

Course Guide

Design and Deploy Full-Stack Cloud Environments with IBM UrbanCode Deploy - v6.2.1.1

Course code UCD62 ERC 1.0



August 2017

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, and the Adobe logo, are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

© Copyright International Business Machines Corporation 2017.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	P-1
Contents	P-3
Course overview	P-7
Document conventions	P-8
Exercises	P-9
Additional training resources	P-10
IBM product help.....	P-11
Unit 1 Overview	1-1
Unit objectives	1-2
What is Blueprint?	1-3
Blueprint Designer multiple cloud supportability	1-4
Separate responsibilities in DevOps organizations	1-5
Creating and maintaining blueprints	1-6
Application components accessibility	1-7
Blueprints in the main interface of IBM UrbanCode Deploy.....	1-8
Components deployed during environment creation	1-9
Run existing processes to update cloud environments	1-10
Blueprints in Git repositories	1-11
Integrations to provision environments and deploy software.....	1-12
How does OpenStack help?	1-13
OpenStack addresses core components	1-14
OpenStack services on the Horizon dashboard	1-15
OpenStack components and their roles.....	1-16
Components provided by IBM UrbanCode Deploy	1-17
The Heat stack is provisioned to the cloud	1-18
The OpenStack architecture with Heat	1-19
Unit summary.....	1-20
Unit 2 Settings and Configuration	2-1
Unit objectives	2-2
Blueprint design server	2-3
Blueprint design server configuration	2-4
Authentication realm	2-5
Importing user accounts	2-6
Different roles	2-7
Authentication realm	2-8
Users and roles.....	2-9
User access	2-10
Demo user access	2-11

Server connections	2-12
IBM UrbanCode Deploy components	2-14
Overview: Configure security for the blueprint design server	2-15
Unit summary.....	2-16
About the exercises	2-17
Exercises: Configuring the blueprint design server integrations	2-20
Exercise 1: Import users from the OpenStack Keystone service.....	2-22
Exercise 2: Grant privileges to teams and users	2-27
Exercise 3: Analyze and extend the OpenStack environment.....	2-34
Exercise 4: Configure the connection to the IBM UrbanCode Deploy server	2-39
Unit 3 Blueprints and Cloud Infrastructure.....	3-1
Unit objectives	3-2
Blueprint Designer user interface	3-3
Diagram and Source editor in Blueprint Designer	3-4
A graphical depiction of the modeled template.....	3-5
Source code editor.....	3-6
Preserving object selection and state	3-7
The Source editor includes code validation	3-8
Blueprints.....	3-9
The Images drawer.....	3-10
The Networks drawer.....	3-11
The Storage drawer	3-12
Access network and resources.....	3-13
Provisioning of environments.....	3-14
Creation of the virtual environment.....	3-15
Environments.....	3-16
OpenStack shows the deployed instance.....	3-17
IBM UrbanCode Deploy shows no activity.....	3-18
Unit summary.....	3-19
Exercises: Modeling a heat stack and provisioning a cloud infrastructure.....	3-20
Exercise 1: Create a simple blueprint	3-21
Exercise 2: Provision a blueprint to a cloud environment	3-31
Exercise 3: Observe the provisioned environment in OpenStack.....	3-36
Exercise 4: Verify no activity in IBM UrbanCode Deploy	3-39
Unit 4 Blueprints and UCD Apps	4-1
Unit objectives	4-2
Deployment process	4-3
Application models include components, processes, and environments.....	4-4
What is a Component?	4-5
What are Processes?	4-6

Environment.....	4-7
Provisioning infrastructure is the current bottleneck in the delivery pipeline.....	4-8
The Components drawer	4-9
Components adding to a blueprint.....	4-10
Deploying sequence for components	4-11
Component process in IBM UrbanCode Deploy.....	4-12
Process steps and their environment properties	4-13
Setting properties in the Blueprint Designer	4-14
Provisioning an environment: Behind the scenes	4-15
Agents on the IBM UrbanCode Deploy server.....	4-16
Provisioning of blueprint from the Blueprint Designer.....	4-17
Deployment of the components begins	4-18
Components in IBM UrbanCode Deploy on running of deployment process.....	4-19
Component process.....	4-20
Components deployed on the environment	4-21
Components are deployed to environments by agents	4-22
The blueprint displays as an application process on the IBM UrbanCode Deploy server.....	4-23
Components deployed shown on the Environment tab	4-24
Successfully provisioned environment.....	4-25
Deleting an environment from either UrbanCode Deploy or Blueprint Designer	4-26
Environments are deleted across all tools	4-27
Unit summary.....	4-28
Exercises: Deploying applications to a cloud environment.....	4-29
Exercise 1: Add IBM UrbanCode Deploy components to the blueprint	4-30
Exercise 2: Explore the application in IBM UrbanCode Deploy	4-37
Exercise 3: Update the environment property in the Blueprint Designer	4-43
Exercise 4: Provision the environment and observing the deployment	4-46
Exercise 5: Validate the application	4-55
Exercise 6: Delete the cloud environment	4-58
Unit 5 Blueprints and UCD Provision.....	5-1
Unit objectives	5-2
Provisioning configuration files on different cloud systems.....	5-3
Importing properties from existing blueprints	5-4
Configuration files list parameters to provision blueprint	5-5
Separate functional areas for the web and database servers.....	5-6
Environments created from blueprints	5-7
Blueprints created in the Blueprint Designer are available in IBM UrbanCode Deploy.....	5-8
Predefined properties and values to use	5-9
Creating an environment	5-10

Instances of servers on OpenStack	5-11
Agent processes are started separately for each resource	5-12
Updating environment in Blueprint Designer	5-13
Blueprints can update a running environment	5-14
Instances name are updated in the Blueprint Designer.....	5-15
Unit summary.....	5-16
Exercises: Modeling a two-server blueprint and provisioning form the IBM UrbanCode Deploy	5-17
Exercise 1: Create a blueprint with two servers.....	5-18
Exercise 2: Assign the IP address to the database server	5-24
Exercise 3: Save parameters to a configuration file	5-29
Exercise 4: Provision the two-server blueprint.....	5-33
Exercise 5: Provision a blueprint from IBM UrbanCode Deploy	5-37
Exercise 6: Update a running environment.....	5-46
Unit 6 Repositories	6-1
Unit objectives	6-2
Storage of files in repositories	6-3
Git functions on the Repositories page	6-5
Minimize conflicts.....	6-6
Forks.....	6-7
Cloned repository.....	6-8
File history	6-9
Transferring changes from local repository to remote repository.....	6-10
Committing changes	6-11
Commit history	6-12
Recent activity	6-13
Editing the source code	6-14
Making changes available to local repositories	6-15
Fetching	6-16
Syncing	6-17
Unit summary.....	6-18
Exercises: Using repositories to store and manage blueprints	6-19
Exercise 1: Copy a project to your local workspace	6-20
Exercise 2: Modify the configuration file and provisioning the blueprint	6-26
Exercise 3: Make file changes and updating the remote repository	6-32

Course overview

Preface overview

IBM UrbanCode Deploy is a tool for standardizing and simplifying the process of deploying software components to each environment in your development cycle. When you use blueprints for OpenStack-based clouds, you can use a full-stack approach to simultaneously model the application and infrastructure layers of your deployment.

In this course, you learn how to administer the cloud through both the Blueprint Designer and the Horizon user interface. Hands-on labs use IBM UrbanCode Deploy in a cloud environment and cover integrations with an OpenStack back-end and IBM UrbanCode Deploy, modeling the cloud infrastructure and application layers, provisioning environments from blueprints, creating and using configuration files, updating a running environment, and using Git repositories to store and manage blueprints.

Intended audience

This is a basic course for new users of the IBM UrbanCode Deploy Blueprint Designer, such as administrators, performance testers, development teams, and operations leads.

Topics covered

Topics covered in this course include:

- Overview
- Settings and Configuration
- Blueprints and Cloud Infrastructure
- Blueprints and UCD Apps
- Blueprints and UCD Provision
- Repositories

Course prerequisites

Participants should have:

- Cloud computing (private and public)
- OpenStack, in particular the Heat project
- IBM UrbanCode Deploy applications and components

Document conventions

Conventions used in this guide follow Microsoft Windows application standards, where applicable. As well, the following conventions are observed:

- **Bold:** Bold style is used in exercise steps to indicate a user interface element that is actively selected or text that must be typed by the participant.
- *Italic:* Used to reference book titles.
- **CAPITALIZATION:** All file names, table names, column names, and folder names appear in this guide exactly as they appear in the application.
To keep capitalization consistent with this guide, type text exactly as shown.

Exercises

Exercise format

Exercises are designed to allow you to work according to your own pace. The exercises are structured as follows:

The task and results section

This section provides a task based set of instructions that presents the question as a series of numbered tasks to be accomplished. The information in the tasks expands on the business case, providing more details on how to accomplish a task. Screen captures are also provided at the end of some tasks and at the end of the exercise to show the expected results.

Additional training resources

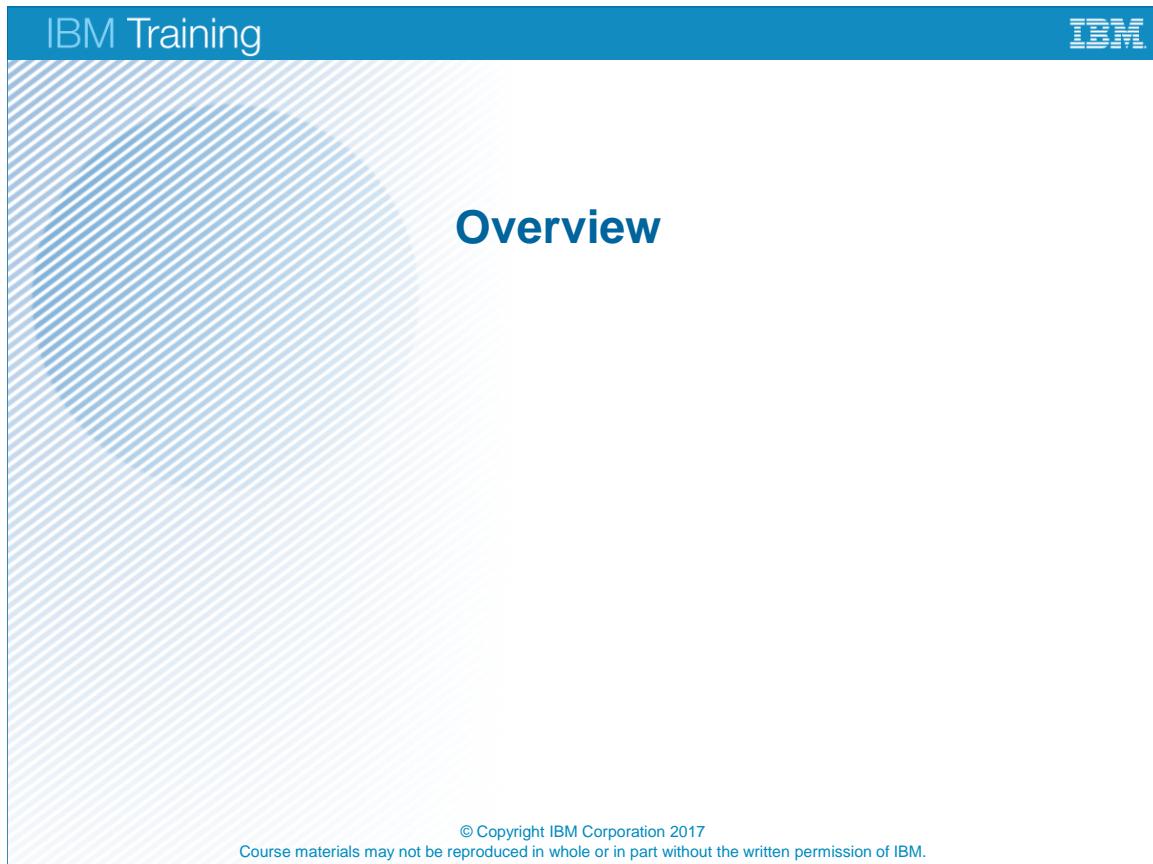
Visit the IBM Skills Gateway (www.ibm.com/training/) for details on:

- Instructor-led training in a classroom or online
- Self-paced training that fits your needs and schedule
- Comprehensive curricula, learning journeys, and training paths that help you identify the courses that are right for you

IBM product help

Help type	When to use	Location
Task-oriented	You are working in the product and you need specific task-oriented help.	https://developer.ibm.com/answers/smartspace/urbancode/
IBM on the Web	You want to access any of the following website: <ul style="list-style-type: none">• IBM Cloud Education Resource Guide• Online support• IBM Website	<ul style="list-style-type: none">• www.ibm.biz/CloudEduResources• https://www.ibm.com/support/home/• http://www.ibm.com

Unit 1 Overview



The slide features a blue header bar with 'IBM Training' on the left and the IBM logo on the right. The main content area has a light blue diagonal striped background. The word 'Overview' is centered in large, bold, dark blue capital letters. At the bottom, there is a copyright notice: '© Copyright IBM Corporation 2017' and 'Course materials may not be reproduced in whole or in part without the written permission of IBM.'

IBM Training

IBM

Overview

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Describe the capabilities and architecture of OpenStack
- Describe the scenario of a Heat stack in software deployment
- Describe how full-stack patterns work across cloud offerings
- Explore the capabilities of the blueprint designer

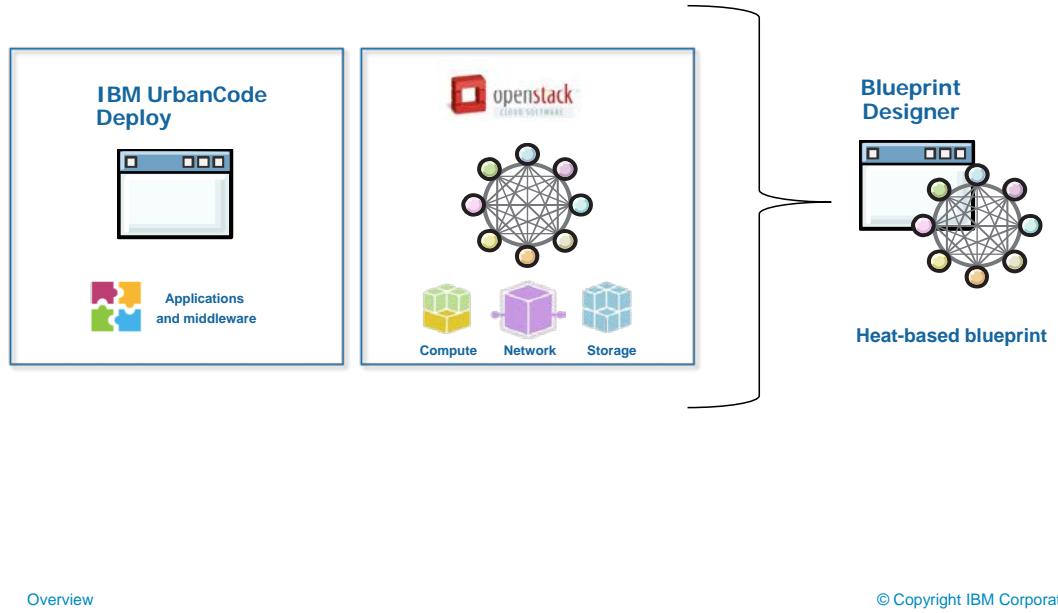
[Overview](#)

© Copyright IBM Corporation 2017

Unit objectives

The IBM UrbanCode Deploy Blueprint Designer is a full-stack environment management and deployment solution that you can use to design, deploy, and update full-stack environments for multiple clouds.

What is Blueprint?



