#)

Figure 6.4.3: Save and Continue



- Click **Save Component Outline**. Nothing has to be changed here. The root password can be changed if necessary.

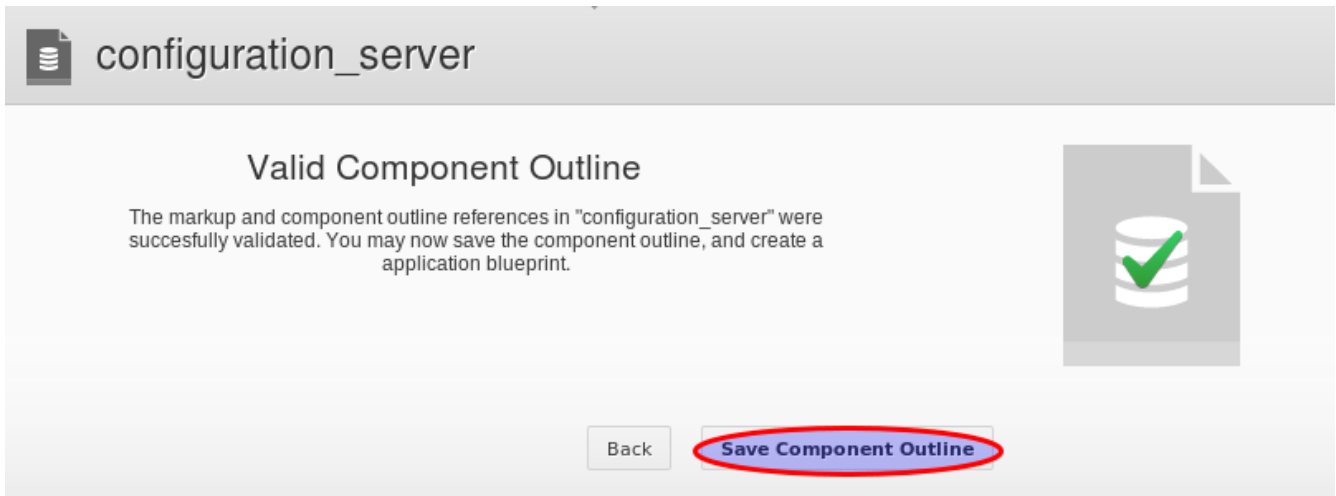


Figure 6.4.4: Save Component Outline

- Click **Build** for the RHEV provider

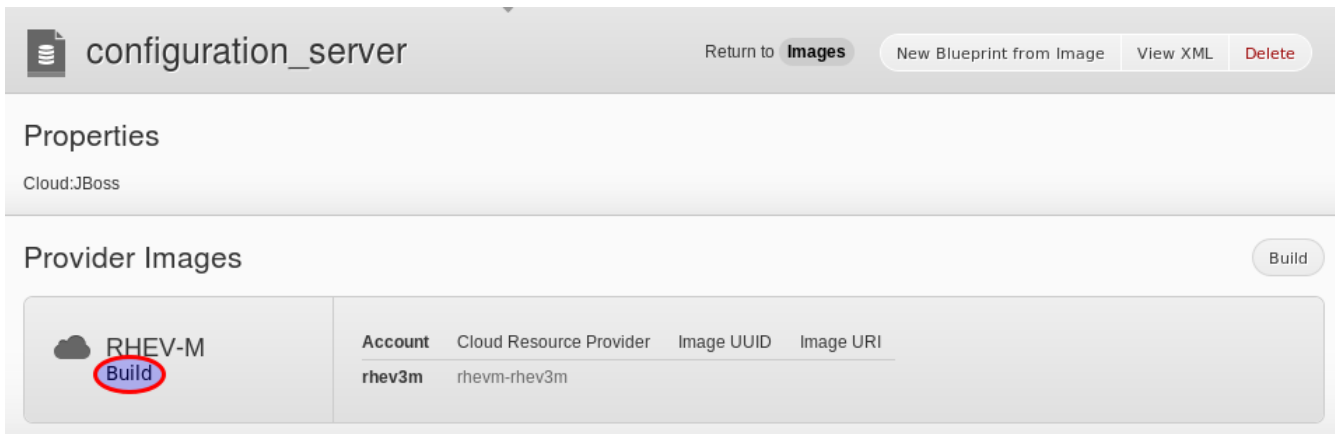


Figure 6.4.5: Build the Image

- Once the build is complete, click **Push**

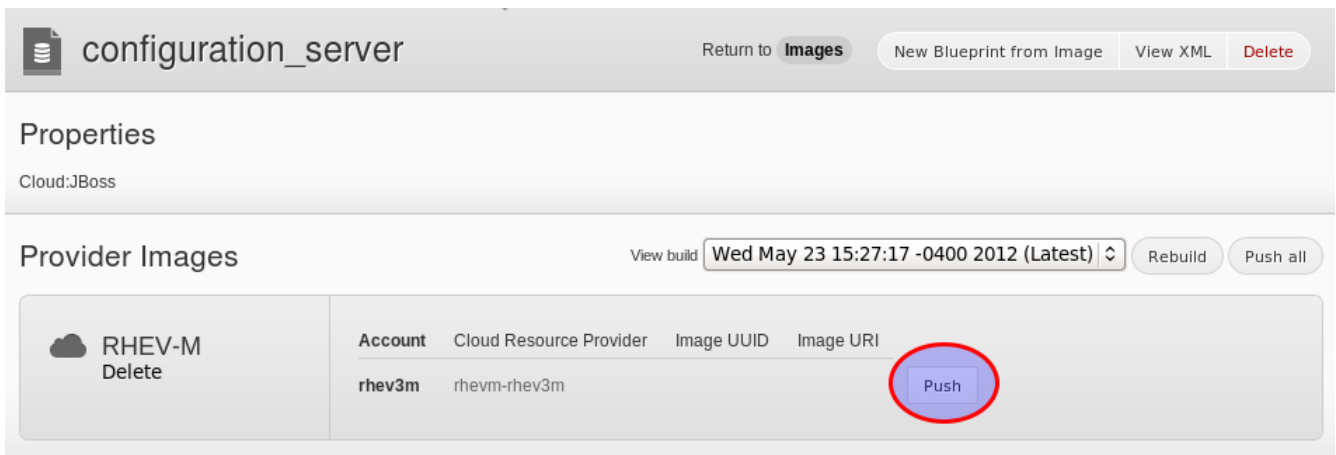


Figure 6.4.6: Push the Image



9. Once the push is complete, click **New Blueprint from Image**

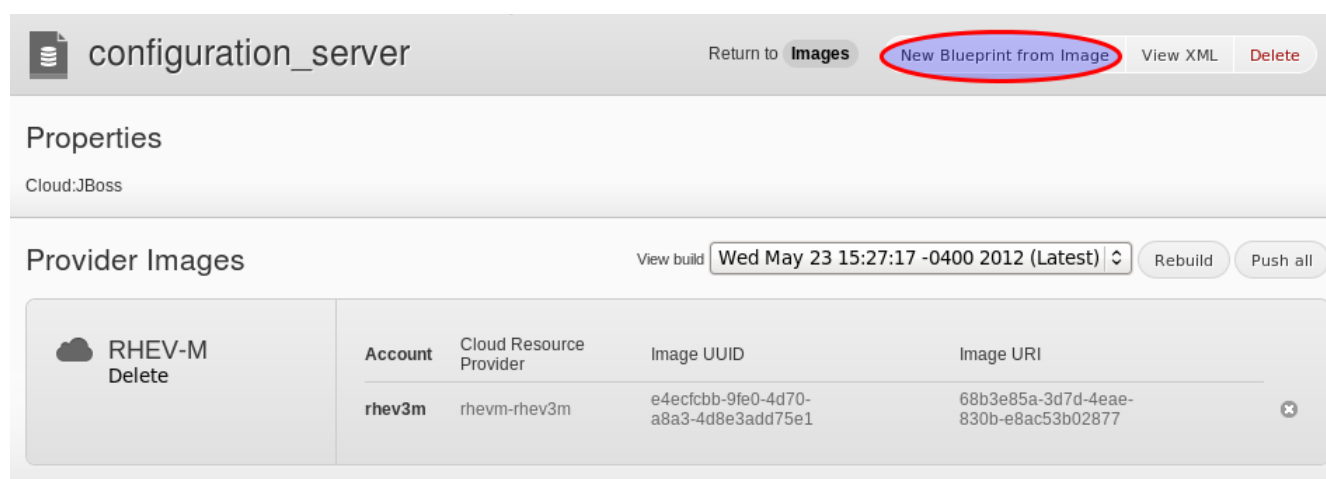


Figure 6.4.7: Create a New Blueprint

10. Choose the catalog and click **Save**

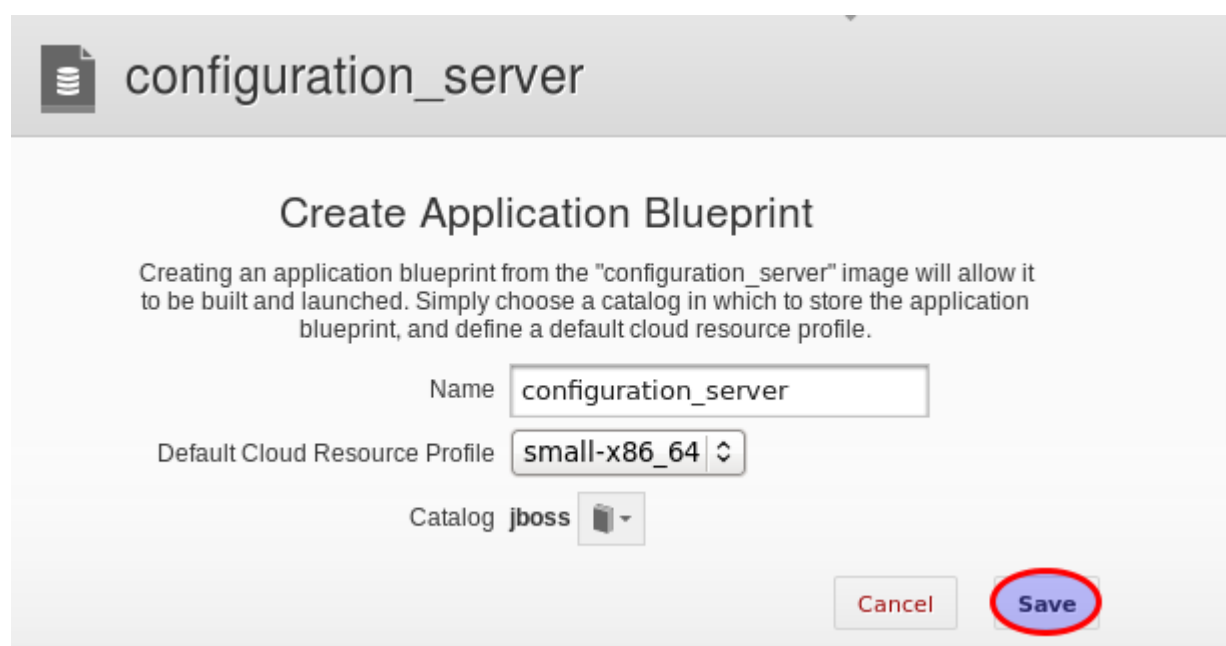


Figure 6.4.8: Save the Blueprint



6.4.2 Launch Configuration Server

1. Click on the **Monitor** tab
2. Expand the **JBossCRZ** cloud resource zone and click **New Application**

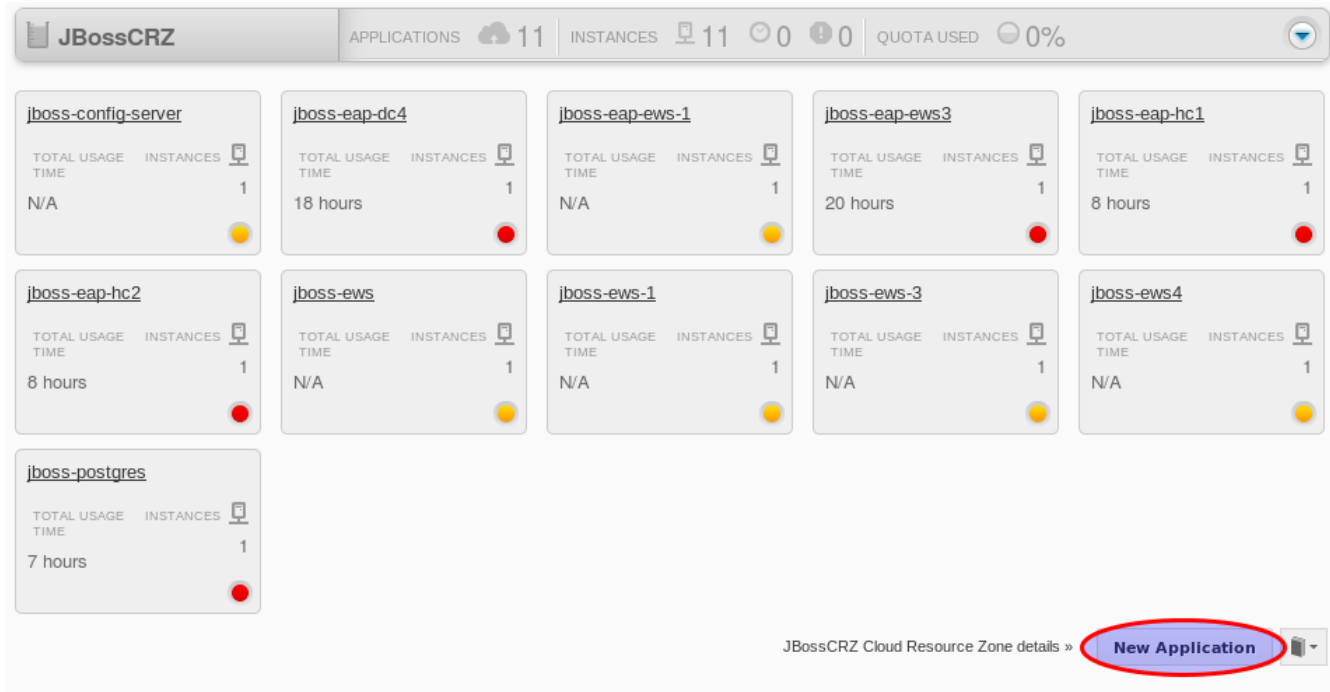



Figure 6.4.9: New Application



3. Provide an application name and select the *configuration_server* Application Blueprint. This reference architecture uses *config_server_application* as the name and click **Next**

 New Application to Jbosscloud Cloud Resource Zone

Application Details

Application Name (uniquely identifies your application)

Name available

Application Blueprint

Cloud Resource Cluster

Next

Figure 6.4.10: Select the Application Blueprint



4. Click **Launch**

Deploy
"config_server_application" to
JBossCRZ Cloud Resource
Zone?

Are you sure you wish to deploy "config_server_application" to the JBossCRZ Cloud Resource Zone? Doing so will utilize 0% of your overall application quota.

Name

Application Blueprint XML
[jboss-config-server](#) ⇒

Cloud Resource Cluster

Image IDs
82fbf37a-a37d-11e1-9f81-001a64760008

Figure 6.4.11: Launch the Application

6.4.3 Configure the Configuration Server

1. Log into the configuration server

```
# ssh root@configsvr
```

2. Run the setup command and take the defaults

```
# aeolus-configserver-setup
```

This script help you configure Apache as a proxy for a Config Server. Typically this is only useful if you are not familiar with Apache configurations and modules, specifically with mod_proxy, mod_auth_basic, and mod_ssl.

Also, this configuration tool assumes that you are not currently running Apache for any purposes on this server. This configuration tool create a Named Virtual Host for *:443. If this server is currently using Apache to serve secure pages on port 443, then this tool should not be used.

Do you wish to continue [y/N]: y

Please provide the web application URL where the Config Server is currently running on this server. If the Config Server was installed from an RPM, then this typically be:



```
http://localhost:4567/
```

The provided URL should be a fully qualified URL, providing the scheme, hostname, and port: `http://HOSTNAME:PORT/`

Enter the application URL [`http://localhost:4567/`]:

Root context: `/`

App URL: `http://localhost:4567/`

Conductor Auth Key: 230177184032792625073537

Conductor Auth Secret: 6YJmVjHPUe0JRKDBY5bJu6mSPg35xodrxibuWmtGUCw1bNSy

`\n\n*** You need to add this config server information to a ***`

`*** provider account in conductor. ***`

running: `echo | /usr/bin/puppet --modulepath /usr/share/aeolus-`

. . . < Output Truncated > . . .

notice: `/Stage[main]/Apache::Ssl/Exec[cert]/returns: executed successfully`

notice: `/File[vhost-443]/ensure: created`

notice: `/Stage[main]/Apache::Base/Exec[graceful-apache]: Triggered 'refresh' from 7 events`

notice: `Finished catalog run in 23.98 seconds`

3. Take note of the Conductor Auth Key and Secret

6.4.4 Add the Configuration Server to Cloud Engine

1. Log into Cloud Engine and click on **Administer**
2. Click on **Cloud Resource Provider**

The screenshot shows the 'rhev-m-rhev3m' configuration page in the Cloud Engine Administer interface. At the top right, there is a 'Choose a provider:' dropdown menu with 'rhev-m-rhev3m' selected and a green 'ON' toggle switch. Below this is an 'Alerts' section with a red circle containing the number '0'. The main content area has tabs for 'Settings' (active), 'Connectivity', 'Accounts' (1), 'Provider Realms' (1), and 'Role Assignments' (1). Under the 'Settings' tab, there are four input fields: 'Cloud Resource Provider name' (rhev-m-rhev3m), 'Cloud Resource Provider URL' (http://localhost:3002/api), 'Cloud Resource Provider Type' (RHEV-M), and 'RHEV-M API_PROVIDER' (https://cf-rhev3m.cloud.lab.eng.bos.redhat.com:8443/api;324c). At the bottom right, there are three buttons: 'Save Changes', 'Test Connection', and 'Delete Provider'.

Figure 6.4.12: Choose the RHEV Provider

3. Choose the **RHEV** provider



4. Click on **Accounts**

rhev3m-rhev3m Choose a provider: rhevm-rhev3m ON

Alerts 0

Settings Connectivity **Accounts** 1 Provider Realms 1 Role Assignments 1

Delete New Account Viewing All Accounts 1

<input type="checkbox"/>	Account Name	Username	Cloud Resource Provider Name	Cloud Resource Provider Type	Priority	Quota Used	Quota Limit
<input type="checkbox"/>	rhev3m	admin@internal	rhev3m-rhev3m	RHEV-M		0%	unlimited

Figure 6.4.13: Accounts

5. Click on the account name **rhev3m**

rhev3m-rhev3m Choose a provider: rhevm-rhev3m ON

Alerts 0

Settings Connectivity **Accounts** 1 Provider Realms 1 Role Assignments 1

Delete New Account Viewing All Accounts 1

<input type="checkbox"/>	Account Name	Username	Cloud Resource Provider Name	Cloud Resource Provider Type	Priority	Quota Used	Quota Limit
<input type="checkbox"/>	rhev3m	admin@internal	rhev3m-rhev3m	RHEV-M		0%	unlimited

Figure 6.4.14: Choose RHEV Account

6. Click on **Add** by Config Server:

Account: rhev3m Return to **Cloud Resource Providers** Edit Test Connection

Properties

Properties for rhev3m
Running instances quota:Unlimited
Priority
Config Server: None **Add**

Cloud Resource Clusters

Viewing All Cloud Resource Clusters 1

Provider Realm Name	Available?
clus1	Yes

Figure 6.4.15: Add the Configuration Server



7. Provide the Server URL, Consumer Key and Consumer Secret. Use the key and secret from the output of `aeolus-configserver-setup`
8. Click **Save**

Config Server

Edit Config Server

Server Endpoint (URL)

Consumer Key

Consumer Secret

Figure 6.4.16: Add Key and Secret

9. Click **Test** by Configuration Server to ensure communication and authentication are working. A success message will appear at the top of the screen.

Account: rhev3m

Properties

Properties for rhev3m
Running instances quota: Unlimited
Priority

Config Server: https://dhcp-126 [Edit] [Test] [Delete]

Figure 6.4.17: Test Connectivity



7 Deploy JBoss Infrastructure

7.1 Overview

In order to deploy the JBoss infrastructure the servers have to be deployed in a certain order. The following list shows the proper method to deploy the servers.