Overview

© Copyright IBM Corporation 2017

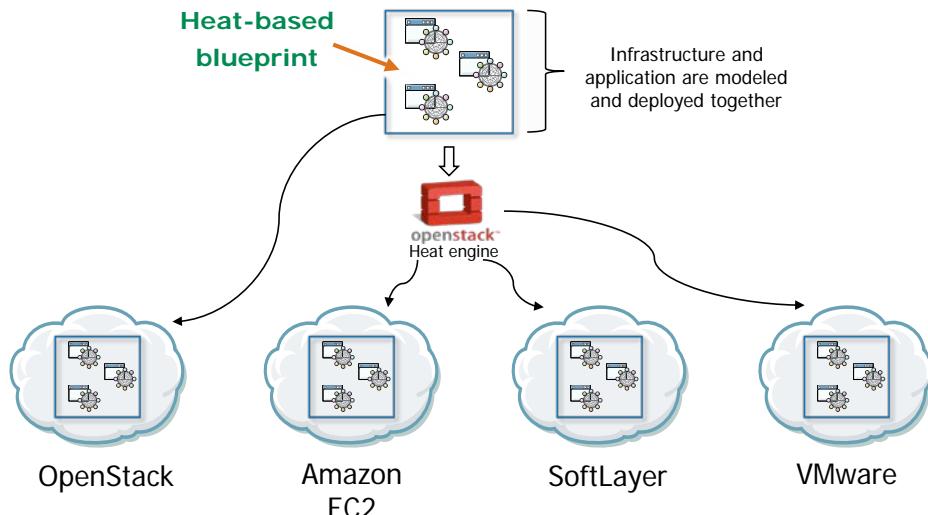
What is Blueprint?

Blueprints model the environment infrastructure and applications in a single document that can be managed and versioned as code.

You provision environments on clouds through HEAT-based blueprints that you create in the Blueprint Designer. Heat-based blueprints use OpenStack Heat Orchestration Templates (HOT) to model the infrastructure of a cloud environment, including its virtual images, network, and storage. A heat-based blueprint can also include software components to be deployed in the environment.

You can create or update a cloud environment based on a blueprint. The blueprint design server runs this provisioning or updating by using a Heat engine. If the blueprint includes software components, the provisioning or updating process also deploys those components through the automation processes that are specified in the blueprint. For example, blueprints can use component processes on the IBM UrbanCode Deploy server. Blueprints can also specify other automation details, such as the deployment order for components.

Blueprint Designer multiple cloud supportability



Overview

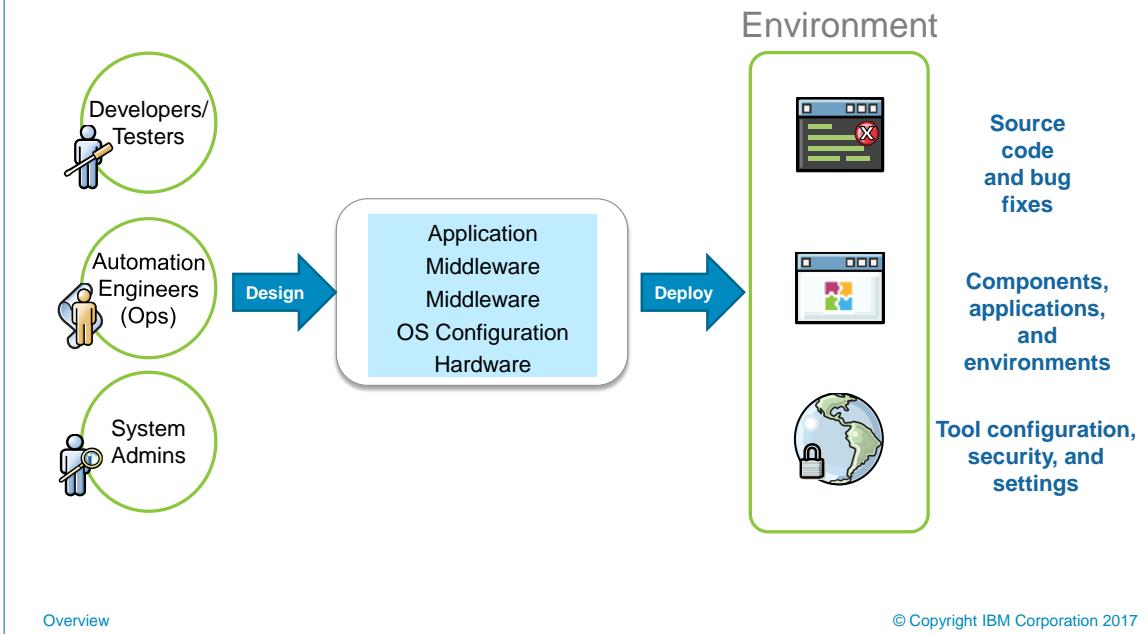
© Copyright IBM Corporation 2017

Blueprint Designer multiple cloud supportability

The Blueprint Designer uses an OpenStack Heat engine to translate blueprints (using a Heat Orchestration Template, or HOT) and make them appropriate for your target cloud. Parameters specific to a cloud provider are defined and stored in a configuration file. A configuration file complements the cloud-agnostic blueprint at provisioning time by providing properties that are unique to the target environment. You can set up one blueprint and deploy it to multiple cloud types, such as OpenStack, IBM SoftLayer, VMware vCenter, and Amazon. This course focuses on OpenStack. The resources that are described in the blueprint capture everything that is required to set up a network environment. Instead of coordinating your infrastructure model separately from your application model, you model and deploy them both together.

Note: For an OpenStack back end, an additional Heat engine does not have to be installed, as Heat is already an OpenStack component.

Separate responsibilities in DevOps organizations



Separate responsibilities in DevOps organizations

IBM UrbanCode Deploy allows for the separation of responsibilities between development and operations. Typically, application developers create and deliver code, deploy code to the development environment for unit testing, and fix defects.

Automation engineers author and maintain components and application processes, define the deployment environments, create the infrastructure elements (compute, network, or storage), and promote versions to their next environment (for example, SIT, QA, PERF, PROD).

System administrators set up and configure the tool for use, ensure that the tools are up to date, and set up and maintain the security of the system.

As a designer, such as an IT specialist or an IT integrator, you can create the complete application stack by designing and adapting the template. Then a consumer, such as a developer, can go into IBM UrbanCode Deploy and create a complete environment with just a few clicks.

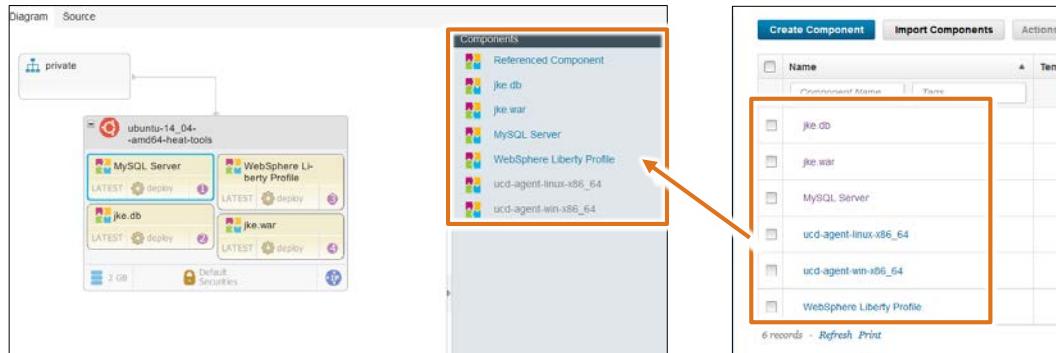
The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, the **Diagram editor** is displayed, featuring a canvas where blueprints are composed using drag-and-drop operations. A palette on the right contains various components like MySQL Server, Jenkins, and MySQL Database. On the far left, there are panes for **Properties** and **Parameters**. An orange arrow points from the **Properties and Parameters panes** to the **Diagram editor**, indicating how these settings are used in the visual design. On the right, the **Source editor** shows the blueprint's configuration in **YAML-based syntax**. Another orange arrow points from the **Diagram editor** to the **Source editor**, illustrating the connection between the visual design and the underlying YAML code.

Creating and maintaining blueprints

The Blueprint Designer has a Diagram editor, where you can design blueprints through drag-and-drop operations. The blueprint is a document complying to the OpenStack Heat Orchestration Template (HOT) format. In addition to the Diagram editor, you can use the Source editor to edit the blueprint document in its YAML-based syntax. Both editors facilitate the setting of input and output parameters through a Properties pane and include facilities to validate and test the created blueprints.

After the blueprint design server is connected to a cloud, you have access to the images, networks, storage, and policies that are available in that cloud. You can drag those elements from the palette into the editor canvas to compose the blueprint.

Application components accessibility



Components in IBM
UrbanCode Deploy

Overview

© Copyright IBM Corporation 2017

Application components accessibility

Application components from IBM UrbanCode Deploy are also accessible in the Blueprint Designer palette after the blueprint design server is connected to the IBM UrbanCode Deploy server. When an environment is created from a blueprint, the Heat engine orchestrates the provisioning of the infrastructure resources that instantiate the environment in the cloud. If the blueprint contains components, the Heat engine installs the IBM UrbanCode Deploy agent in the new environment. As a result, the component processes specified in the blueprint are invoked and carry out the deployment of those components.

The screenshot shows the IBM UrbanCode Deploy application interface. On the left, there's a navigation bar with tabs like Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below this, the main area shows an application named 'JKE'. It has sections for Created By (admin), Created On (7/29/2016, 3:13 PM), Description (Sample JKE Banking applic...), and Environments. Two environments are listed: 'JKE-UAT' (red background) and 'JKE-DEV' (green background). A modal window titled 'Create Environment' is open, prompting for a 'Name' (JKE-UA), 'Description', 'Blueprint' (dropdown menu showing 'JKE', 'JKE-2-DEMO-TESTS', and 'JKE2'), 'Teams' (dropdown menu showing 'JKE'), 'Require Approvals' (checkbox), 'Exempt Processes' (checkbox), and 'Lock Snapshots' (checkbox). A color palette at the bottom indicates the color 'Bright Cerulean (Blue)'. At the bottom of the modal are 'Save' and 'Cancel' buttons. A callout box with an orange arrow points to the 'JKE-2-DEMO-TESTS' option in the Blueprint dropdown, with the text: 'Blueprints that are created in the Blueprint Designer have cloud icons'.

Blueprints in the main interface of IBM UrbanCode Deploy

The integration between the blueprint design server and the IBM UrbanCode Deploy server not only allows application components to become part of a blueprint, it also allows the blueprint to be accessible in the IBM UrbanCode Deploy user interface for creating environments. Environments created from blueprints are listed in the Environments view in the IBM UrbanCode Deploy user interface, along with regular environments. You can recognize environments generated from blueprints by the cloud icon before their name in the Applications or Environments view.

The screenshot shows the IBM UrbanCode Deploy interface. At the top, there's a navigation bar with tabs like Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below the navigation bar, it says "Application: JKE". Underneath that, there's a sub-navigation bar with tabs for Environments, History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. A "Create Environment" button is visible. The main area displays two environments: "JKE-UAT" and "JKE-DEV". The "JKE-UAT" environment is expanded, showing a table with four rows of components. Each row includes columns for Component, Version, Date Created, Snapshot, Properties, Status, Compliancy, and Actions. All components are listed as Active and Compliant (1/1). A yellow callout box with the text "Deployed components" has an orange bracket pointing to the "JKE-UAT" table. The bottom right corner of the screenshot contains the copyright notice "© Copyright IBM Corporation 2017".

Components deployed during environment creation

Components are automatically deployed during environment creation. The creation of an environment from a blueprint instantiates the server and other infrastructure in the cloud. Then it deploys the components included in the blueprint by executing the corresponding component deployment processes that are included in the blueprint. As a result, the creation of the infrastructure and the deployment of components both take place upon creation of the environment.

The screenshot shows the IBM UrbanCode Deploy application interface. At the top, there's a navigation bar with tabs for Dashboard, Components, Applications (which is selected), Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below the navigation bar, the page title is "Run existing processes to update cloud environments". The main content area shows an application named "JKE". It includes details like "Created By: admin" and "Created On: 7/29/2016, 3:13 PM". A "Description" field states "Sample JKE Banking application". Below this, there's a toolbar with buttons for Environments, History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. A prominent "Create Environment" button is visible. The main pane lists environments: "JKE-UAT" (status bar is red, indicating an issue) and "JKE-DEV" (status bar is green). Each environment row has columns for actions (a play icon with a circle), a thumbnail, three dots, and the environment name. To the right of the environments, there's a "Compliance" section showing values like "4 / 4" for JKE-UAT and "0 / 0" for JKE-DEV. At the bottom of the main pane, there are search fields for "Search by Name" and "Search by Blueprint", along with "Expand All" and "Collapse All" buttons. The footer of the screenshot includes "Overview" and "© Copyright IBM Corporation 2017".

Run existing processes to update cloud environments

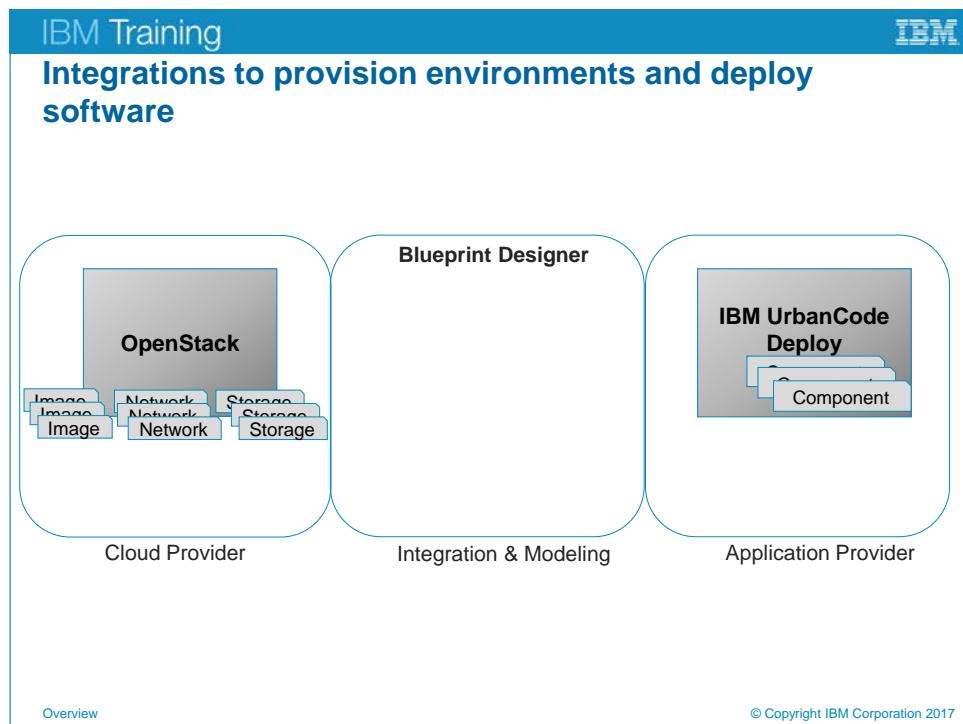
Cloud environments created from blueprints perform like standard environments. Cloud environments have the added advantage that their creation might include component deployment if the blueprints from which they are created include software components. You can update an environment by updating the blueprint in the Blueprint Designer, such as when adding a component, and then applying the updated blueprint to the environment.

An important difference between standard environments and environments generated from blueprints is that these blueprints also act as application processes. If you change a blueprint in the Blueprint Designer, you can apply the change to the environment from which the blueprint was created by invoking a *Run Process* and selecting the blueprint from the process drop-down list. By treating blueprints as application processes in IBM UrbanCode Deploy, you can evolve the model captured in a blueprint and update the environments with the changed blueprints by running the blueprint as an application process on that environment.