1. Deploy the JBoss EWS Server
2. Deploy the PostgreSQL Server
3. Deploy the JBoss Domain Controller
4. Deploy the JBoss Host Controllers
5. Deploy the JBoss Quickstart Application

When the JBoss EWS server starts, the configuration is handled by the configuration server. The **mod_cluster** page is available but it does not yet show any registered hosts. The PostgreSQL server is also automatically configured with a *ksink* database and set to listen on all ports for all hosts, this can be changed in the scripts if desired. After the PostgreSQL server is deployed, then the JBoss Domain Controller and Host Controllers can be deployed. The host controllers use the PostgreSQL server as a datasource for the JBoss *kitchensink* quickstart application.

7.2 Deploy JBoss Enterprise Web Server

The first JBoss system to be deployed in this Reference Architecture is the JBoss EWS server. This server is used to load balance traffic to the EAP host controller servers. However, there is no traffic to the EWS server until the EAP host controller servers are up and members of the *other-server-group* which has a *full-ha* profile. For more information regarding the *other-server-group* and the different profiles available in JBoss EAP 6, refer to the JBoss online documentation¹⁶. Follow the steps below to deploy the EWS.



7.2.1 Build and Push JBoss Enterprise Web Server

1. Log into Cloud Engine <https://cf-cloudforms2/conductor/>
2. Click the **Administer** tab
3. Click **Clouds**
4. Click **New Image** for the JBoss Cloud

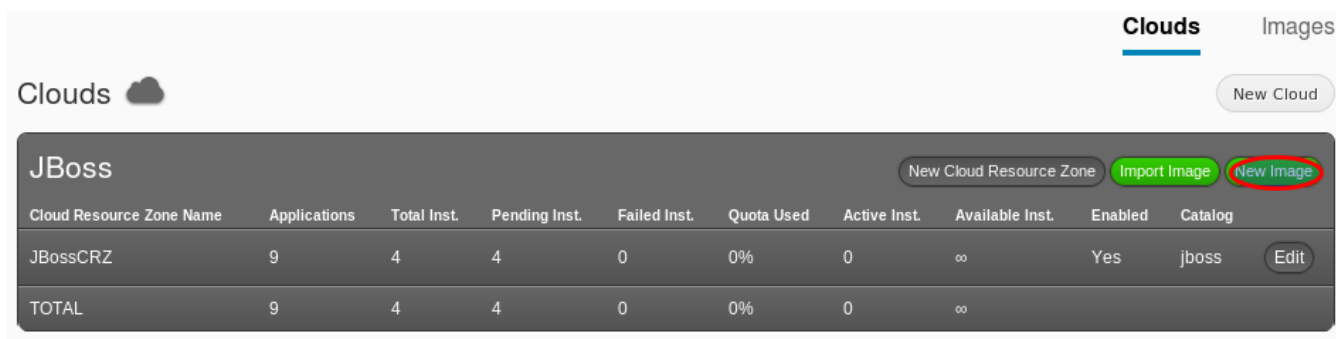


Figure 7.2.1: Create a New Image

5. Provide a name and component outline. This reference architecture uses the name of **jboss_ews**
 1. Browse to the *jboss_eap_ews_template-dev-export.xml* system template
 2. Click **Save and Continue**

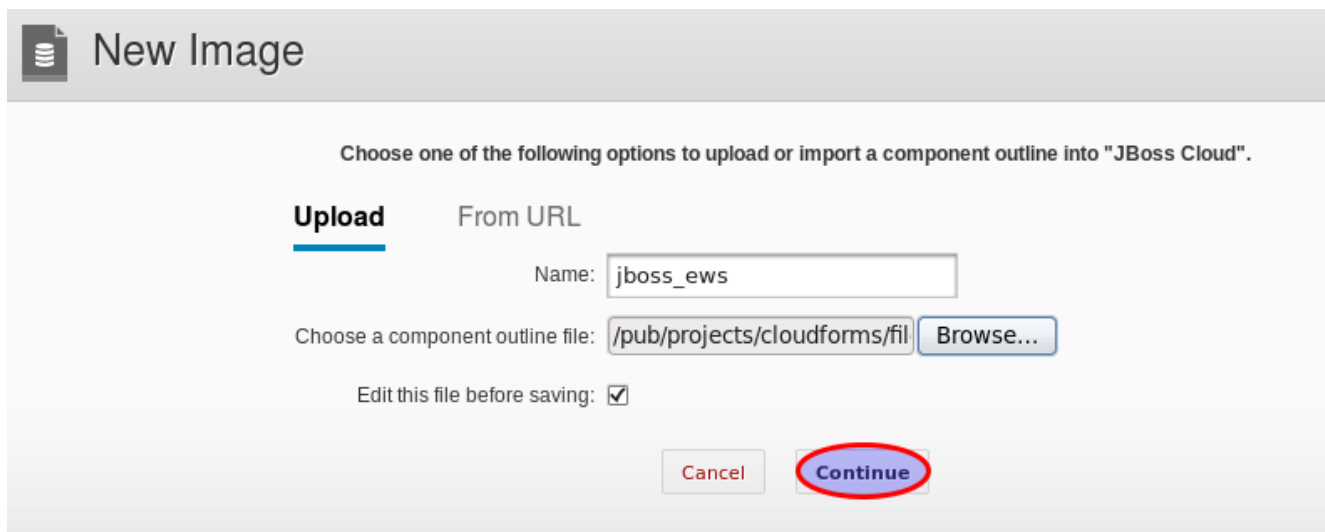


Figure 7.2.2: Choose Component Outline



6. Click **Save Component Outline**

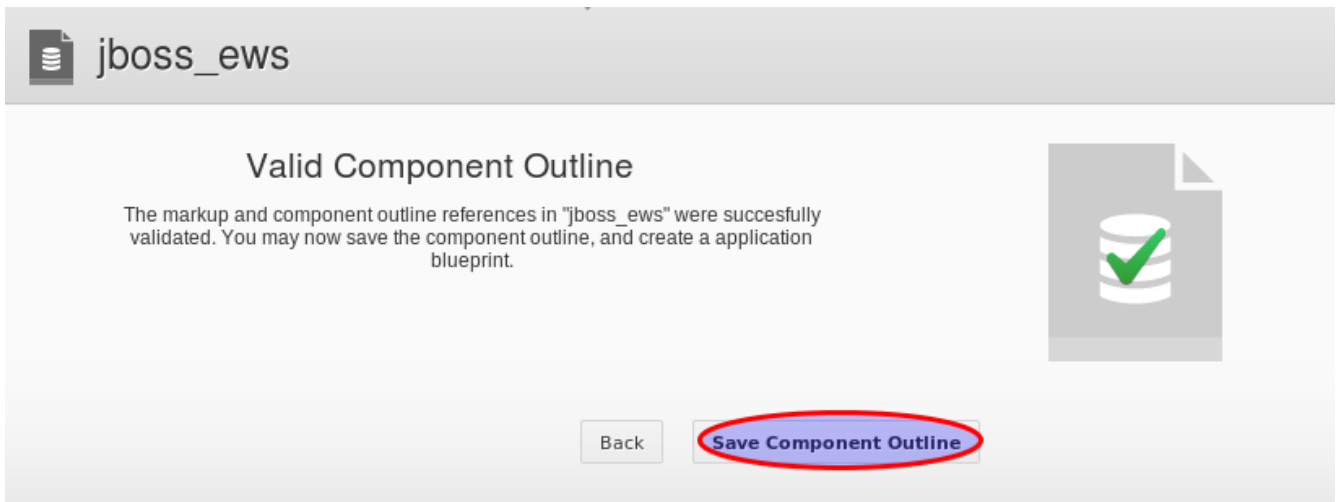


Figure 7.2.3: Save the Component Outline

7. Click **Build** for RHEV

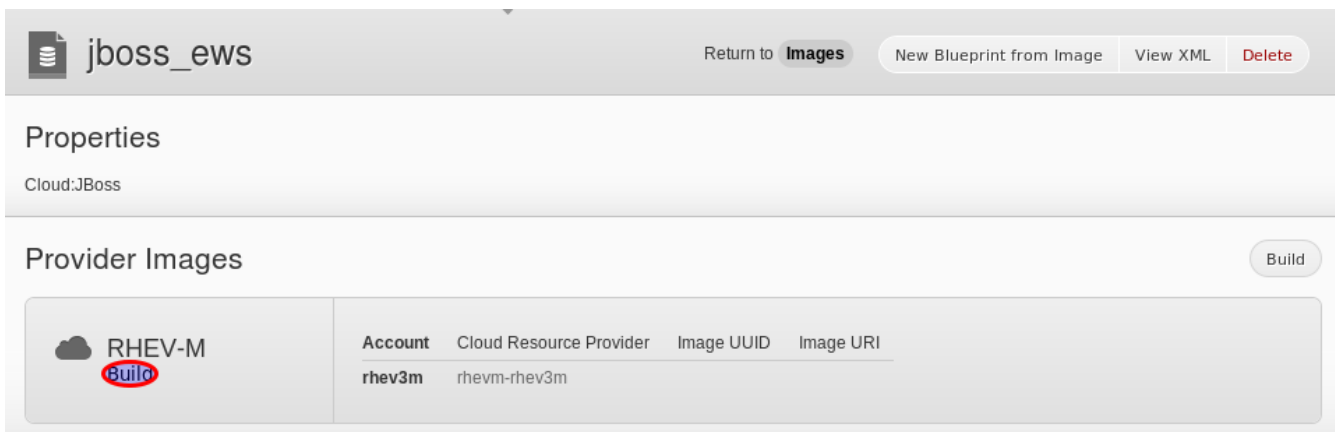


Figure 7.2.4: Build the Image



8. Once the build is complete, click **Push**



Figure 7.2.5: Push the Image

9. Once the push is complete, click **New Blueprint from Image**

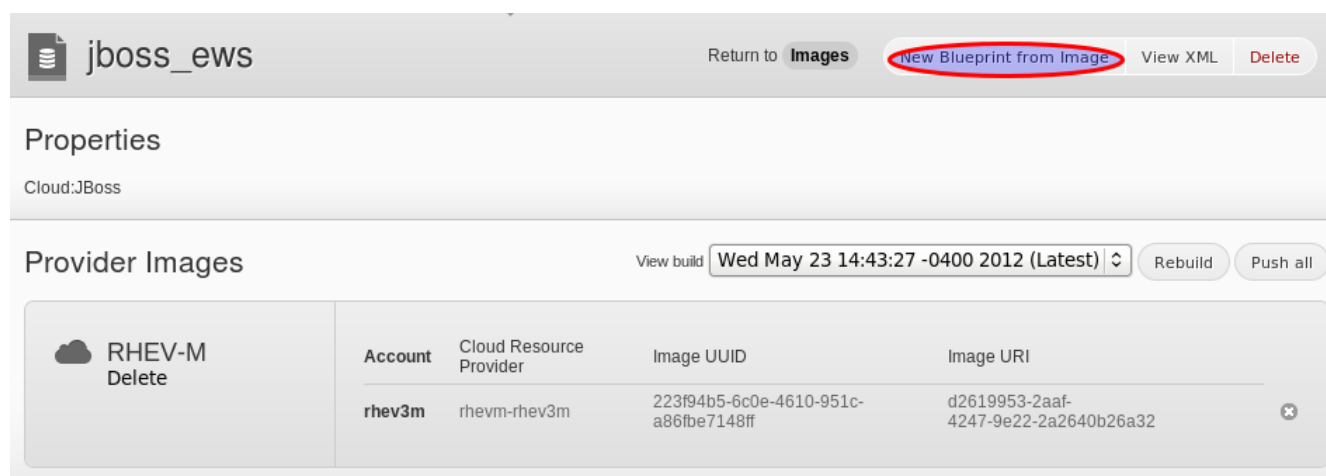


Figure 7.2.6: Create a New Blueprint



10. Choose the catalog and click **Save**

jboss_ews

Create Application Blueprint

Creating an application blueprint from the "jboss_ews" image will allow it to be built and launched. Simply choose a catalog in which to store the application blueprint, and define a default cloud resource profile.

Name

Default Cloud Resource Profile

Catalog **jboss**

Figure 7.2.7: Save the Blueprint



7.2.2 Add the XML service to the Image

When the XML service code is added to the Application Blueprint the Conductor launch process enables the “Configure Launch Time Parameters” screen at launch time. When the “Configure Launch Time Parameters” screen is enabled, the opportunity to modify parameters presents itself. Once the parameters are set, the Conductor launch process coordinates with the Configuration Server to ensure the variables are passed to the scripts being run on the AppForm.

1. Click **Administer**
2. Click **Content**
3. Click the JBoss Catalog
4. Select the **jboss_ews** Application Blueprint

The screenshot shows the JBoss Admin Console interface. At the top, there's a header with the 'JBoss' logo and buttons for 'New Catalog', 'Edit', and 'Delete'. Below this is a 'Properties' section showing 'Name: JBoss' and 'Cloud Resource Zone: JBossCRZ'. The main section is titled 'Application Blueprints'. It contains a table with two columns: 'Name' and 'Application Blueprint XML'. There are three rows in the table: 'configuration_server' and 'jboss_eap'. The 'jboss_eap' row is selected, and its name is circled in red. Above the table, there are buttons for 'Remove' and 'New Application Blueprint', and a 'Viewing' dropdown menu set to 'All Application Blueprints' with a count of '2'.

<input type="checkbox"/>	Name	Application Blueprint XML
<input type="checkbox"/>	configuration_server	configuration_server
<input checked="" type="checkbox"/>	jboss_eap	jboss_eap

Figure 7.2.8: Select the Application Blueprint



5. Click **Edit XML**

jboss_ews Delete Edit Edit XML Launch

Images

IMAGES VALID

Name	Images	Cloud Resource Profile	HDD	RAM	ARCH	UUIDs
jboss-ews	1	small-x86_64		512	x86_64	Show/Hide

Build Status

rhevm
rhevm-rhev3m | rhev3m

All Images are pushed and recent.

Figure 7.2.9: Edit the XML



6. Take the contents of the *jboss_ews_nodes_files.xml* file, which should be located on your System Engine server, and paste immediately below the **<image id>** element and before the **</assembly>** element.

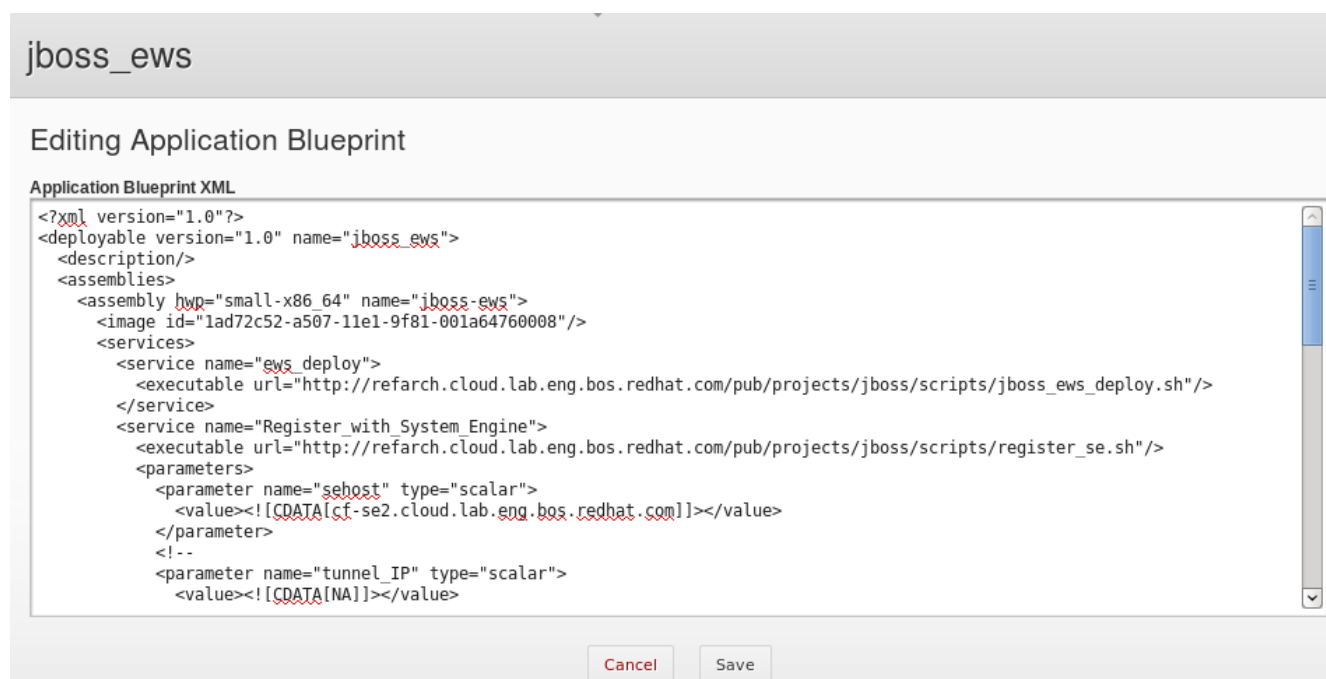


Figure 7.2.10: Add the XML

7. Click **Save**



7.2.3 Launch JBoss Enterprise Web Server

1. Click on the **Monitor** tab
2. Expand the JBoss cloud resource zone and click **New Application**

Your Cloud Resource Zones **2**

Default	JBossCRZ
APPLICATIONS 0 INSTANCES 0 QUOTA USED 0%	APPLICATIONS 6 INSTANCES 6 QUOTA USED 0%

Application	Total Usage Time	Instances	Status
jboss-config-server	N/A	1	Yellow
jboss-eap-dc4	18 hours	1	Red
jboss-eap-ews3	20 hours	1	Red
jboss-eap-hc1	8 hours	1	Red
jboss-eap-hc2	8 hours	1	Red
jboss-postgres	7 hours	1	Red

JBossCRZ Cloud Resource Zone details » **New Application**

Figure 7.2.11: Create a New Application

3. Provide an application name and select the **jboss_ews** application blueprint. This reference architecture uses **jboss_eap_ews** app as the name, click **Next**

New Application to Jboss-crz Cloud Resource Zone

Application Details

Application Name (uniquely identifies your application)

Name available

Application Blueprint


Cloud Resource Cluster

Next

Figure 7.2.12: Select Application Blueprint



4. Click the each tab on the left side and provide the name for the System Engine host on your network and click **Finalize**.

 JBossCRZ Cloud Resource Zone

Configure launch-time parameters for your application:

✓ Ews Deploy

✓ Register With System Engine

Register With System Engine

Sehost:

Org:

Environment:

Username:

Password:

Finalize

Figure 7.2.13: Provide the System Engine Host Information



5. Click **Launch**



The image shows a web form titled "JBossCRZ Cloud Resource Zone". The main heading is "Deploy 'jboss_ews' to JBossCRZ Cloud Resource Zone?". Below this is a confirmation message: "Are you sure you wish to deploy 'jboss_ews' to the JBossCRZ Cloud Resource Zone? Doing so will utilize 0% of your overall application quota." The form contains several fields: "Name" with the value "jboss_ews", "Application Blueprint XML" with the value "jboss_eap_ews", "Cloud Resource Cluster" with a dropdown menu set to "Auto-select", and "Image IDs" with the value "0b90ea56-a392-11e1-9f81-001a64760008". To the right of these fields is a large green arrow pointing right, superimposed on a cloud icon. At the bottom of the form are three buttons: "Cancel", "back", and "Launch". The "Launch" button is highlighted with a red circle.

Figure 7.2.14: Launch AppForm

7.2.4 Test JBoss Enterprise Web Server

1. Log into the **mod_cluster** interface http://dhcp-122:6666/mod_cluster-manager and confirm connectivity. This Reference Architecture uses DHCP to assign a hostname for the JBoss EWS.