The screenshot shows the UrbanCode Deploy Blueprint Designer application. At the top, there's a header bar with the IBM logo and the title "Blueprints in Git repositories". Below the header is a navigation bar with tabs for "Repository: demo" and "Reference: master => origin/master". The main area is divided into two panes. The left pane, titled "Working Directory Changes", shows a summary: "Active Branch (master)", "Working Directory Changes: 5 files changed, 0 files ready to commit", and sections for "Outgoing (0)" (No Changes), "Incoming (0)" (No Changes), and "History" (Initial commit, root on 8/15/2016, 1:29:36 PM). The right pane, titled "Working Directory Changes", contains a "Commit" message input field ("Enter the commit message") and a list of selected files: "configurations/JKE-OPENSTACK-CONFIG.yml", "configurations/JKE-OPENSTACK-CONFIG1.yml", "JKE-2-SERVERS/JKE-2-SERVERS.yml", ".JKE/JKE.yml", and "JKE2/JKE2.yml". There are checkboxes for "Amend previous commit" and "Select All". A status bar at the bottom right indicates "0 files selected".

Blueprints in Git repositories

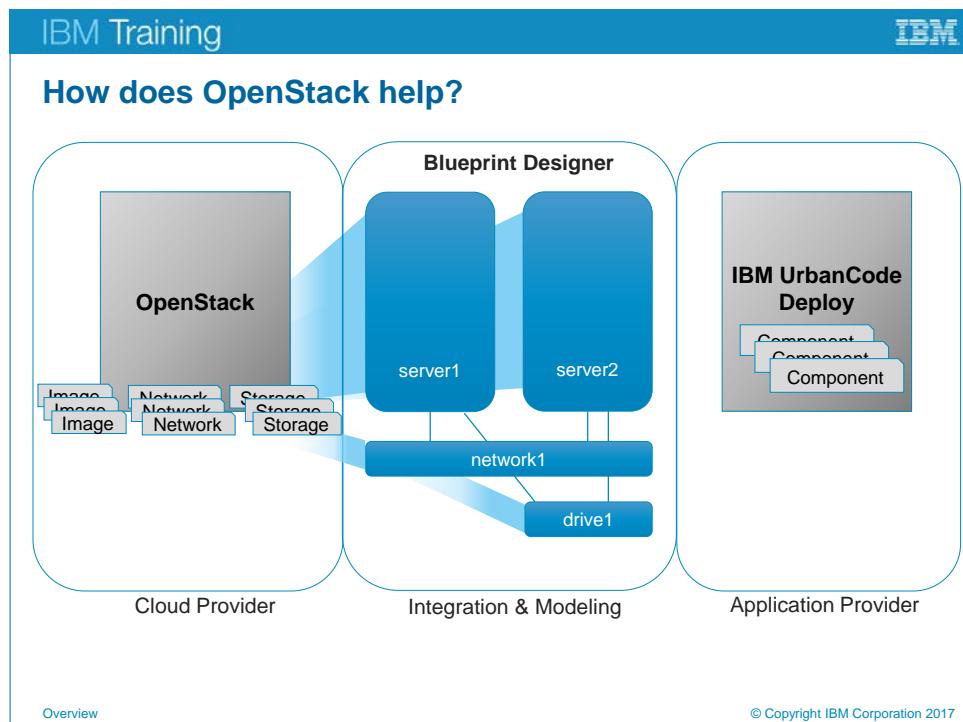
The Blueprint Designer includes a Repositories view where you can treat blueprints like source code. You can create and clone Git repositories, commit changes, and check out previous versions. From a default repository that is automatically created with the product installation, you can create and provision blueprints immediately. When you create a team, a repository is also automatically created.



Integrations to provision environments and deploy software

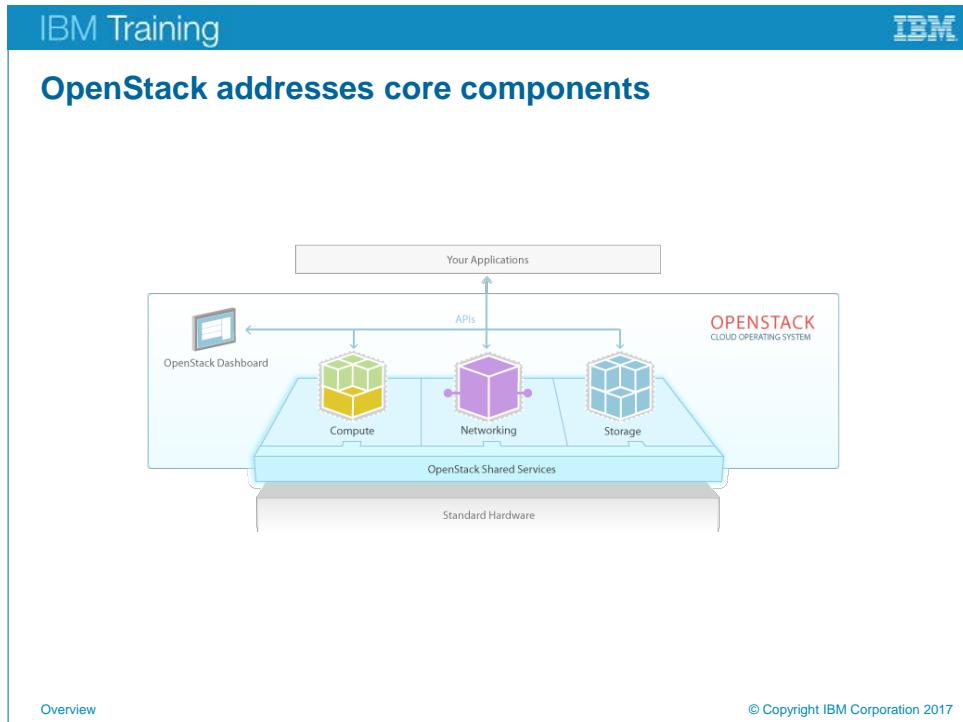
Integrations provide the Blueprint Designer with the resources to provision environments and deploy software. A Heat stack is the collection of resources that Heat creates. This collection might include instances (virtual machines), networks, subnets, routers, ports, router interfaces, security groups, security group rules, auto-scaling rules, and others.

In this high-level scenario, first the infrastructure is created; then the applications are deployed to it. The left side of this diagram shows the integration between the Blueprint Designer and a cloud system (OpenStack); the right side shows the connection to IBM UrbanCode Deploy.



How does OpenStack help?

OpenStack provides the infrastructure model. When the Blueprint Designer is integrated with a cloud system, the cloud system provides the compute resources (images, networks, and storage) that you can use to model a cloud infrastructure. You use these resources when you design a Heat stack in the Blueprint Designer.



OpenStack addresses core components

OpenStack is an open software to use public and private clouds. The Compute resource provisions and manages large networks of virtual machines. The Storage resource provisions and manages block-based and object storage. The Network resource provisions and manages network connectivity.

The screenshot displays the IBM Training interface for OpenStack services on the Horizon dashboard. The main title is "OpenStack services on the Horizon dashboard". On the left, a sidebar menu lists categories: Project, Compute, Network, Orchestration, and Identity. Under Compute, "Overview", "Instances", "Volumes", and "Images" are visible. Under Network, "Overview", "Network", "Compute", and "Identity" are listed. Under Orchestration, "Overview", "Compute", and "Identity" are listed. Under Identity, "Overview", "Compute", and "Identity" are listed. The central area shows a "bluebox" logo and a detailed "Overview" section. It includes "Limit Summary" with pie charts for AutoScale (Used 2 of 10), VCPU (Used 4 of 20), RAM (Used 6,110 of 31,200), Floating IPs (Used 1 of 10), and Security Groups (Used 2 of 10). Below this is "Usage Summary" with a date range from 2016-01-01 to 2016-01-01, showing "This Period's Total Usage: 100%". A table titled "Usage" provides detailed usage data for Compute, Network, and Volume services.

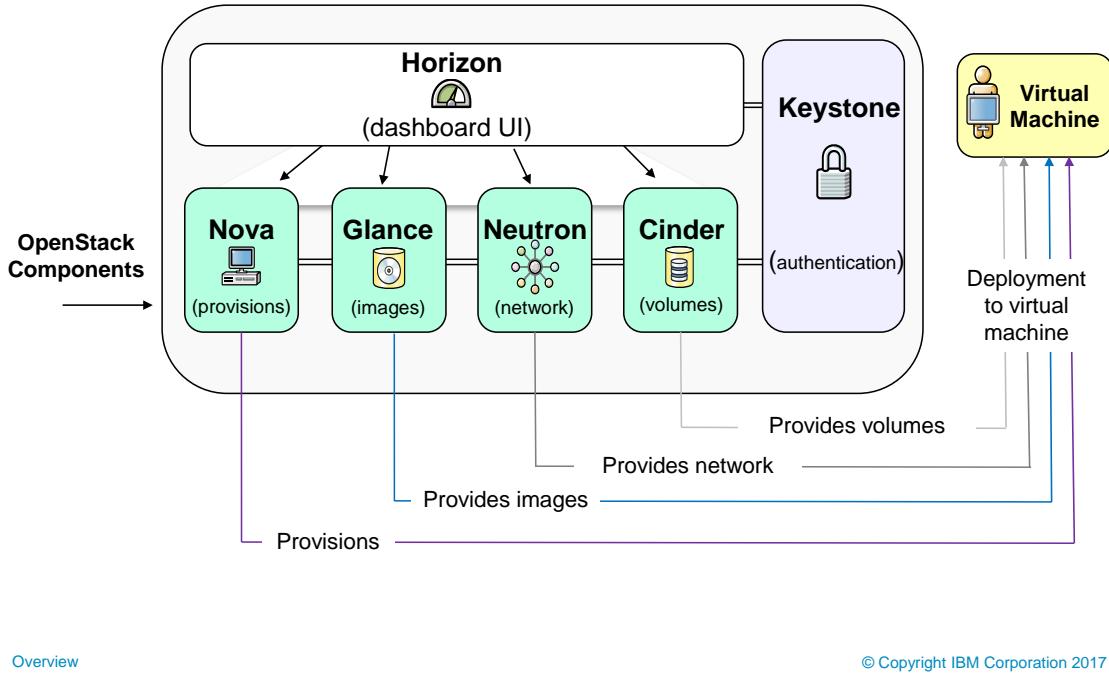
Resource Type	VCPUs	RAM	Floating IP	Security Groups
Compute	2	4010	1	2
Network	0	60GB	0	0
Volume	0	0	0	0

© Copyright IBM Corporation 2017

OpenStack services on the Horizon dashboard

The OpenStack Dashboard, called Horizon, provides an interface for administrators and users to access and provision cloud-based resources through a self-service portal. With this web GUI, you can perform most operations on your cloud. You can create volumes, launch an instance, assign IP addresses, and set access controls.

OpenStack components and their roles



Overview

© Copyright IBM Corporation 2017

OpenStack components and their roles

OpenStack is a software platform that consists of interrelated components that control processing, storage, and networking resources throughout a data center. It uses code names for these components:

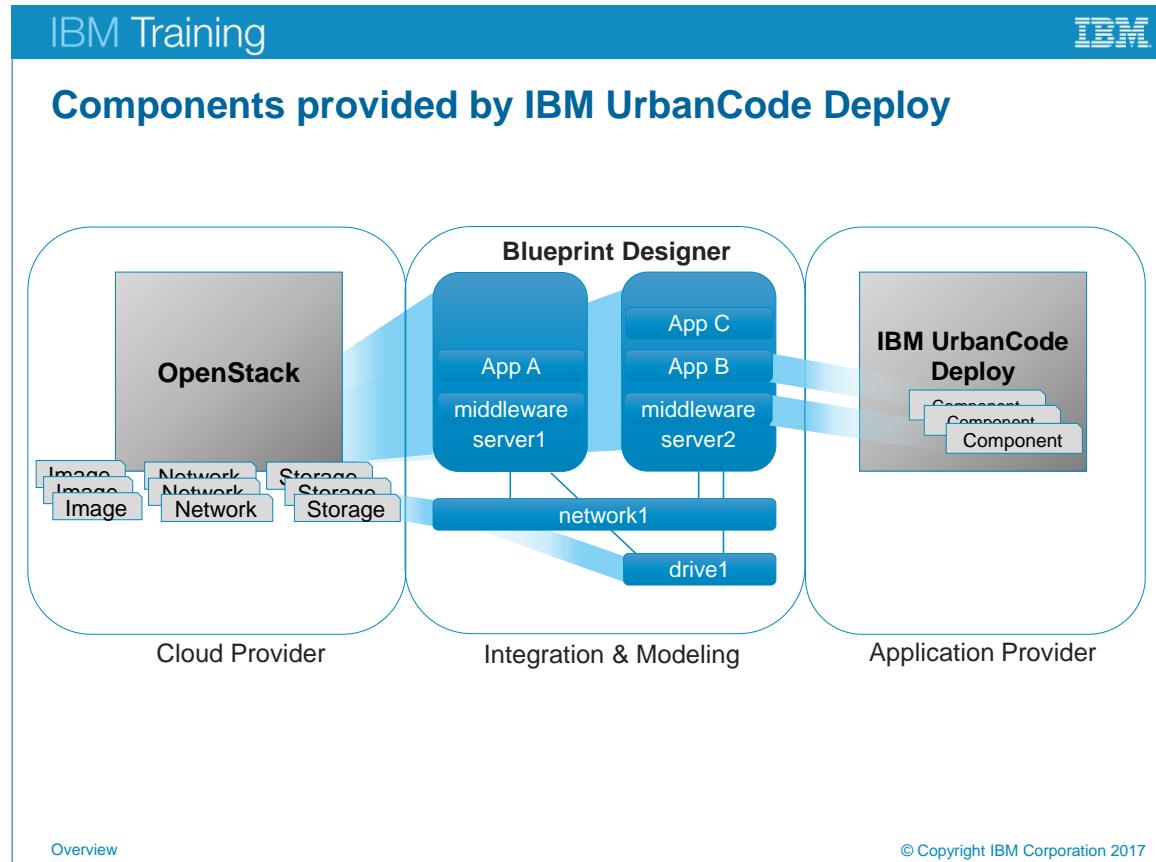
Nova (compute): Creates the virtual machine and manages the lifecycle of compute instances in an OpenStack environment.

Neutron (networking): Provides the network for the virtual machines and can make it accessible over the public network.

Cinder (block storage): Adds additional disks and provides persistent block storage to running instances. It manages the creation, attaching, and detaching of block devices to servers.

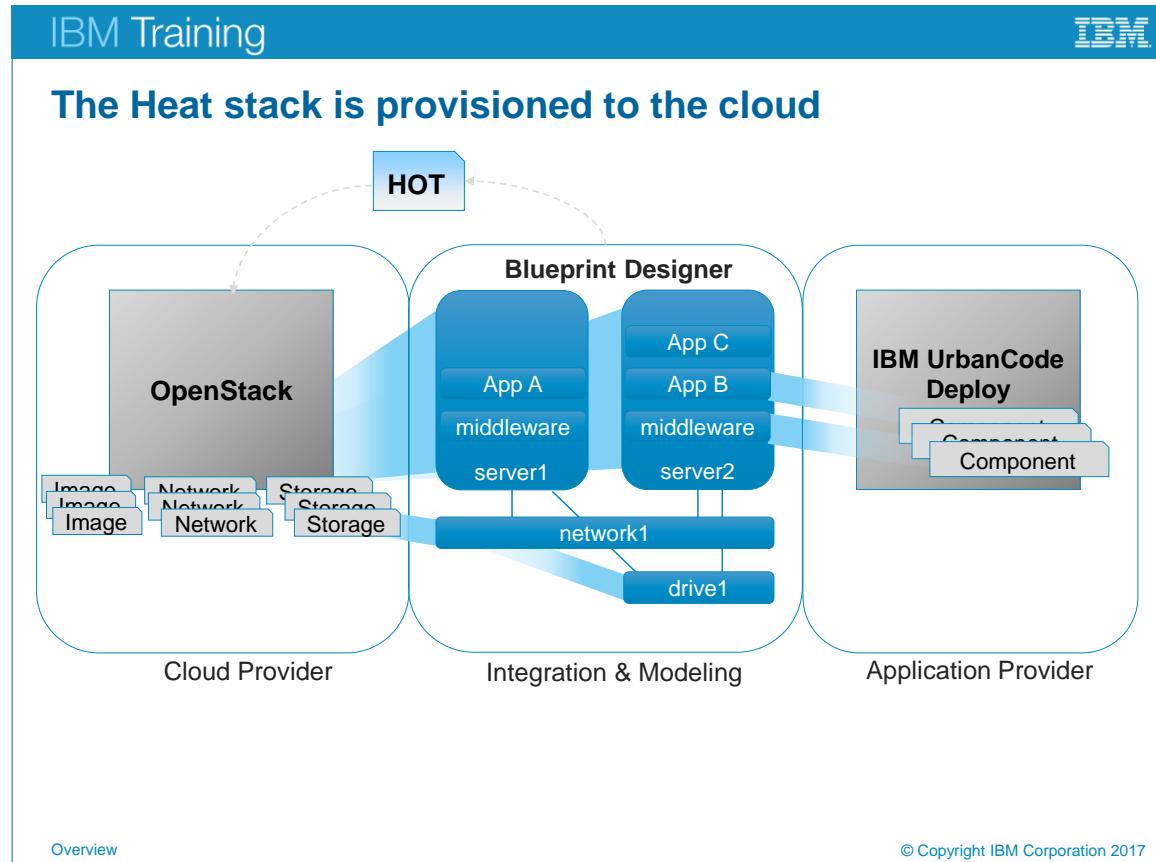
Keystone (identity): Manages users and what they have access to and is typically backed by Active Directory or other LDAP. Provides an authentication and authorization service for other OpenStack services.