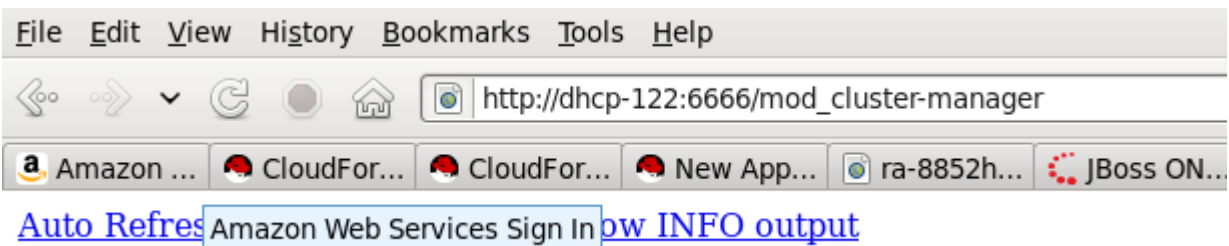


Figure 7.2.15: EWS

At this point there is no information to display in the `mod_cluster-manager` output. After the host controllers are added and placed in the *other-server-group*, the output contains the name of the Host Controllers and any applications that are deployed on them.



7.3 Deploy PostgreSQL Server

The PostgreSQL server provides the database that backs the kitchensink quickstart application.

7.3.1 Build and Push PostgreSQL Server

1. Log into the Cloud Engine interface
2. Click **Administer**
3. Click **Clouds**
4. Click on **New Image** for the JBoss Cloud
5. Provide a name and component outline. This reference architecture uses the name of **postgres_server**
 - Browse to the *postgres_server_template-dev-export.xml* component outline
6. Click **Continue**
7. Verify the XML and click **Save and Continue**
8. Click **Save Component Outline**
9. Click **Build** for the RHEV provider
10. Once the build is complete, click **Push**
11. Once the push is complete, click **New Blueprint from Image**
12. Choose the catalog and click **Save**



7.3.2 Add the XML configuration to the Image and Launch

1. Click **Administer**
2. Click **Content**
3. Click the **JBoss** Catalog
4. Select the **jboss_ews** Application Blueprint
5. Click **Edit XML**
6. Take the contents of the *jboss_postgres_files.xml* file and paste immediately below the **<image>** element. See the downloaded scripts and *services.xml* files for complete Application Blueprint example.
7. Click **Launch**
8. Click the each on the left side tab and provide the name for the System Engine host on your network. Also, change the password for the postgres database if needed. By default it is 123123.

Configure launch-time parameters for your application:

✓ Register With System Engine

✓ Configure Postgres

Configure Postgres

Postgrespassword:

Sehost:

Figure 7.3.1: PostgreSQL Password and SEHost

6. Click **Finalize**
7. Click **Launch**

7.3.3 Test PostgreSQL Server

Log into the PostgreSQL server and show the databases.

1. SSH into the PostgreSQL server and *su* to postgres user

```
# su - postgres
-bash-4.1$
```



2. List the databases and confirm the `ksink` database was created.

```
-bash-4.1$ psql -l
```

List of databases					
Name	Owner	Encoding	Collation	Ctype	Access privileges
ksink	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres
postgres=Ctc/postgres					:
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres
postgres=Ctc/postgres					:

(4 rows)

3. Access the PostgreSQL server and then logout.

```
$ psql
psql (8.4.9)
Type "help" for help.
```

4. Confirm the Configuration Server made the changes to the `pg_hba.conf` file as the root user.

```
# diff /var/lib/pgsql/data/pg_hba.conf /var/lib/pgsql/data/pg_hba.conf.orig
72c72
< host      all            all            0.0.0.0/0          trust
---
> host      all            all            127.0.0.1/32       ident
```



5. Confirm the PostgreSQL server is registering with the System Engine server. Login to the System Engine server, change to the appropriate organization and click the **Systems** tab.

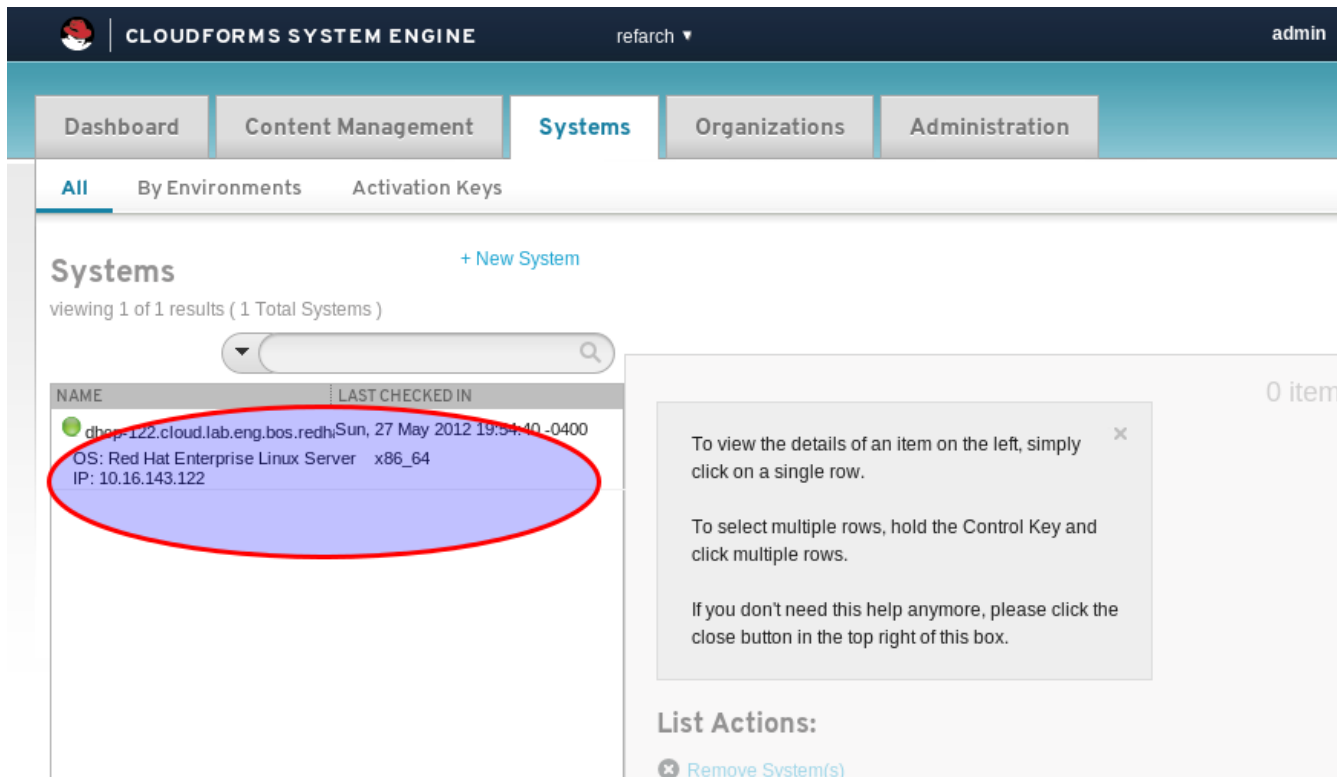


Figure 7.3.2: Registration with the System Engine Server

7.4 Deploy JBoss Enterprise Application Domain Controller

The JBoss EAP Domain Controller provides the coordination of events for the Host Controllers. The JBoss EAP Domain Controller and the Host Controllers both come from the same image. The configuration that differentiates the two is handled at launch time with Configuration Server. For the JBoss Instances, use the larger **jboss_large** cloud resource profile instead of the default small cloud resource profile. This provides more memory for the instance.

7.4.1 Create Application Blueprint for JBoss Enterprise Application Domain Controller

The EAP servers can use the same image as the EWS server. The difference is the configuration that is applied post boot. In this case, create a new Application Blueprint from the EWS image.

1. Log into Cloud Engine <https://cf-cloudforms2/conductor/>



2. Click **Administer**
3. Click **Clouds**
4. Click **Images**
5. Click **jboss_ews**
6. Click **New Blueprint from Image**
7. Provide a name of **jboss_eap**, provide a **Default Cloud Resource Profile** of *rhevm_large* and select the JBoss Default Cloud Resource Profile and the JBoss Catalog
8. Click **Save**

7.4.2 Add the XML configuration to the Image and Launch

1. Click **Administer**
2. Click **Content**
3. Click the **JBoss** Catalog
4. Select the *jboss_eap* Application Blueprint
5. Click **Edit XML**
6. Take the contents of the *jboss_eap_dc_hc_node_files.xml* file and paste immediately below the **<image>** element and before the **</assembly>** element. See the downloaded scripts and *services.xml* files for a complete Application Blueprint example. Click **Save**.
7. Click **Launch**
8. Click the each tab and provide the name for the System Engine host on your network. Also, change the password for the PostgreSQL database if needed.
9. Click on the **Config Jbossdc** tab and change **Is Domain Controller** to **true**
 - Provide the PostgreSQL server IP address in the **Postgre Sql Server** field.
10. Click on the **Jboss Jon Install** tab and put in the IP address of the existing JON server or leave blank if there is not one available.
11. Click **Finalize**
12. Change the name to **jboss_dc**
13. Click **Launch**. Due to the many scripts that have to be run and the many large files that need to be copied over, this could take a few minutes. Using the Red Hat Enterprise Virtualization management tool, open a console to the instance that is being launched. The launching agent should display “running aeolus-agent”. This could take a while.



7.4.3 Test Enterprise Application Domain Controller

1. Access the JBoss management console and confirm functionality. To access the console, open a browser and access `http://jboss_mgmt_address:9990/console`. Use the password that was passed using configuration server in the **Config Jbossdc** tab. By default it's 123123

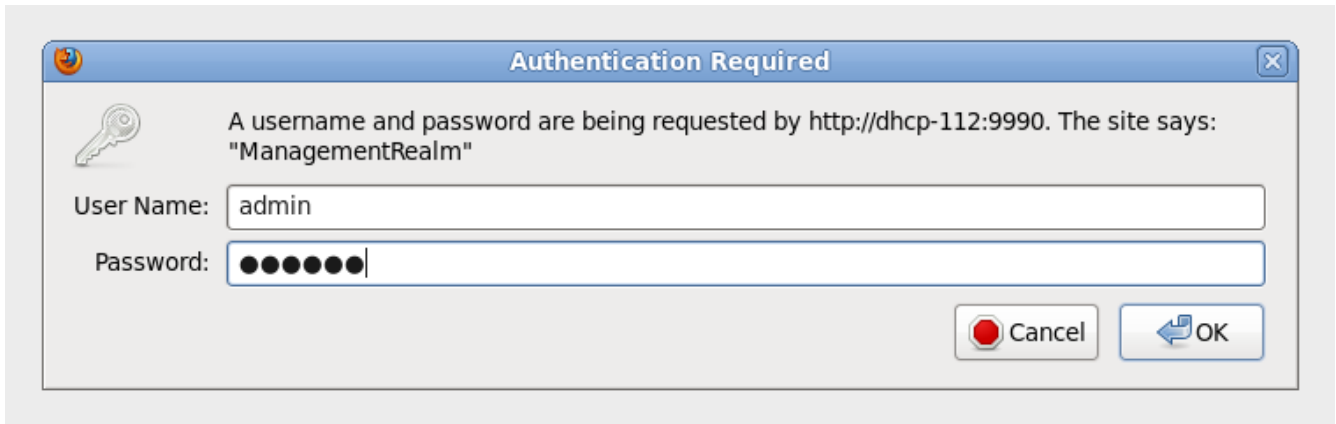


Figure 7.4.1: Log into Domain Controller

2. View the interface

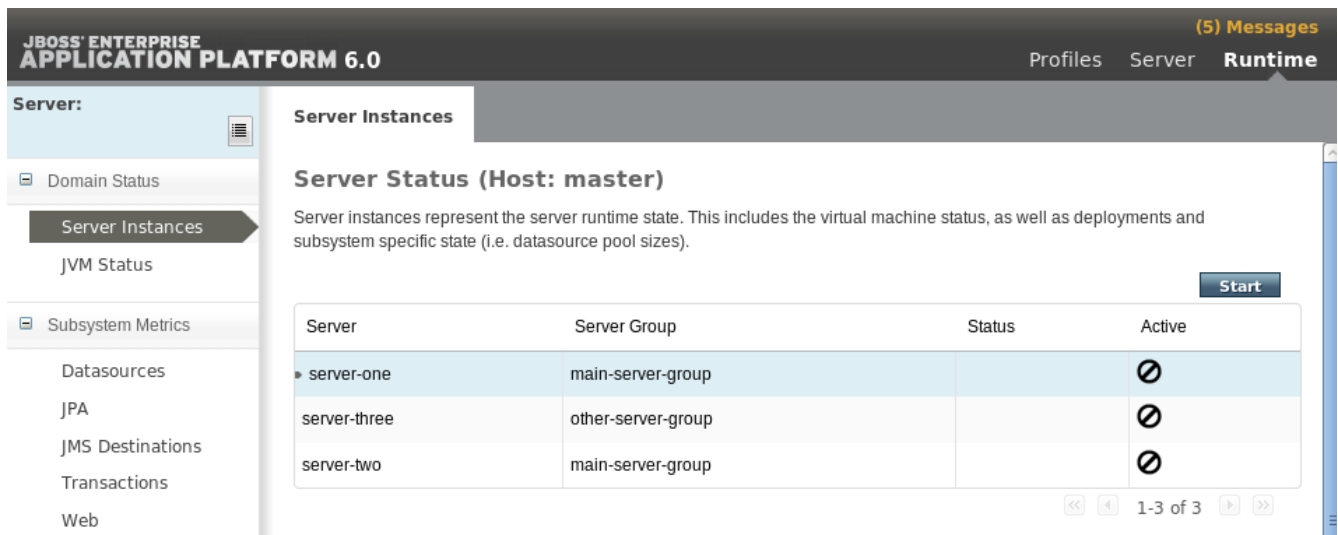


Figure 7.4.2: Confirm Functionality



- Log into the existing JON server and confirm auto-discovery worked. The JON server is typically accessed using the `http://jon.hostname.here.com:7080/coregui/` URL.

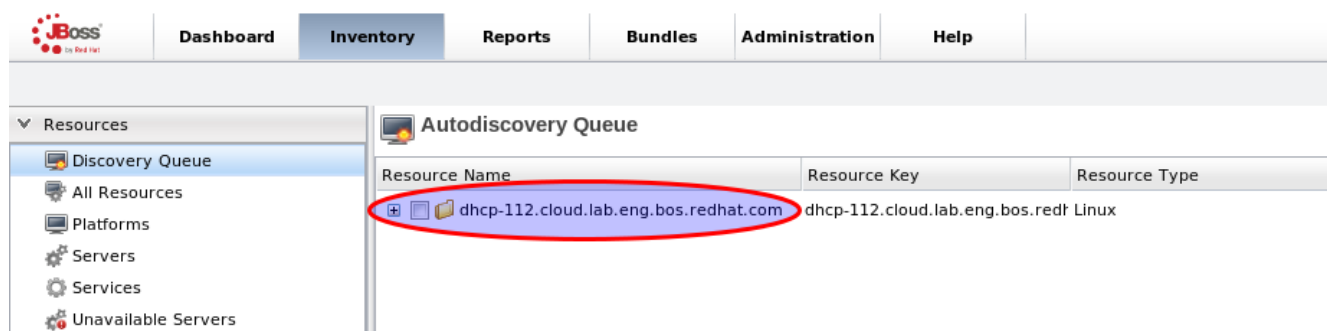


Figure 7.4.3: Auto-Discovery with JON

In this case the Domain Controller is named dhcp-112

- Confirm the `jboss-as-kitchensink.war` application is in the `pub` directory on the JBoss Domain Controller server either by logging in or via a browser.

```
# ls /var/www/html/pub/jboss-as-kitchensink.war
/var/www/html/pub/jboss-as-kitchensink.war
```

7.5 Deploy the JBoss Enterprise Application Host Controllers

The EAP host controller servers can use the same image and Application Blueprint as the EAP Domain Controller server. The difference is that instead of choosing Domain Controller during launch, Host Controller is picked.

7.5.1 Launch the JBoss Enterprise Application Host Controller

- Click **Monitor** tab
- Expand the **JBossCRZ** and select the **JBoss** catalog from the dropdown

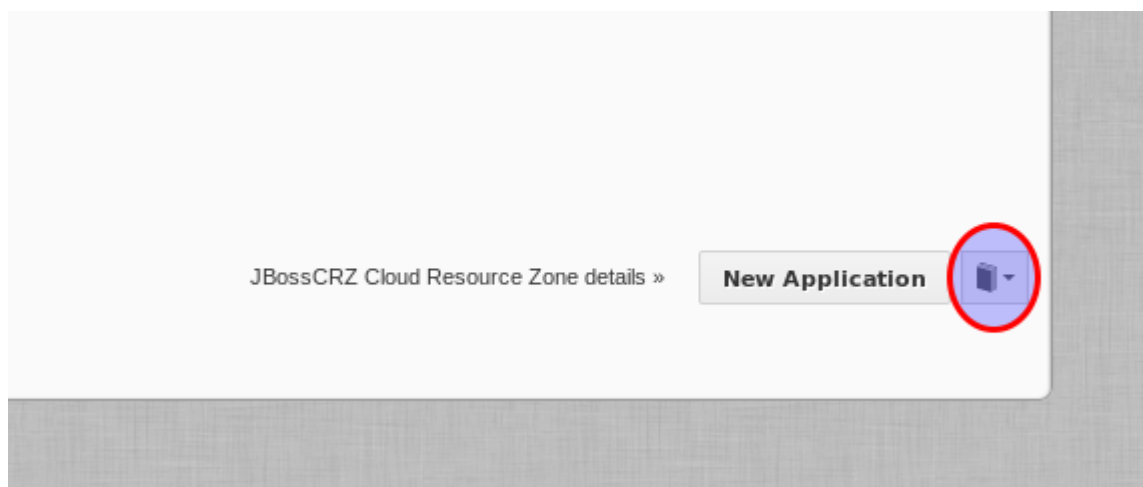


Figure 7.5.1: Launch New Application

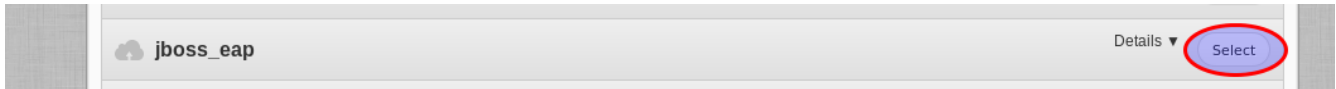


Figure 7.5.2: Choose Blueprint

3. Select the **jboss_eap** application blueprint
4. Click the **Config Jbosshc** tab and change *Is Host Controller* from **false** to **true**. Provide the Domain Controller IP as well as a Host Controller name and password then click **Finalize**. The password here is used to register with the JBoss Domain controller. The scripts that are run base64 encode the password and place it in the *domain.xml* file on the host controller. When the configuration scripts run on the host controller, the host controllers use the SSH keys created earlier to log into the Domain Controller and register themselves with it. This is what creates the scaling capability of the solution. If the password is not changed, it uses the default of 123123.

Configure launch-time parameters for your application:

Config Jbosshc

Is Host Controller:

Dc Ip:

Host Controller Name:

Hc Password:

Finalize

Figure 7.5.3: Confirm DC Functionality

5. Provide a name of **jboss_eap_hc1** and click **Launch**

Repeat this process and deploy a second host controller called hc2. More can be added if desired.

7.5.2 Confirm the Host Controllers

Log into both the JBoss EAP management console and the JON console to ensure the host controllers have registered with both services.

1. Log into the JBoss EAP management console at <http://dhcp-112:9990/console> and make sure both controllers are registered.