Glance (image service): Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this image service during instance provisioning.



Components provided by IBM UrbanCode Deploy

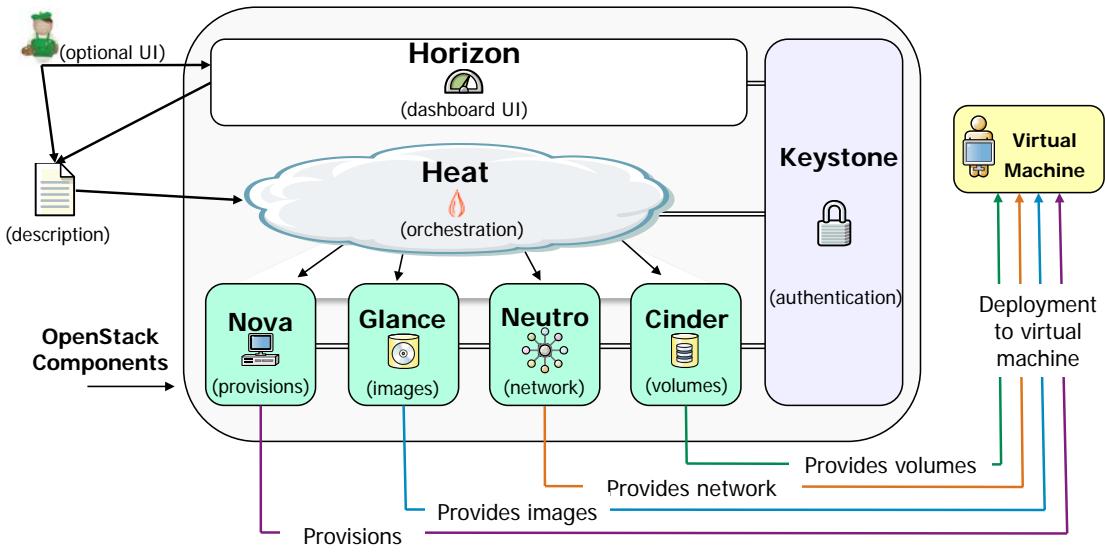
When the Blueprint Designer is connected to IBM UrbanCode Deploy, you can model the application layer of the blueprint by adding these components.



The Heat stack is provisioned to the cloud

The infrastructure model and the application is then deployed together. When you provision environments from the blueprint, the engine automatically installs the components.

The OpenStack architecture with Heat



Overview

© Copyright IBM Corporation 2017

The OpenStack architecture with Heat

The OpenStack Heat (orchestration) component provides the functions to define the virtual machines (stacks) and deploy them to the cloud.

Unit summary

- Describe the capabilities and architecture of OpenStack
- Describe the scenario of a Heat stack in software deployment
- Describe how full-stack patterns work across cloud offerings
- Explore the capabilities of the Blueprint Designer

[Overview](#)

© Copyright IBM Corporation 2017

Unit summary

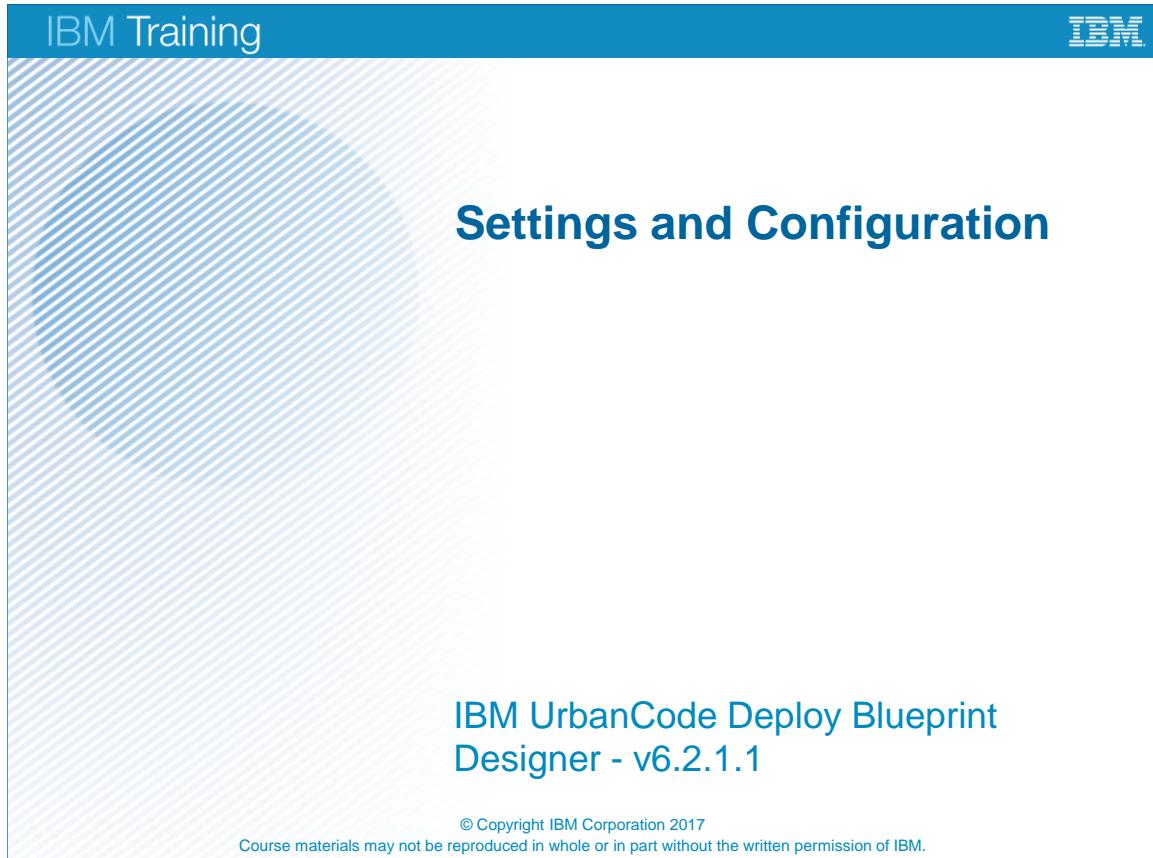
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center.

The Blueprint Designer helps you to design and deploy full-stack application environments for multiple cloud types by using software defined environments and IBM UrbanCode Deploy.

Blueprints provision an environment in the cloud and deploy applications into that cloud environment in a single step. The Blueprint Designer uses an OpenStack Heat engine to translate blueprints (by using HOT) and make them appropriate for your target cloud.

Configuration files provide properties that are unique to the target cloud when provisioning environments.

Unit 2 Settings and Configuration



The slide has a blue header bar with 'IBM Training' on the left and the IBM logo on the right. The main content area has a light blue diagonal striped background. The title 'Settings and Configuration' is centered in large blue font. Below it, the subtitle 'IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1' is also centered in blue font. At the bottom, there is a small copyright notice: '© Copyright IBM Corporation 2017' and 'Course materials may not be reproduced in whole or in part without the written permission of IBM.'

IBM Training

IBM

Settings and Configuration

IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Describe the configuration options in the Blueprint Designer
- Configure access control for users, roles, and teams
- Explore the available artifacts from the integrations in the Blueprint Designer

Settings and Configuration

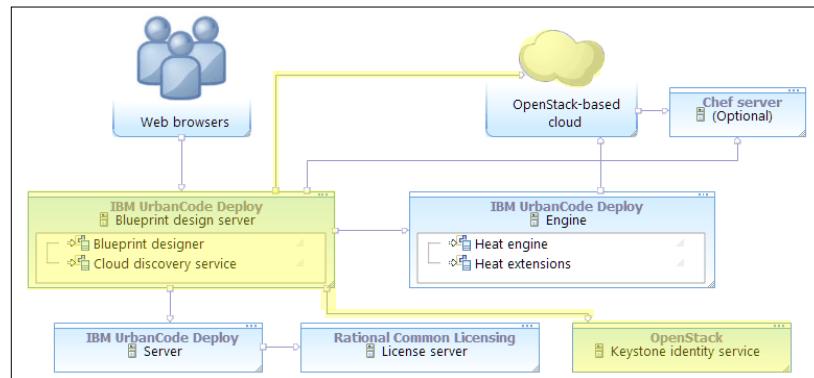
© Copyright IBM Corporation 2017

Unit objectives

To use cloud compute resources, such as images, networks, and storage, you must connect the blueprint design server to a cloud system. To deploy applications to these resources, you must also connect to the IBM UrbanCode Deploy server.

Blueprint design server

- The blueprint design server retrieves account and authentication information from the Keystone identity service



Settings and Configuration

© Copyright IBM Corporation 2017

Blueprint design server

The Keystone identity service provides authentication validation and data about users, projects, and roles. The access control settings for the blueprint design server are similar to the settings for the IBM UrbanCode Deploy server. You can create teams and roles, assign users to those teams and roles, and assign resources to those teams. However, the blueprint design server is different because it does not store its own list of users. Instead, the blueprint design server imports accounts from authentication realms, including OpenStack Keystone services, LDAP servers, and single sign-on servers. Therefore, to configure access control, you must first import user accounts from an authentication realms. Then, you associate those user accounts with roles and teams.

This diagram shows the scenario of when the blueprint design server retrieves account and authentication information from the Keystone identity service. If you are using an OpenStack cloud, the easiest way to set up authentication is to use that cloud's Keystone identity service. In this case, you log in to the blueprint design server with the same credentials that you use on the cloud.

Note: These teams and roles are separate from the teams and roles for the IBM UrbanCode Deploy server.

IBM Training IBM

Blueprint design server configuration

- Configure the blueprint design server from the Settings menu

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. At the top, there is a navigation bar with tabs: Users (highlighted in blue), Roles, Teams, Clouds, Cost Centers, Event Sources, and System Settings. Below the navigation bar is a main content area titled "UrbanCode Deploy Blueprint Designer". On the left, there is a sidebar with icons for Home, User, Team, Cloud, and Cost Center. The main content area has a sub-navigation bar with tabs: Users, Groups, and Edit. It includes buttons for Import Users and Update Users. A table lists users with columns for Name, Email, and Actions. A message at the bottom says "No users have been created yet." To the right of the main content area is a vertical sidebar titled "ADMINISTRATION" which contains links for Users, Roles, Teams, Clouds, Cost Centers, Event Sources, and System Settings. A yellow callout box labeled "Settings menu" points to the "System Settings" link in the sidebar.

Blueprint design server configuration

You access the **Settings** menu by clicking the icon in the upper-right corner of the user interface. You can configure access control settings as they pertain to users, roles, teams, and clouds.

Notice that the **ucdpadmin** user is logged in (refer to the slide). This is an administrative user that does not connect to any cloud.

In this course, you explore **Users**, **Roles**, **Teams**, **Clouds**, and **System Settings**:

- Users:** Each user represents an account on the Blueprint Designer.
- Roles:** A role is a set of permissions. For example, a role can include the permission to view, create, or edit blueprints.
- Teams:** Teams associate users with roles. When you assign a user to a team, you must also assign that user to a role.
- Clouds:** A cloud project includes a functional ID with access to a cloud system.
- System Settings:** System-wide settings are the general for the blueprint design server, such as product integrations.

Authentication realm

- An authentication realm gives access to an OpenStack cloud through user accounts

The screenshot shows the 'UrbanCode Deploy Blueprint Designer' application. In the top navigation bar, the 'Users' tab is selected. On the left sidebar, under 'Realms', there is a list with 'Internal Authentication' at the top, followed by 'OpenStack'. A 'Create New Realm' button is also visible. The main content area displays the configuration for the 'OpenStack' realm. The 'Name' field is set to 'OpenStack'. The 'Type' dropdown is set to 'OpenStack Identity Service'. The 'Identity URL' field contains the value 'http://192.168.27.100:5000/v2.0/'. Other fields include 'Allowed Login Attempts' (set to 0), 'Admin Username' (set to 'admin'), 'Admin Password' (redacted), 'Admin Tenant' (set to 'admin'), and 'Domain' (set to 'Default'). At the bottom of the form are 'Save' and 'Test Connection' buttons.

Settings and Configuration

© Copyright IBM Corporation 2017

Authentication realm

By default, the blueprint design server has an authentication realm that is named Internal Authentication, which includes the administrator and any users that you add to the server manually. You can create other authentication realms to import users from other sources.

Importing user accounts

- You import user accounts from an OpenStack server to the blueprint design server

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, under 'Realms', there is a section for 'OpenStack' authentication. A yellow box labeled 'OpenStack identity service URL' highlights the 'Identity URL' field, which contains the value 'http://192.168.27.100:5000/v2.0/'. An orange arrow points from this field to the 'Import Users' button in the main 'Users' tab. Another orange arrow points from the 'Import Users' button to the right-hand table, which displays a list of imported user accounts. A yellow box labeled 'Imported user accounts' covers the entire table area.