2. Make sure the **Runtime** environment is selected

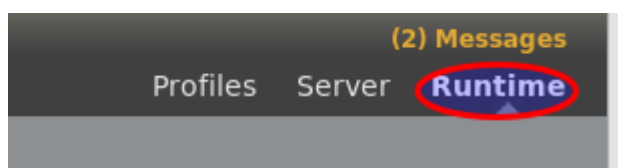


Figure 7.5.4: Runtime

3. Click the **Server** dropdown at the top of the left navigation pane and ensure both host controllers are registered

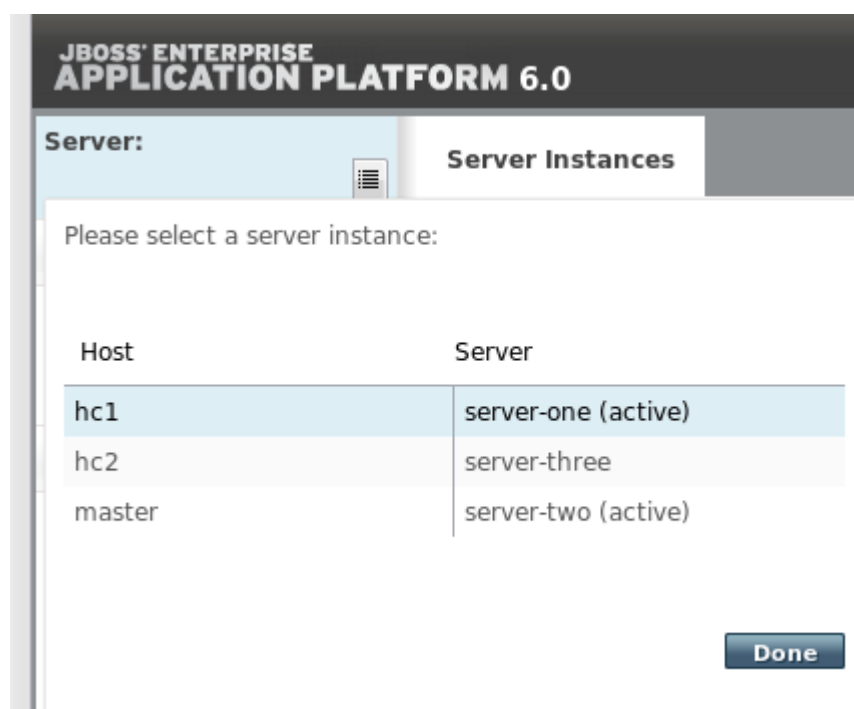


Figure 7.5.5: Instances



- The host controllers should have an active *other-server-group*.

Server Status (Host: hc1)

Server instances represent the server runtime state. This includes the virtual machine status, as well as deployments and subsystem specific state (i.e. datasource pool sizes).

Server Status (Host: hc1)			
Stop			
Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✓
server-two	main-server-group		✓

1-3 of 3

Figure 7.5.6: Active other-server-group

- Confirm the host controllers are also registered with the System Engine server and the JON server.
- Confirm that the host controllers are registered with the JBoss Enterprise Web Server. Access the EWS console by opening a browser and hitting http://ews.server.address.here/mod_cluster-manager. The content here has changed since the host controllers are now running with the full-ha profile as shown in **Figure 7.5.7: Enterprise Web Server Functionality**. Now the host controllers show up and any application running on them show up.

[Auto Refresh](#) [show DUMP output](#) [show INFO output](#)

Node hc1:server-three (<http://dhcp-114.cloud.lab.eng.bos.redhat.com:8330>):

[Enable Contexts](#) [Disable Contexts](#)

Balancer: mycluster,Domain: ,Flushpackets: Off,Flushwait: 10000,Ping: 10000000,Smax: 1,Ttl: 60000000,Elected: 0,Read: 0,Transferred: 0,Connected: 0,Load: 1

Virtual Host 1:

Contexts:

/, Status: ENABLED [Disable](#)

Aliases:

default-host
localhost
example.com

Node hc2:server-three (<http://dhcp-135.cloud.lab.eng.bos.redhat.com:8330>):

[Enable Contexts](#) [Disable Contexts](#)

Balancer: mycluster,Domain: ,Flushpackets: Off,Flushwait: 10000,Ping: 10000000,Smax: 1,Ttl: 60000000,Elected: 0,Read: 0,Transferred: 0,Connected: 0,Load: 1

Virtual Host 1:

Contexts:

Figure 7.5.7: Enterprise Web Server Functionality

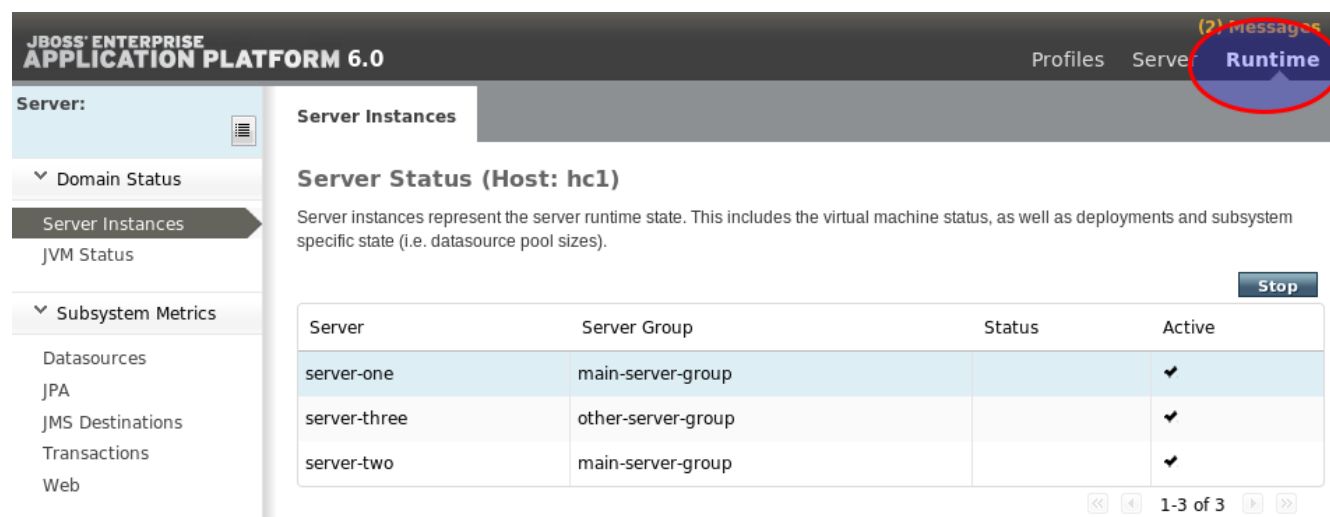


8 Deploy JBoss Application

As part of the post boot configuration scripts that the Audrey Agent ran on the JBoss Domain Controller, an application is built. This application is placed in the `/var/www/html/pub` directory on the JBoss Domain Controller. The only changes made to the `jboss-as-kitchensink.war` application are that the `persistence.xml` file now points to the PostgreSQL JNDI data source and the `kitchensink-quickstart-ds.xml` file is removed because the datasource is configured in the profile of the `domain.xml` file. For more information on how the datasource is configured along with the other customization's made to JBoss, please refer to the scripts provided with the Reference Architecture.

8.1 Deploy the Application

1. Copy the `/var/www/html/pub/jboss-as-kitchensink.war` file from the JBoss Domain Controller to a location that your browser has access to so it can be uploaded.
2. Log into the JBoss management console and click on the **Runtime** tab.



The screenshot shows the JBoss Enterprise Application Platform 6.0 management console. The top navigation bar includes 'Profiles', 'Server', and 'Runtime' (highlighted with a red circle). The left sidebar shows a tree view with 'Domain Status', 'Server Instances' (selected), 'JVM Status', 'Subsystem Metrics', 'Datasources', 'JPA', 'JMS Destinations', 'Transactions', and 'Web'. The main content area is titled 'Server Status (Host: hc1)' and contains a table of server instances. A 'Stop' button is visible in the top right of the table area.

Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✓
server-two	main-server-group		✓

1-3 of 3

Figure 8.1.1: Runtime



3. Click **Manage Deployments**

The screenshot shows the JBoss Enterprise Application Platform 6.0 console. The left sidebar contains a tree view with the following items: Domain Status, Server Instances (selected), JVM Status, Subsystem Metrics (with sub-items: Datasources, JPA, JMS Destinations, Transactions, Web), Runtime Operations (with sub-item: OSGi), Deployments (with sub-item: Manage Deployments circled in red), and Webservices. The main content area is titled 'Server Instances' and shows the 'Server Status (Host: hc1)'. It includes a description of server instances and a table of server groups. A 'Stop' button is visible in the top right. Below the table, there is a 'Status' section with an 'Availability' tab, and a 'Server Instance: server-one' section showing 'Server Configuration: server-one' and 'Running?: true'.

Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✓
server-two	main-server-group		✓

Figure 8.1.2: Manage Deployments

4. Click **Add Content**

The screenshot shows the 'Deployment Content' section of the JBoss Enterprise Application Platform 6.0 console. The 'Server Group Deployments' tab is selected. The 'Content Repository' section is empty, showing 'No items!'. An 'Add Content' button is circled in red in the top right corner of the Content Repository section. Below the button is a table with columns: Name, Runtime Name, Add to Groups, and Remove. The table is currently empty.

Name	Runtime Name	Add to Groups	Remove
No items!			

Figure 8.1.3: Add Content



5. Browse out to where the *jboss-as-kitchensink.war* file was placed and click **Next**.

Upload

Step 1/2: Deployment Selection

Please choose a file that you want to deploy.

/pub/tmp/jboss-as-kitchensink.war

Figure 8.1.4: Select Deployment

6. Click **Save**.

Upload

Step 2/2: Verify Deployment Names

Key: gLb7eOjUWh4kvJQavrm/jup+rIY

Name: jboss-as-kitchensink.war

Runtime Name: jboss-as-kitchensink.war

Figure 8.1.5: Save Deployment



7. Click the **Add to Groups** link

Deployment Content **Server Group Deployments**

Content Repository

Add Content

Name	Runtime Name	Add to Groups	Remove
jboss-as-kitchensink.war	jboss-as-kitchensink.war	Add to Groups	Remove

<< < 1-1 of 1 > >>

Figure 8.1.6: Add to other-server-group

8. Select the **other-server-group** and click **Save**.

Select server groups

Select server groups for jboss-as-kitchensink.war

Server Group	Profile	Add to Group
main-server-group	full	<input type="checkbox"/>
other-server-group	full-ha	<input checked="" type="checkbox"/>

<< < 1-2 of 2 > >>

☒ Enable jboss-as-kitchensink.war

Save Cancel

Figure 8.1.7: Save Configuration



9. Confirm that the application is registered with the **JBoss EWS** by logging into the EWS interface and refreshing the screen.

Node hc1:server-three

[Enable Contexts](#) [Disable Contexts](#)

Balancer: mycluster,Domain: ,Flushpackets: Off,Fl

Virtual Host 1:

Contexts:

```
/, Status: ENABLED Disable  
/iboss-as-kitchensink, Status: ENABLED Disable
```

Figure 8.1.8: Application is Registered with EWS



9 Confirm Functionality

Once the application is deployed, it is available two ways. The first is via using a URL that points directly to each JBoss EAP host controller server running the application. In this case there is more than one. The second way is by accessing a URL that points to the JBoss EWS server. This second option is the preferred method. The reason for this is because any sessions opened to individual servers may be lost if that EAP server loses connection. By opening a connection to the EWS server, all the connections are load balanced to the EAP nodes. This means that any connection to the EAP server that is lost can be load balanced to a different EAP server.