User	Name	Email	Actions
nova	nova		Edit Delete
neutron	neutron		Edit Delete
heat	heat		Edit Delete
glance	glance		Edit Delete
demo	demo	demo@example.com	Edit Delete
cinder	cinder		Edit Delete
admin	admin		Edit Delete

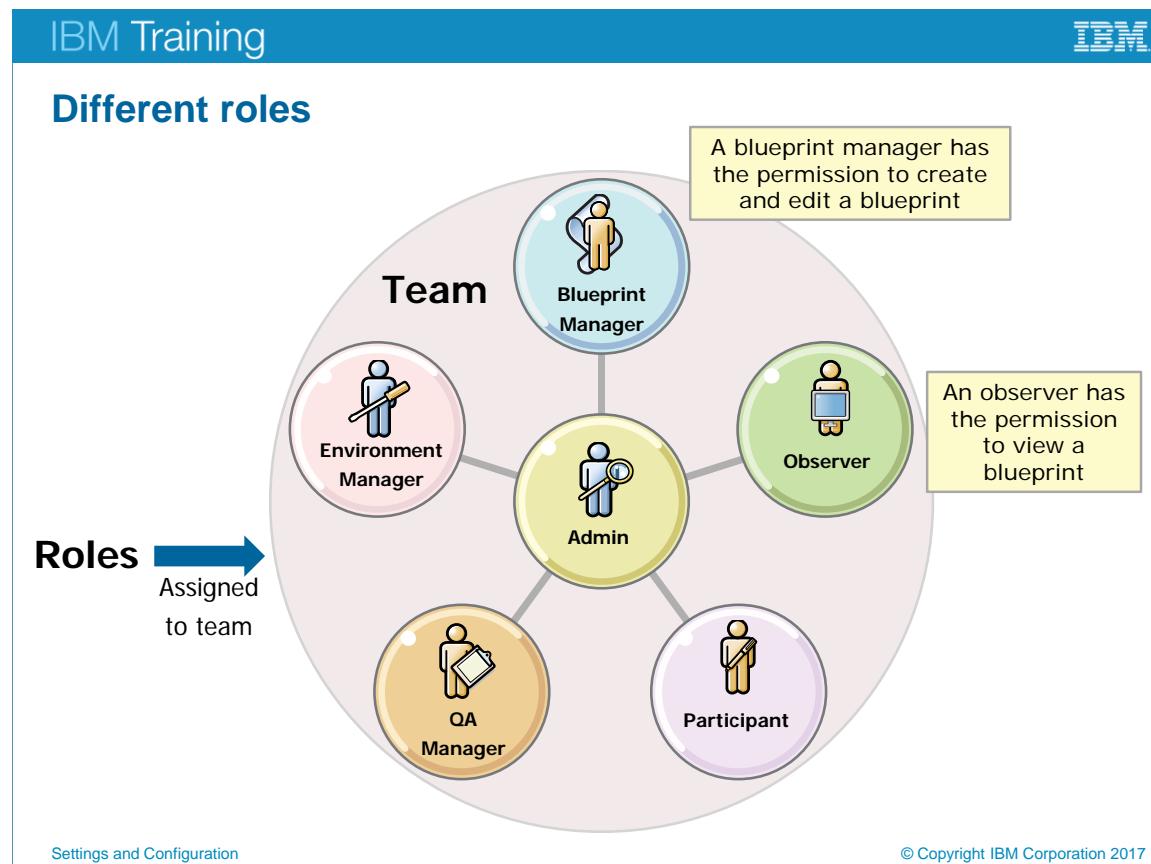
Settings and Configuration

© Copyright IBM Corporation 2017

Importing user accounts

When you create a realm, the Blueprint Designer creates the following artifacts:

- A user in the authentication realm for each user on the Keystone server.
- A cloud connection on the **Clouds** tab. (Click **Settings > Clouds**.)
- A cloud project for each project on the cloud.



Different roles

Roles define access to the various capabilities and resources of the blueprint design server. A role is defined by what a user can do to what resources, such as create a blueprint.

For example, a role can include the permission to view, create, or edit blueprints. You cannot assign roles directly to users. Instead, you assign the role to a team. Then, the users in the team have the permissions in the role.

The screenshot shows the 'Users' tab in the UrbanCode Deploy Blueprint Designer. On the left, a sidebar lists realms: 'Internal Authentication' and 'OpenStack'. A yellow box highlights 'OpenStack'. On the right, a table lists users with columns for User, Name, Email, and Actions. An orange bracket groups the first six users (nova, neutron, heat, glance, demo, cinder) under the 'OpenStack' realm heading.

User	Name	Email	Actions
nova	nova		Edit Delete
neutron	neutron		Edit Delete
heat	heat		Edit Delete
glance	glance		Edit Delete
demo	demo	demo@example.com	Edit Delete
cinder	cinder		Edit Delete
admin	admin		Edit Delete

Authentication realm

Authentication realms manage users and determine user identity within authorization realms for the server.

Authentication realms use authorization realms to associate users with groups and to determine user access. IBM UrbanCode Deploy includes an internal database for storing security information and integration with the Lightweight Directory Access Protocol (LDAP). LDAP is a widely used protocol for accessing distributed directory information over IP networks.

Each user represents an account on the Blueprint Designer.

IBM Training IBM

Users and roles

- Teams associate users with roles

The screenshot shows the 'Teams' section of the Blueprint Designer. A yellow box highlights the 'Team name' field, which is set to 'demo'. Another yellow box highlights the 'Members' list under the 'demo' team, showing a single member named 'demo'. An orange arrow points from the 'Team name' field to the 'Members' list. A third yellow box highlights the 'Roles' section, where 'Blueprint Manager' is selected. A fourth yellow box highlights the 'Permission "set" with capabilities' section, which lists 'Blueprint Manager' under 'Artifact' and 'Create Blueprint', 'Edit Blueprint', and 'View Blueprint' under 'Capability'. A bracket groups the 'Roles' and 'Permission' sections.

Team name

Team members in team demo

Blueprint Manager role capabilities

Permission "set" with capabilities

© Copyright IBM Corporation 2017

Users and roles

You assign users to teams on a specific role basis. For example, after a user is imported into the blueprint design server, you can assign the user to a team and then assign a role with access to whatever capabilities are defined in that role.

You must add users to teams to give the users permission to work with blueprints. Teams also include a list of resources that the team owns and cloud projects that the team can use. To set up a team, you specify the members of the team, including users and groups. Then, you assign roles to the users and groups to specify their permissions.

IBM Training IBM

User access

- Give users access to cloud systems through cloud projects

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, there's a sidebar with icons for Home, Clouds, Cost Centers, and System Settings. The main navigation bar at the top includes links for Users, Roles, Teams, Clouds (which is highlighted in blue), Cost Centers, Event Sources, and System Settings. The user 'ucdpadmin' is logged in.

Clouds View: A 'Clouds' section shows an 'OpenStack' provider with a 'Cloud system' entry. An orange arrow points from this entry to a yellow box labeled 'Cloud authorization for the team demo'.

Authorization Dialog: A modal window titled 'Authorization' is open, showing fields for 'Name (Project name from OpenStack)' (set to 'demo'), 'Functional ID' (set to 'demo'), 'Password' (set to '****'), and 'Domain' (set to 'Default'). Buttons for 'Save' and 'Cancel' are at the bottom.

Teams View: A 'Teams' section shows a team named 'demo' (Internal Team). An orange arrow points from the 'Cloud system' entry in the Clouds view to this team. The 'Cloud Authorization' tab is selected in the team's configuration dialog, which lists 'demo@OpenStack'.

At the bottom of the interface, there are links for 'Settings and Configuration' and '© Copyright IBM Corporation 2017'.

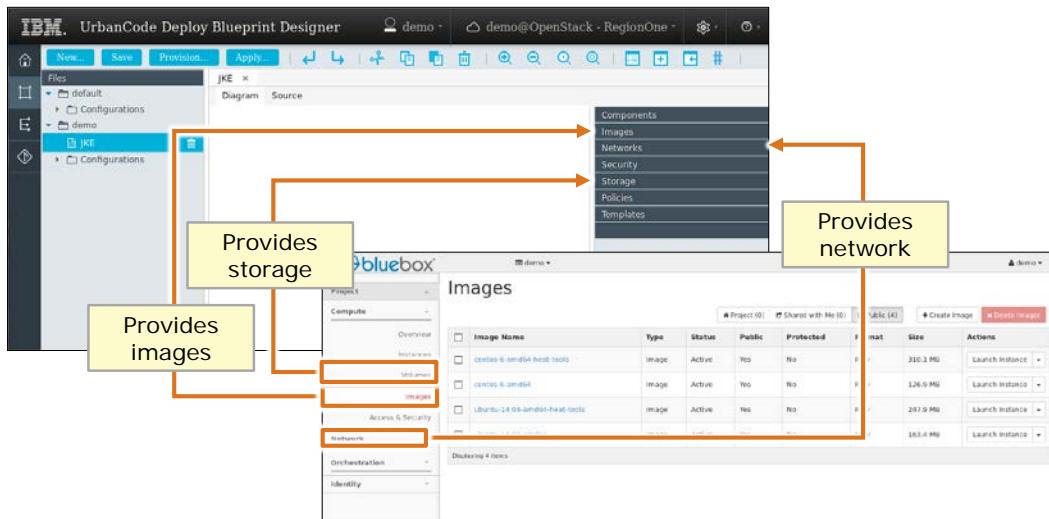
User access

When you import users from the authentication realm, the blueprint design server creates a cloud connection that is based on this authentication realm and cloud projects that are based on the projects on the cloud. A cloud project includes a functional ID with access to a cloud system. When you assign a cloud project to a team, users can work with the resources. The roles that you selected for the users determine what the users can do with those resources.

Demo user access

- The demo user has access to the OpenStack cloud resources

Demo user is logged in



Settings and Configuration

© Copyright IBM Corporation 2017

Demo user access

The **ucdpadmin** user does not have the authorization to access the OpenStack cloud. The **demo** user does have the authorization to access the cloud resources. So when the **demo** user is logged in to the Blueprint Designer, the resources become available and can be used in blueprints.

IBM Training IBM

Server connections (1 of 2)

- Connect the IBM UrbanCode server to the blueprint design server

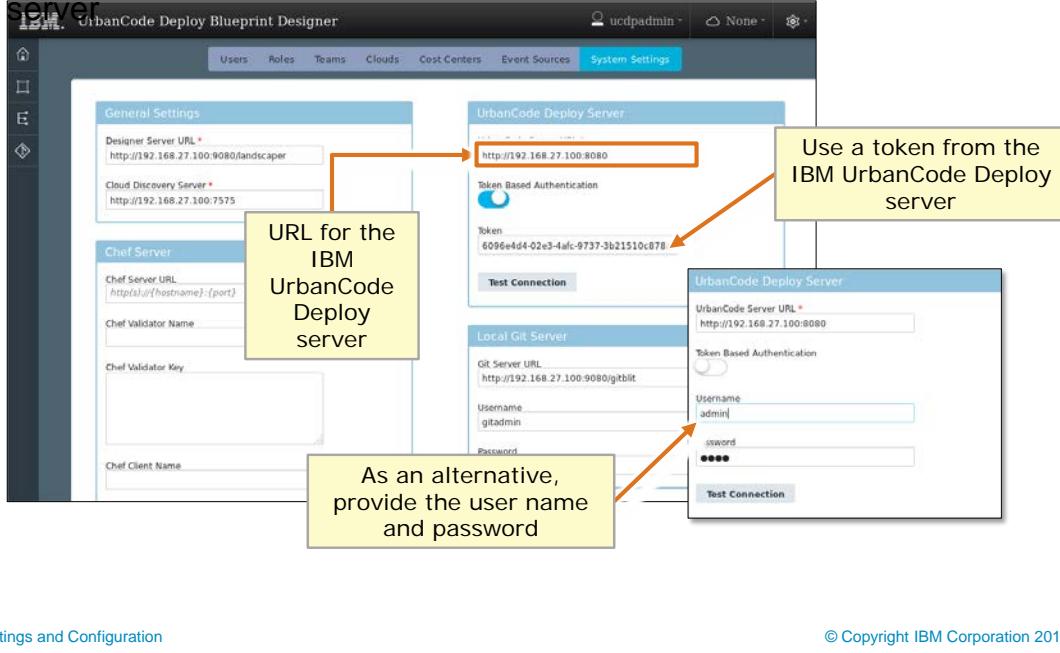
The screenshot shows the 'Edit Blueprint Designer Integration' dialog box overlaid on the main application interface. The dialog contains fields for 'Name' (set to 'landscaper'), 'Description', 'URL' (set to 'http://192.168.27.100:9080/landscaper'), and 'Response' (with two radio button options: 'Always authenticate with one blueprint designer user name' and 'Always authenticate with one UrbanCode Deploy user'). Below the dialog, the main application interface is visible, showing the 'Automation' section with various sub-options like 'Automation Plugins', 'Source Configuration Plugins', etc. A specific link 'Blueprint Designer Integrations' is highlighted with a red box and an arrow pointing to it.

Server connections

Components in IBM UrbanCode Deploy are available in the Blueprint Designer when the servers are connected, which in most cases you connect at installation time, but you can also connect them later. You can make this connection by clicking **Settings > Blueprint Designer Integrations** in IBM UrbanCode Deploy.

Server connections (2 of 2)

- Connect the blueprint design server to the IBM UrbanCode Deploy server



Settings and Configuration

© Copyright IBM Corporation 2017

From the Settings page of the Blueprint Designer, you connect the blueprint design server to the IBM UrbanCode Deploy server by specifying its complete URL. You can authenticate with either a token or a user name and password.

Tokens provide authorization for agents, agent relays, users, and external systems or applications from the server. Agents use tokens when they run process steps and communicate with the IBM UrbanCode Deploy server and external services. Users can use tokens with the command-line interface (CLI) client instead of supplying a user name and password in certain situations. To specify a token, you must first generate it from the **Settings > Tokens** page in IBM UrbanCode Deploy. Then, you add that value to the **Token** field in the Blueprint Designer.

To specify a user name and password, first set **Token Based Authentication** to **Off**. Then, type the IBM UrbanCode Deploy server user name and password in the fields.

IBM UrbanCode Deploy components

- The components in IBM UrbanCode Deploy are available in the Blueprint Designer

The screenshot shows the IBM UrbanCode Deploy Blueprint Designer interface. On the left, there's a sidebar with 'Components' selected. Below it, a table lists various components:

Name	Description	Created	Last Modified	Owner
jke-db	This component contains the process to setup MySQL database on MySQL Server.	7/29/2014, 7:13 PM	7/29/2014, 7:13 PM	admin
jke-war	JKE Banking Sample Application Resource Archive	7/29/2014, 7:13 PM	7/29/2014, 7:13 PM	admin
MySQL Server	... contains the process to install MySQL Server	7/29/2014, 7:12 PM	7/29/2014, 7:12 PM	admin
uccd-agent-linux-x86_64	Agent packages wth embedded JRE for x86_64 Linux	6/13/2014, 8:58 AM	6/13/2014, 8:58 AM	admin
uccd-agent-win-x86_64	Agent packages wth embedded JRE for x86_64 Windows	6/13/2014, 8:58 AM	6/13/2014, 8:58 AM	admin
WebSphere Liberty Profile	This component contains the binaries and process to install WebSphere Liberty Profile	7/29/2014, 7:12 PM	7/29/2014, 7:12 PM	admin

On the right, a sidebar titled 'Components' lists several items with icons:

- Referenced Component
- jke-db
- jke-war
- MySQL Server
- uccd-agent-linux-x86_64
- uccd-agent-win-x86_64
- WebSphere Liberty Profile

An orange arrow points from the 'Components' sidebar to the 'jke-db' row in the main table.

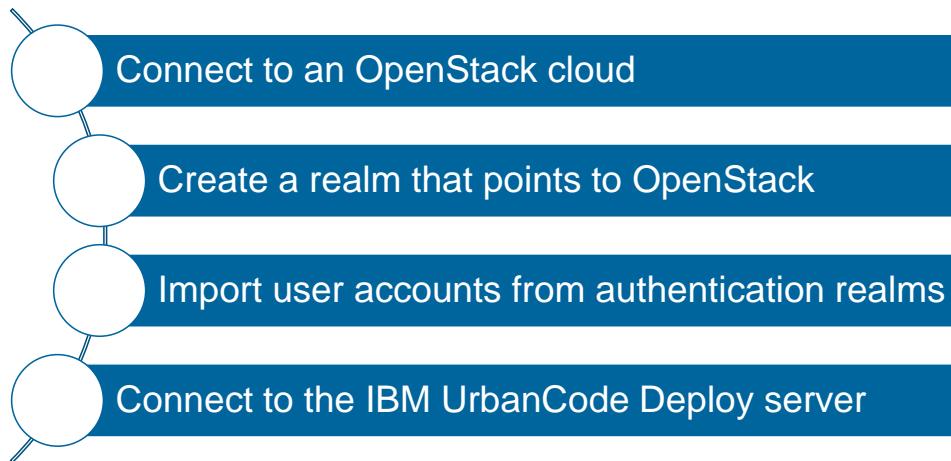
Settings and Configuration

© Copyright IBM Corporation 2017

IBM UrbanCode Deploy components

You can now model the application layer of a blueprint.

Overview: Configure security for the blueprint design server



Settings and Configuration

© Copyright IBM Corporation 2017

Overview: Configure security for the blueprint design server

The four major steps involved with configuring security for the blueprint design server are as mentioned on the slide.

Unit summary

- Describe the configuration options in the Blueprint Designer
- Configure access control for users, roles, and teams
- Explore the available artifacts from the integrations in the Blueprint Designer

Settings and Configuration

© Copyright IBM Corporation 2017

Unit summary

To use the accounts on an OpenStack Keystone server, you create an authentication realm that points to the Keystone server and configure security for that authentication realm.

Authentication realms manage users and determine user identity within authorization realms for the server. Authorization realms are used by authentication realms to associate users with groups and to determine user access.

Roles define access to the various capabilities and resources of the blueprint design server. You assign users to teams on a specific role basis. Users can access a cloud project by using a cloud's functional ID to request cloud resources.

About the exercises

Login instructions to start with the lab:

- Log in into the training environment
- Access the Blueprint Designer

Settings and Configuration

© Copyright IBM Corporation 2017

About the exercises

1. Open the virtual image.

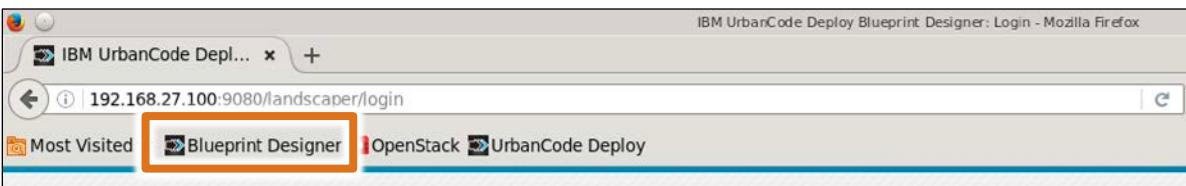


2. Enter the user name **vagrant**, the password **vagrant**, and then click the icon.

You are now logged into the training environment.



3. On the desktop, click the **Firefox Web Browser** icon.
4. In the browser's Bookmarks Toolbar, click the **Blueprint Designer** bookmark.



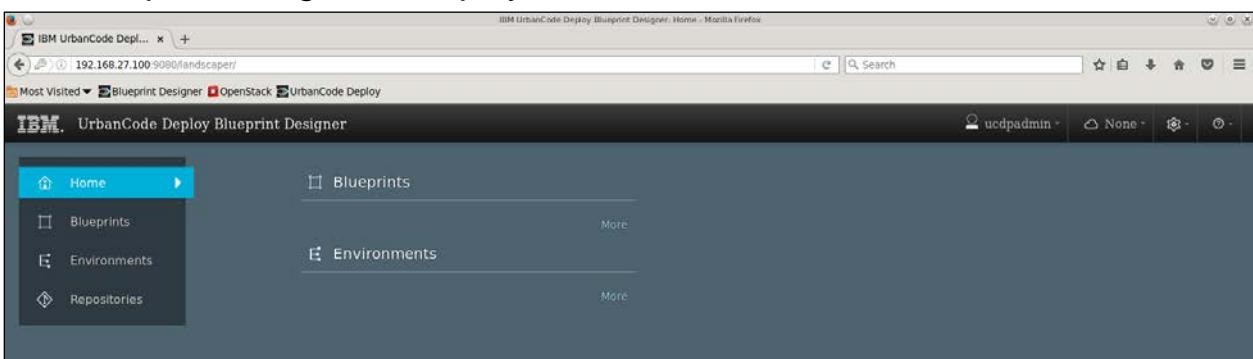
Alternatively, you may type the URL
<http://192.168.27.100:9080/landscaper/login>, in the address field.
The Blueprint Designer page is displayed.

5. At the Blueprint Designer login page, type ucdpadmin in the **User Name** field and ucdpadmin in the **Password** field.

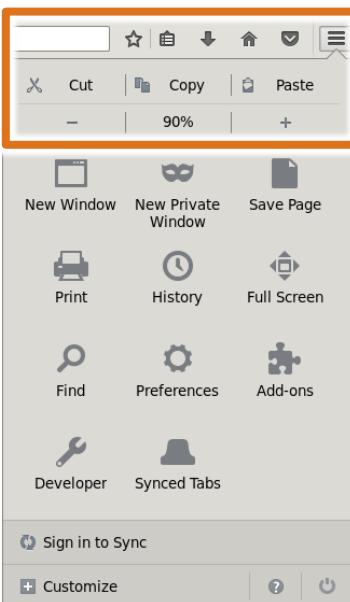


6. Click **Log In**.

The Blueprint Designer is displayed.



Note: For a better working on the tools, zoom out the browser window to 90%.



Exercises: Configuring the blueprint design server integrations

In the following exercises, you will:

- Connect the blueprint design server to an OpenStack cloud and the IBM UrbanCode Deploy server
- Import users from the OpenStack Keystone service and grant privileges to teams and users in the Blueprint Designer
- Explore the OpenStack dashboard and view the cloud resources

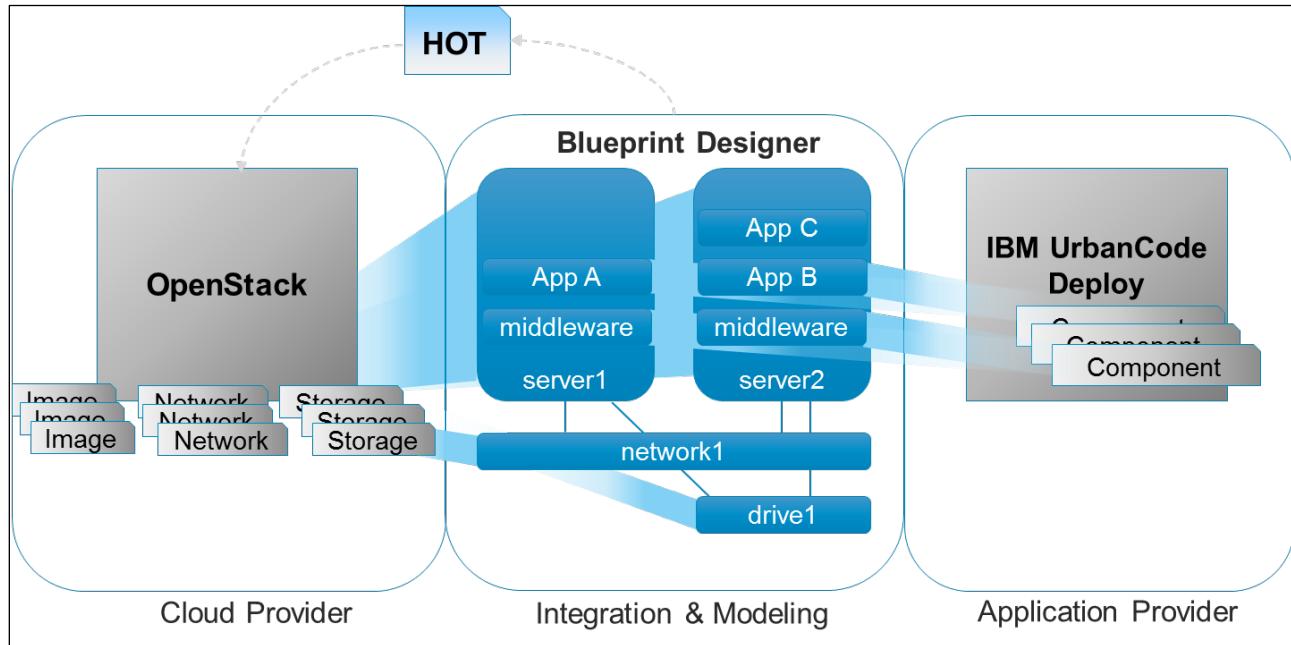
[Settings and Configuration](#)

© Copyright IBM Corporation 2017

Exercises: Configuring the blueprint design server integrations

The Blueprint Designer accelerates application development by using a single blueprint to combine the tasks of provisioning a cloud system with the tasks of modeling an application.

In the following exercises, you view the connection between the blueprint design server and an OpenStack cloud, which provides the image, network, and storage resources for creating a cloud infrastructure in your blueprints. Then, you view the integration between the blueprint design server and the IBM UrbanCode Deploy server, which provides the components that you deploy to that cloud infrastructure.



Although this lab focuses on using OpenStack as the cloud provider, you can also use other cloud types, such as Azure, Amazon EC2, vCenter, and Softlayer.

IBM Training

Exercise 1

Import users from the OpenStack Keystone service

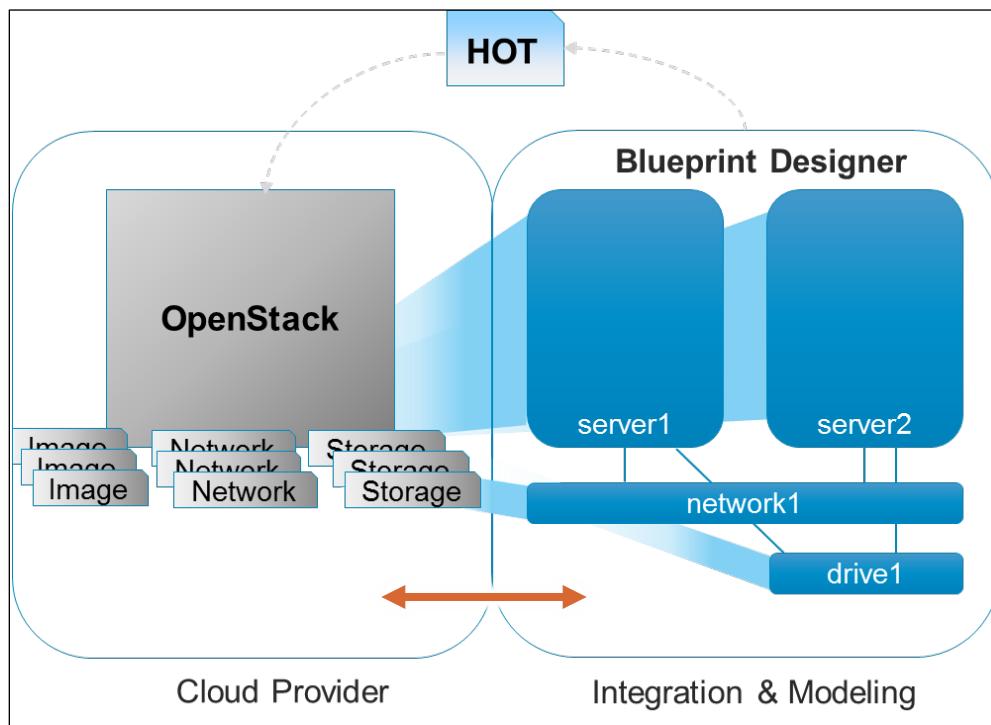
Settings and Configuration

© Copyright IBM Corporation 2017

Exercise 1: Import users from the OpenStack Keystone service

Exercise 1: Import users from the OpenStack Keystone service

In this exercise, you review the integration settings of the blueprint design server and the OpenStack cloud environment. The following section of this diagram illustrates the connection and the resources that are available to you for designing blueprints.



You connect to an OpenStack cloud by using the user accounts from the OpenStack Keystone server, which provides authentication to the OpenStack cloud connection. First, you review the settings in the authentication realm that points to the Keystone server. Then, you configure security for that authentication realm.

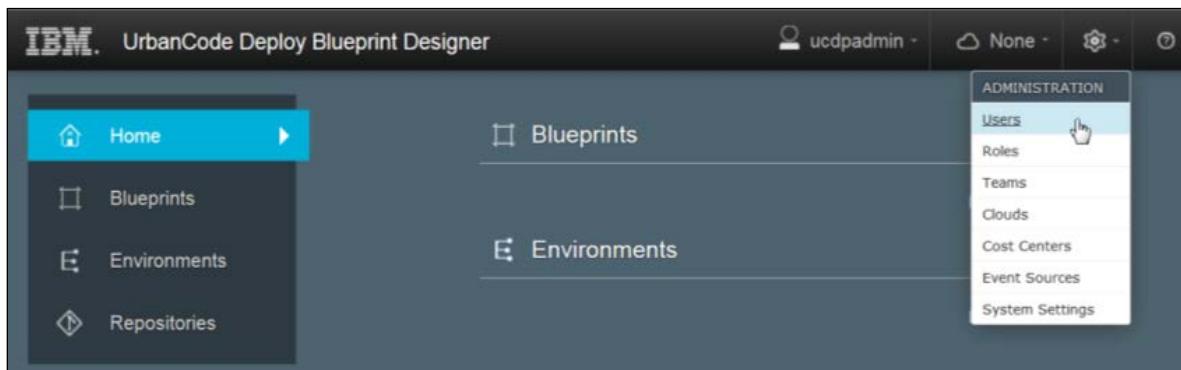
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 1: Tasks and results

Task 1. View the authentication realm settings.

To use the accounts on an OpenStack Keystone server for authentication to the OpenStack cloud connection, you must have an authentication realm that points to the Keystone server. Then, you can configure security for that authentication realm. Authentication realms manage users and determine user identity within authorization realms for the server.

7. If needed, log in to the Blueprint Designer by typing `ucdpadmin` in the **Username** field and `ucdpadmin` in the **Password** field.



8. In the Realms panel, click **OpenStack**, and click the **Edit** tab.

The screenshot shows the 'Edit' tab for the 'OpenStack' realm. The 'Name' field is set to 'OpenStack'. The 'Type' dropdown is set to 'OpenStack Identity Service'. Under 'OpenStack Identity Service', the 'Identity URL' is 'http://192.168.27.100:5000/v2.0/'. The 'Timeout in Minutes' is '60'. The 'Facing Type' is 'Public'. The 'Orchestration Engine URL' is 'http://192.168.27.100:8004'. On the right side, there are fields for 'Admin Username' (admin), 'Admin Password' (redacted), 'Admin Tenant' (admin), and 'Domain' (Default). At the bottom are 'Save' and 'Test Connection' buttons.

In this exercise, the settings are already configured for this cloud server.

9. Click **Test Connection** to ensure the OpenStack connection is valid, and click **Close**.

Task 2. Import users from the OpenStack Keystone server.

When you import the users from an authentication realm, the blueprint design server creates a cloud connection that is based on that authentication realm and cloud projects that are based on the tenants or projects on the cloud.

1. For the OpenStack realm, click the **Users** tab, and then click **Import Users**.

The screenshot shows the 'Users' tab selected. The 'Import Users' button is highlighted with a cursor icon. Below it is a table with columns 'User', 'Name', 'Email', and 'Actions'. A message at the bottom says 'No users have been created yet.' and there is a 'Refresh' link.

2. Click **Import** to confirm the request.

The users are imported into the blueprint design server from OpenStack Keystone. For example, you can see that the **admin** user and other internal users for neutron, glance, and so on are imported.

The screenshot shows the 'Users' tab selected in the top navigation bar. On the left, there's a sidebar titled 'Realms' with 'Internal Authentication' and 'OpenStack' selected. A 'Create New Realm' button is also present. The main content area has tabs for 'Users', 'Groups', and 'Edit'. Below these are buttons for 'Import Users' and 'Update Users'. A table lists the imported users:

User	Name	Email	Actions
nova	nova		Edit Delete
neutron	neutron		Edit Delete
heat	heat		Edit Delete
glance	glance		Edit Delete
demo	demo	demo@example.com	Edit Delete
cinder	cinder		Edit Delete
admin	admin		Edit Delete

At the bottom, there are links for '7 records - Refresh Print', a page navigation section (1 / 1), and a 'Rows' dropdown set to 10.

IBM Training

IBM

Exercise 2

Grant privileges to teams and users

Settings and Configuration

© Copyright IBM Corporation 2017

Exercise 2: Grant privileges to teams and users

Exercise 2: Grant privileges to teams and users

In this exercise, you assign the OpenStack cloud to the demo project; then, you grant the demo user the privileges to manage blueprints and environments.

To do this, you will:

- Connect the OpenStack cloud project to the demo project.
- Assign the team to the OpenStack cloud project.
- Grant Blueprint Designer access to the demo user.
- Log in as the demo user to create blueprints.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 2:

Tasks and results

Task 1. Connect the OpenStack cloud project to the "demo" project.