9.1 Access the Application Directly

1. Access the application directly on the host controller URLs. Open a browser and go to the the URL for the kitchensink quickstart application:

<http://dhcp-114.cloud.lab.eng.bos.redhat.com:8330/jboss-as-kitchensink/index.jsf>

<http://dhcp-135.cloud.lab.eng.bos.redhat.com:8330/jboss-as-kitchensink/index.jsf>

Figure 9.1.1: Kitchensink Application



At this point, there are no registered host controllers. The host controller will host the Jboss kitchensink application once it is deployed.

Welcome to JBoss!

You have successfully deployed a Java EE 6 web application.

Your application can run on:

**JBoss® ENTERPRISE
APPLICATION PLATFORM**

(Supported)

&



JBoss Application Server 7

(Community)

Member Registration

Enforces annotation-based constraints defined on the model class.

Name:

Email:

Phone #:

Register

• Registered!

Members

Id	Name	Email	Phone #	REST URL
1	JBoss User	jboss@jboss.org	5555555555	/rest/members/1
2	New JBoss	newjboss@jboss.org	5523555555	/rest/members/2

REST URL for all members: /rest/members

Figure 9.1.2: New Users

2. Access the other host controller URL and refresh the screen. The users should show up as they are accessing the same PostgreSQL database on the back-end.



9.2 Access the Application via JBoss EWS

1. Access the application using the URL that is provided by JBoss EWS.
<http://dhcp-122.cloud.lab.eng.bos.redhat.com/jboss-as-kitchensink/index.jsf>
2. Add another user and make sure it's accessible.

Welcome to JBoss!

You have successfully deployed a Java EE 6 web application.

Your application can run on:

**JBoss Enterprise
Application Platform**

(Supported)



JBoss Application Server 7

(Community)

Member Registration

Enforces annotation-based constraints defined on the model class.

Name:

Email:

Phone #:

Register

• Registered!

Members

Id	Name	Email	Phone #	REST URL
3	EWS User	ews@jboss.org	5555555555	/rest/members/3
1	JBoss User	jboss@jboss.org	5555555555	/rest/members/1
2	New JBoss	newjboss@jboss.org	5523555555	/rest/members/2

REST URL for all members: /rest/members

Figure 9.2.1: New User from other Host Controller



9.3 Confirm Load Balancing Functionality

Log-in to the JBoss Management interface and disable the **other-server-group** on **hc1**. If the browser is already connected to **hc1**, then no delay in service is detected. If a 404 error is received, refresh the browser a few times and it should be load balanced to the other node.

1. Log into the JBoss management interface and click on the **Runtime** tab.
2. Select the **other-server-group** and click **Stop** for the **hc1** server.

JBoss Management console screenshot showing the 'Runtime' tab. The 'Server Status (Host: hc1)' section displays a table of server instances. The 'other-server-group' is highlighted, and the 'Stop' button is circled in red.

Server	Server Group	Status	Active
server-one	main-server-group		✓
server-three	other-server-group		✓
server-two	main-server-group		✓

Figure 9.3.1: Failover

3. Access the EWS **mod_cluster_manager** URL and refresh the browser to ensure that **hc1** disappears from the available nodes.
4. Access the kitchen sink EWS URL and refresh the browser.
<http://dhcp-122.cloud.lab.eng.bos.redhat.com/jboss-as-kitchensink/index.jsf>
5. Repeat steps 2 and 3 for the **hc2** server and stop the application.
6. Access the kitchensink EWS URL and refresh the browser.
<http://dhcp-122.cloud.lab.eng.bos.redhat.com/jboss-as-kitchensink/index.js>

To add additional nodes in the future, just launch more JBoss host controller nodes. The nodes self register with the domain controller and add themselves to the *other-server-group*, the JBoss EWS, System Engine and the JON server if it exists. This is accomplished by utilizing the SSH keys that are in the *domain_controller_ssh_keys* directory on the System Engine server. As the host controllers come up, they SSH into the domain controller and use the JBoss CLI tools to automate the registration.



10 Summary

This Reference Architecture described the different components that are required to integrate CloudForms, JBoss Enterprise Application Platform, JBoss Enterprise Web Services, JBoss Operations Network and PostgreSQL. The use of configuration server and scripting was also covered to demonstrate how these components can be automated. By utilizing the described methodology to deploy JBoss host controllers in this manner, a truly scalable environment can be created. Because EAP 6 enables command line configuration with the `jboss-cli` command, automated host controller deployment is enabled.



Appendix A: Revision History

Revision 1.0
Initial Release

Monday June 26, 2012

Scott Collier



Appendix B: Troubleshooting

Configuration Server

The Configuration server is comprised of two components:

- Server
- Agent

The Configuration Server can be deployed to any of the three supported providers. This Reference Architecture deploys the Configuration Server to the Red Hat Enterprise Virtualization provider. The Configuration Server listens for requests from the agent that gets launched on a Appform that is being deployed.

To have a better idea of what's going on behind the scenes, take note of the log files on both server and agent boxes. To watch the server log files, issue the following command on the configuration server before launching an instance that will make a call to it:

```
# tail -f /var/log/aeolus-configserver/configserver.log
```

Once the image has contacted the configuration server, configured, and booted, the directory of importance is in **/var/audrey/tooling/*** on the instance as well as the log file **/var/log/audrey.log**. In the **/var/audrey/tooling** directory the following structure exists:



```
[root@dhcp-132 tooling]# tree /var/audrey/tooling/
/var/audrey/tooling/
├── auto_register_with_system_engine.log
├── jboss_domain_controller_configure.log
├── jboss_host_controller_configure.log
├── jboss_maven_jon_install.log
├── jboss_service_startup.log
├── log
├── user
│   ├── config_jbossc
│   │   ├── domain_datasource.xml
│   │   ├── domain_driver.xml
│   │   └── start
│   ├── config_jbosshc
│   │   └── start
│   ├── jboss_jon_install
│   │   ├── apache-maven-3.0.4-bin.tar.gz
│   │   ├── jboss-as-quickstarts-7.1.1.CR1-dist.zip
│   │   ├── jboss-eap-6.0.0.Beta1-maven-repository.zip
│   │   ├── jboss-eap-6.0.0.Beta1.zip
│   │   ├── jon-plugin-pack-tech-preview-3.0.1.GA.zip
│   │   └── start
│   ├── Register_with_System_Engine
│   │   └── start
│   └── Start_JBoss
│       └── start
```

6 directories, 18 files

Figure 10.1: Audrey Agent Directory Structure

The log files are at the root of `/var/audrey/tooling/`. There are custom logs that the configuration scripts place, like `jboss_service_startup.log`. There is also a default log called **log** that records all activities. In addition to the logs, all services actions are recorded as scripts and placed in the `/var/audrey/tooling/user/service_name/` directory. In addition to any scripts that are run being recorded here, any files that are transferred are recorded in the same location. For example, notice how in the `/var/audrey/tooling/user/jboss_jon_install` directory, the `jboss*.zip` files are placed as shown in **Figure 10.1: Audrey Agent Directory Structure**. These are great places to start when diagnosing problems with the configuration server and any scripts that it runs on the images.



JBoss Enterprise Web Server

The JBoss EWS runs out of the `/opt/jboss-ews-1.0/httpd/sbin` directory. The startup script is called `apache_ews` and the logs are stored in the `/opt/jboss-ews-1.0/httpd/logs` directory. To troubleshoot the EWS, it is helpful to issue a **tailf -f** against those log files.

```
# tail -f /opt/jboss-ews-1.0/httpd/logs/*
```



Appendix C: Contributors

I would like to thank the following individuals for their time and patience as we collaborated on this process. This document would not have been possible without their many contributions.

Contributor	Title	Contribution
Vijay Trehan	Director of Solutions Architectures	Reviews
Joe Fernandes	Product Marketing Manager	Content, Reveiws
Vinny Valdez	Principal Consultant	Content, Reviews
Gordon Haff	Senior Product Marketing Manager	Content, Diagrams
Brett Thurber, RHCA	Senior Software Engineer	Reviews
Babak Mozaffari	Managing Principal Architect	Application Content
Shane Johnson	Product Manager	Reviews
Steven Reichard	Senior Principal Software Engineer	Reviews
Jacob Liberman	Principal Software Engineer	Reviews

- 1 <http://www.redhat.com/ressourcelibrary/reference-architectures/deploying-a-jboss-eap-6-managed-environment-cloudforms-v1>
- 2 <http://www.redhat.com/ressourcelibrary/reference-architectures/deploying-a-jboss-eap-6-managed-environment-cloudforms-v1>
- 3 <http://www.redhat.com/ressourcelibrary/reference-architectures/red-hat-cloudforms-v1-infrastructure-and-application-deployment-fundamentals>
- 4 <http://www.redhat.com/ressourcelibrary/reference-architectures/red-hat-cloudforms-v1-infrastructure-and-application-deployment-fundamentals>
- 5 <http://www.redhat.com/ressourcelibrary/reference-architectures/red-hat-cloudforms-v1-infrastructure-and-application-deployment-fundamentals>
- 6 <https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=7063>
- 7 <https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=12343>
- 8 <https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=11333>
- 9 <https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=11313>
- 10 <https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=11343>
- 11 <http://www.apache.org/dyn/closer.cgi/maven/binaries/apache-maven-3.0.4-bin.tar.gz>
- 12 <http://jdbc.postgresql.org/download/postgresql-9.1-902.jdbc3.jar>
- 13 https://access.redhat.com/sites/default/files/csrepo_v1_0.tgz
- 14 <http://www.redhat.com/ressourcelibrary/reference-architectures/deploying-a-jboss-eap-6-managed-environment-cloudforms-v1>
- 15 <http://www.redhat.com/ressourcelibrary/reference-architectures/deploying-a-jboss-eap-6-managed-environment-cloudforms-v1>
- 16 https://access.redhat.com/knowledge/docs/JBoss_Enterprise_Application_Platform/