A cloud project for the Blueprint Designer represents a tenant or project on a cloud system. You can give users access to cloud systems through cloud projects.

1. From the top navigation, click **Clouds**.
2. In the Clouds panel, click **OpenStack**.
3. In the Authorization panel, click the **demo** project.
4. To add the following credentials for the demo user within the demo team, in the **Functional ID** field, type **demo**; in the **Password** field, type **labstack**.

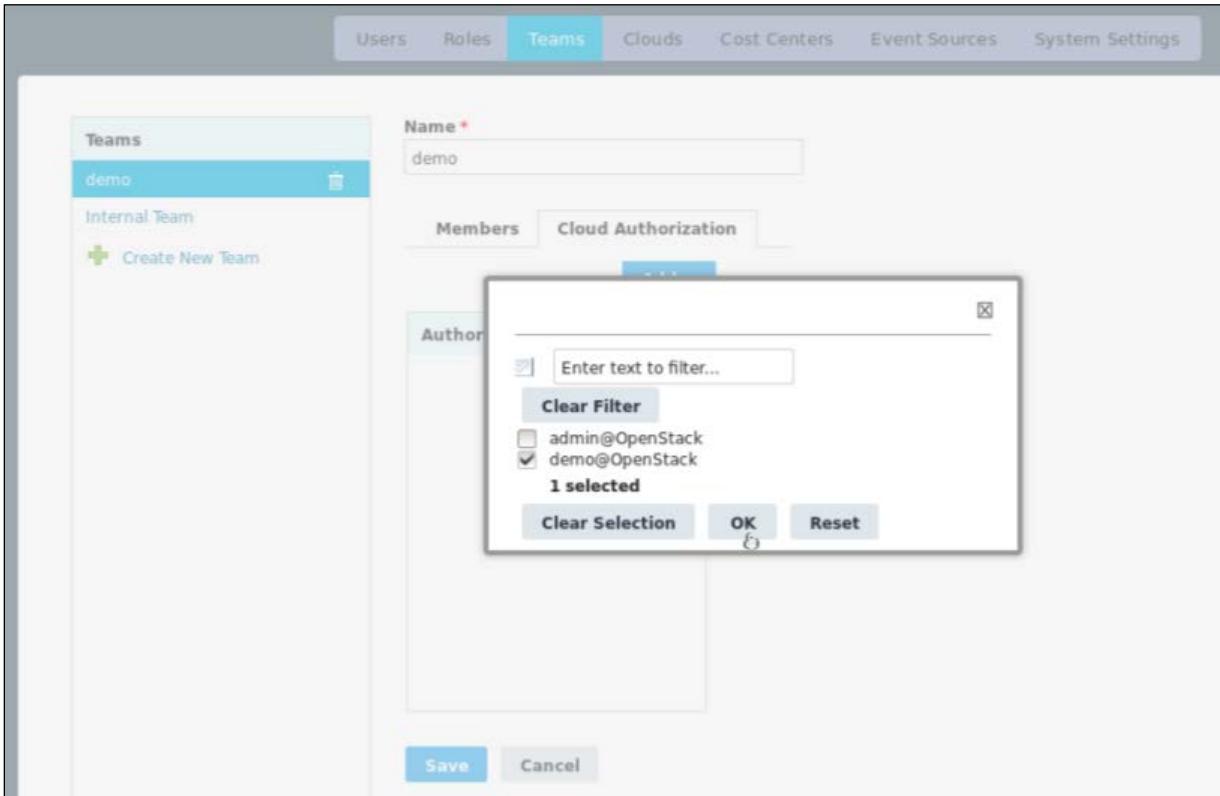
5. Click **Test Connection** to verify the credentials, and then close the Connection Successful window.
6. Click **Save**.

Task 2. Assign the team to the OpenStack cloud project.

Users on a team have access to the cloud projects that are associated with that team. You can grant specific capabilities to a user by assigning roles.

1. From the top navigation, click the **Teams** tab.
2. In the Teams panel, click **demo**, and click the **Cloud Authorization** tab.

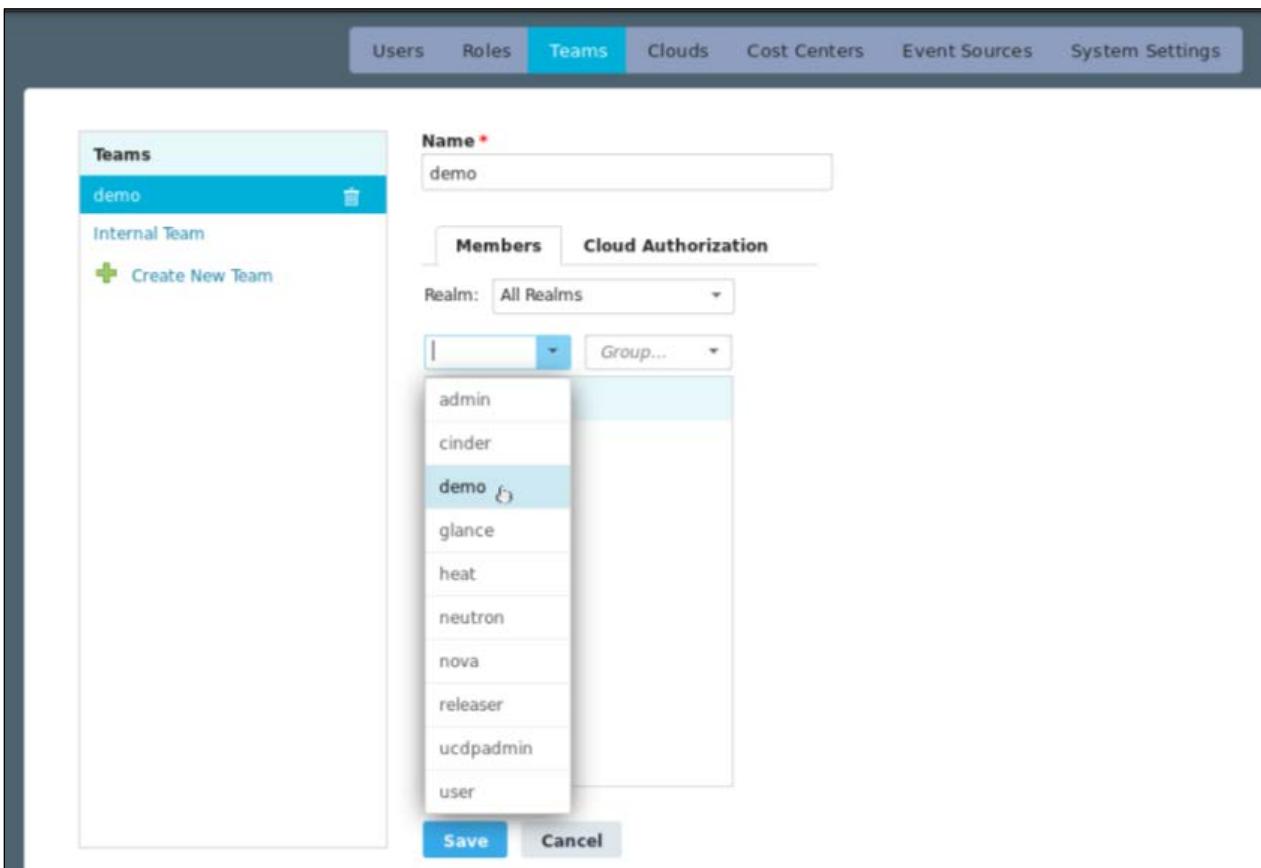
3. Click **Add**, and select **demo@OpenStack**.



4. Click **OK**, then click **Save**.

Task 3. Grant Blueprint Designer access to the demo user.

1. In the **Teams** panel, ensure that **demo** is selected.
2. To add the *demo* user to the *demo* team, from the **Members** tab, open the **User** drop-down list, and click the **demo** user.



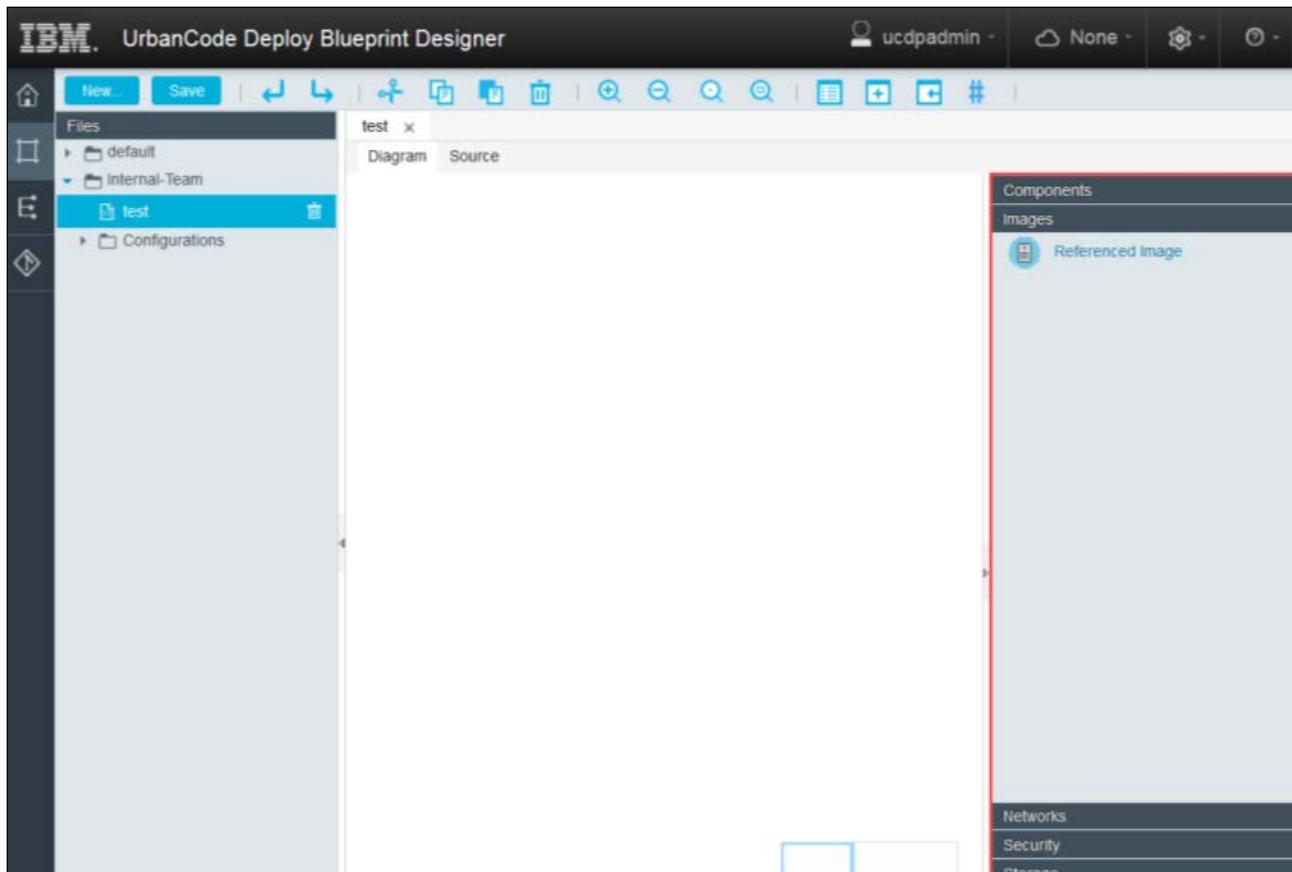
3. In the Roles section, select **Blueprint Manager** and **Environment Manager** for the demo user.

4. Click Save.

The screenshot shows the 'Teams' page in the IBM Cloud console. The top navigation bar includes 'Users', 'Roles', 'Teams' (which is highlighted in blue), 'Clouds', 'Cost Centers', 'Event Sources', and 'System Settings'. On the left, there's a sidebar titled 'Teams' with a list containing 'demo' (marked as an 'Internal Team') and a 'Create New Team' button. The main content area has tabs for 'Members' (selected) and 'Cloud Authorization'. Under 'Members', there's a table with one row for 'demo'. In the 'Roles' section, several checkboxes are available: 'Admin' (unchecked), 'Blueprint Manager' (checked), 'Environment Manager' (checked), 'Observer' (unchecked), 'Participant' (unchecked), and 'QA Manager' (unchecked). At the bottom are 'Save' and 'Cancel' buttons.

5. Log out of the Blueprint Designer. Now you can log in to the Blueprint Designer as the demo user, which is a user account on the Keystone server who has authorization to the OpenStack cloud.

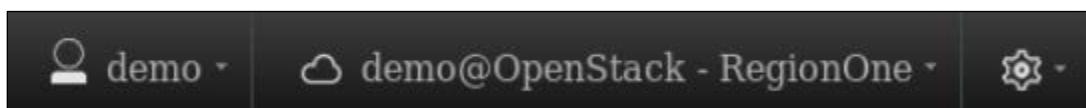
Note: As the **ucdpadmin** user, you do not have authorization to access the OpenStack cloud. In the second lab, you create a blueprint as the demo user. Currently, while logged in as the ucdpadmin, you would not see any artifacts from the OpenStack cloud if you were to look at the resource drawers to the right of a new blueprint.



When you log in to the Blueprint Designer as the **demo** user, the resources become available, and you can use them in blueprints.

Task 4. Log in as the demo user to create blueprints.

1. Log in to the Blueprint Designer by typing **demo** in the **Username** field and **labstack** in the **Password** field.
2. At the top of the page, the cloud connection and project display as **demo@OpenStack - RegionOne**. When you edit blueprints, the Blueprint Designer shows the resources that are available from the cloud that the user account is associated with.



Exercise 3

Analyze and extend the OpenStack environment

Settings and Configuration

© Copyright IBM Corporation 2017

Exercise 3: Analyze and extend the OpenStack environment

Exercise 3: Analyze and extend the OpenStack environment

In this exercise, you use the OpenStack dashboard, called Horizon, to get familiar with the OpenStack resources, such as images, networks, and storage.

To do this, you will:

- Analyze and extend the OpenStack environment

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

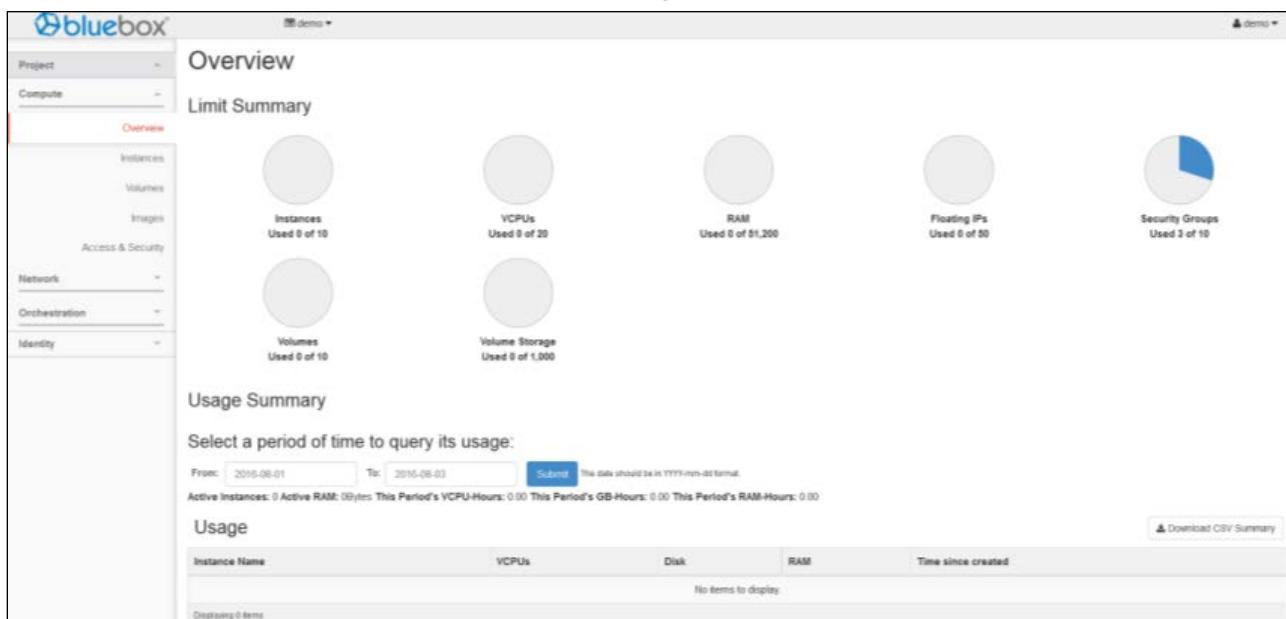
Exercise 3:

Tasks and results

Task 1. Analyze and extend the OpenStack environment.

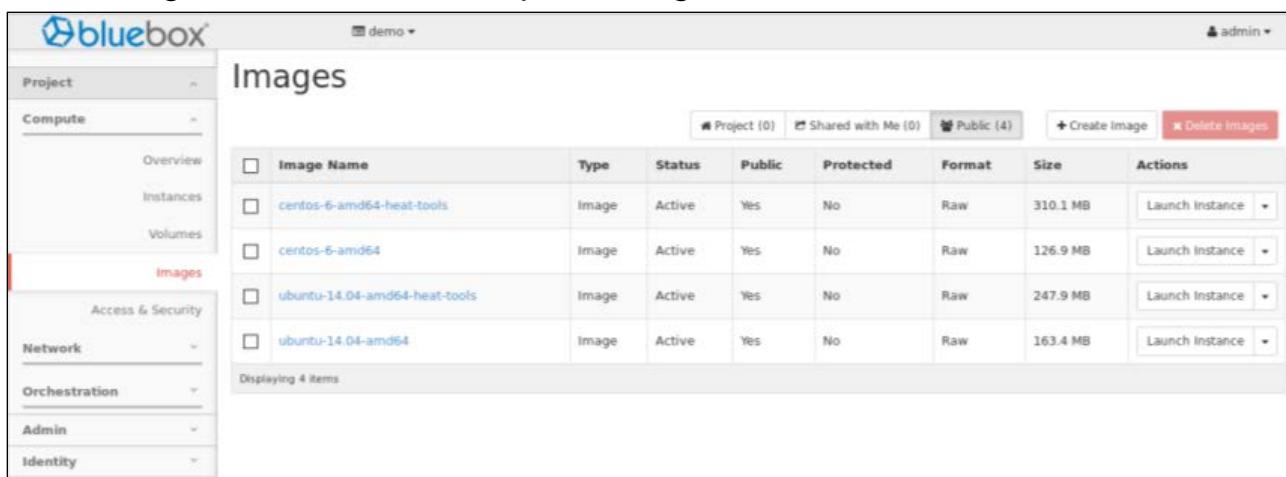
- From the browser, open a new tab and click the **OpenStack** bookmark, or type the URL, <http://192.168.27.100/dashboard/auth/login>, in the address field.
- To log in to the default domain, type **demo** in the **User Name** field and **labstack** in the **Password** field.

The dashboard opens to the overview page for the project.

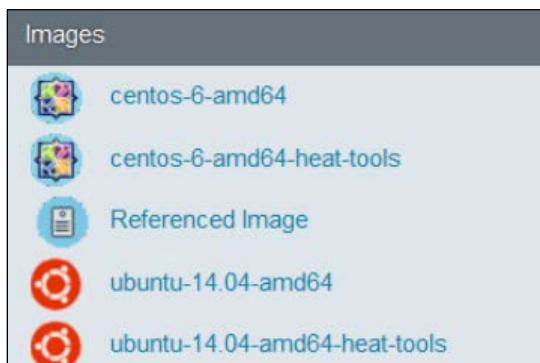


- In the panel on the left, click **Images**.

The images section stores and retrieves virtual machine disk images. OpenStack makes use of this during instance provisioning. The four images in the following screen capture are the same as the ones that are made available to the images drawer of the Blueprint Designer:



Here are the same images that you will see in the Blueprint Designer in the next lab:



- Click **Volumes**. You see an empty list.

The screenshot shows the "Volumes" section of the Blueprint Designer. On the left is a sidebar with "Project" and "Compute" dropdowns, and "Overview", "Instances", "Volumes", "Images", "Access & Security", "Network", "Orchestration", and "Identity" options. The main area is titled "Volumes" with tabs for "Volumes" and "Volume Snapshots". Below is a table header with columns: Name, Description, Size, Status, Type, Attached To, Availability Zone, Bootable, Encrypted, and Actions. A message at the bottom says "No items to display".

Storage volumes represent new virtual disks that are created when the blueprint is provisioned and enable persistent storage. It is a detachable block storage device, similar to a USB hard drive, which you can attach to a running instance. You can also detach it and attach it to another instance at any time. This lab does not cover storage volumes.

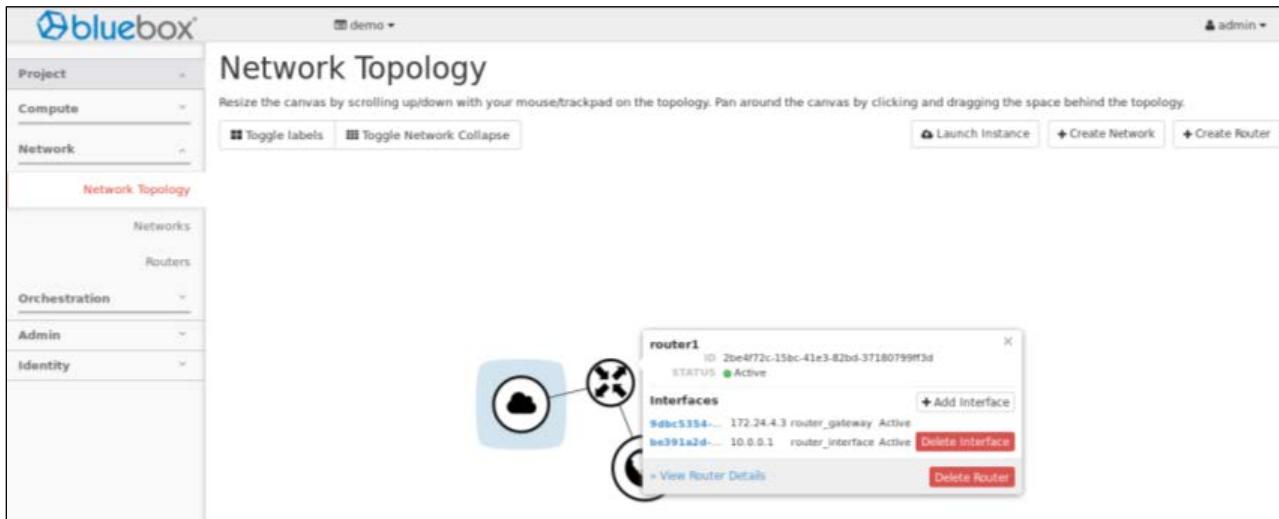
Note: In the Blueprint Designer, the Storage drawer shows the cloud storage volumes.

- Click **Project > Network > Network Topology**.

The **Network Topology** tab shows the functions that are related to software-defined networks and provides a graphical topology. In this lab, there are two networks defined, and both networks are connected by a router.

- Public:** The public network, where the controller (OpenStack) machine is running.
- Private:** The private network, where the virtual machines are deployed.
- Router1:** The router that connects both networks.

6. Click the router to see the details:



Exercise 4

Configure the connection to the IBM UrbanCode Deploy server

Settings and Configuration

© Copyright IBM Corporation 2017

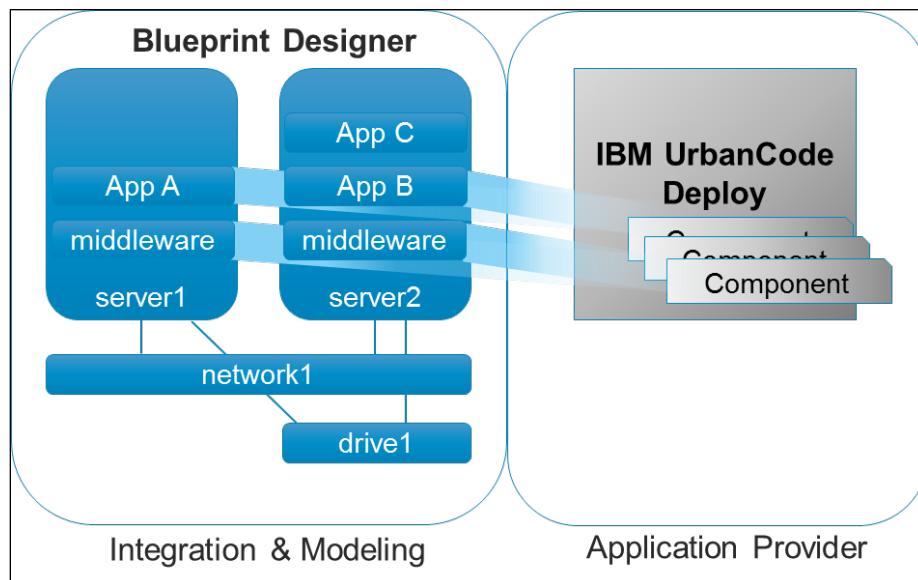
Exercise 4: Configure the connection to the IBM UrbanCode Deploy server

Exercise 4:

Configure the connection to the IBM UrbanCode Deploy server

In this exercise, you view the pattern integration between IBM UrbanCode Deploy and the blueprint design server. In most cases, you connect the design server when you install it, but you can also connect later. When you edit a blueprint on the design server, the Component drawer of the Blueprint Designer shows the components from the IBM UrbanCode Deploy server.

The following diagram illustrates the connection between the blueprint design server and the IBM UrbanCode Deploy server. It shows the resources (components) that are available to you for designing blueprints.



To do this, you will:

- View the integration for the blueprint design server in IBM UrbanCode Deploy.
- View the integration for the IBM UrbanCode Deploy server in the Blueprint Designer.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 4:

Tasks and results

Task 1. View the integration for the blueprint design server in IBM UrbanCode Deploy

1. From the browser, open a new tab and click the **UrbanCode Deploy** bookmark, or type the URL, <http://192.168.27.100:8080>, in the address field.
2. In the **User name** field, type admin; in the **Password** field, type admin.
3. From the top navigation bar, click **Settings**.
4. Under the **Automation** panel, click **Blueprint Designer Integrations**.

The screenshot shows the 'Settings' page in IBM UrbanCode Deploy. The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The 'Settings' link is underlined. Below the navigation is a breadcrumb trail: Home > Settings. The main content area is divided into three panels: Automation, Security, and System. The Automation panel contains links for Automation Plugins, Source Configuration Plugins, Running Version Imports, Locks, Blueprint Designer Integrations (which has a cursor icon over it), Post Processing Scripts, and Statuses. The Security panel contains links for API Keys And Certificates, Authentication (Users), Authorization (Groups), Teams, Tokens, Role Configuration, and Type Configuration. The System panel contains links for Logging, Network, Notification Schemes, Output Log (with a Download link), Audit Log, Diagnostics (with a Download link), Patches, Properties, CodeStation, and System Settings.

Under the Name column on the Automation page, you see an existing integration, called *landscaper*. This provides the integration information for the Blueprint Designer.

5. Under the Actions column, click **Edit**.

You can see that the URL for the Blueprint Designer is already defined.

6. Change the authentication information to a user who has permission to create blueprints in the Blueprint Designer:
 - In the **Username** field, type demo.
 - In the **Password** field, type labstack.

The IBM UrbanCode Deploy server has access to only the blueprints that this user name has access to.

7. Click **Test Connection** to ensure a successful connection.
8. Click **Close**, and click **Save**.

Edit Blueprint Designer Integration

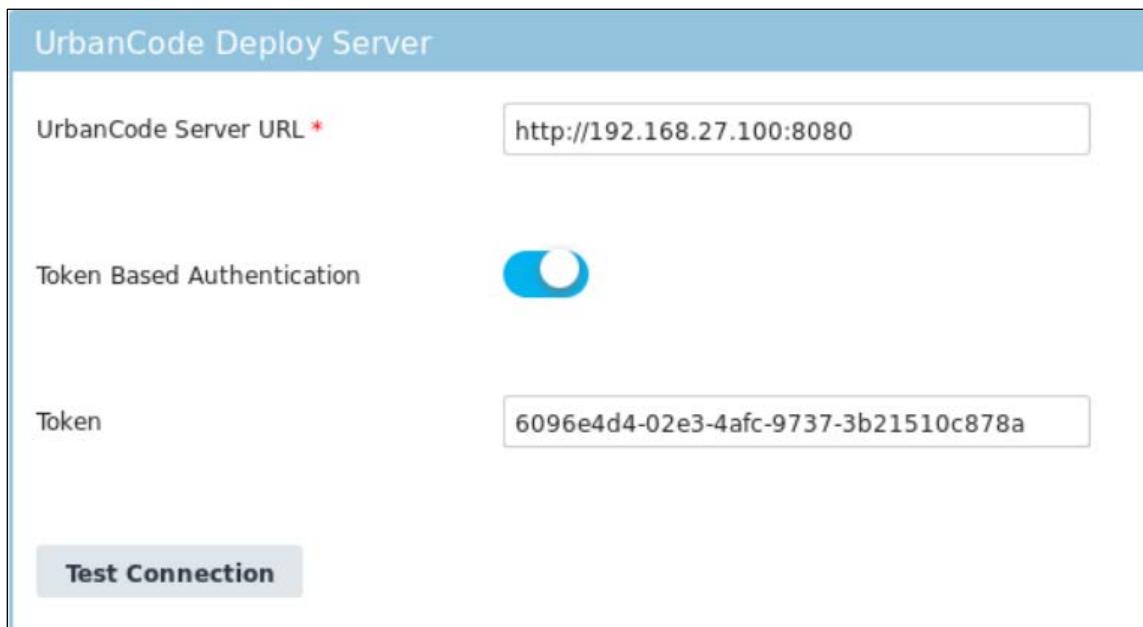
Name *	Blueprint design server
Description	
URL *	http://192.168.27.100:9080/landscaper
Response *	<input type="radio"/> Automatically authenticate for each UrbanCode Deploy user. <input checked="" type="radio"/> Always authenticate with one blueprint designer user name.
Ensure that blueprint designer user name that you enter has permissions to access the cloud platforms where you provision cloud environments. The default administrative account for the blueprint design server, ucdpadmin, might not have access to the required resources on the target cloud platforms.	
Username *	demo
Password *	*****

Save **Cancel** **Test Connection**

Task 2. View the integration for the IBM UrbanCode Deploy server in the Blueprint Designer.

1. Switch to the **Blueprint Designer** tab, log out as the demo user, and log in as the admin user with the user name of `ucdpadmin` and the password of `ucdpadmin`.
2. At the upper-right corner of the page, click the **Settings** icon , and click **System Settings** from the drop-down list.

In the UrbanCode Deploy Server section, observe the predefined settings for the IBM UrbanCode Deploy server.



The screenshot shows the 'UrbanCode Deploy Server' configuration page. It includes fields for 'UrbanCode Server URL' (set to `http://192.168.27.100:8080`), 'Token Based Authentication' (switched on), and a 'Token' field containing the value `6096e4d4-02e3-4afc-9737-3b21510c878a`. A 'Test Connection' button is also visible.

Notice that the **Token** field has a long string of letters and numbers. When a user edits a blueprint, that user can see the components that are associated with this token.

You generate a token in the IBM UrbanCode Deploy by clicking **Settings > Tokens**, and clicking **Create Token**. You copy the token into the **Token** field of the Blueprint Designer.

The screenshot shows the 'Tokens' tab selected in the navigation bar of the IBM UrbanCode Deploy interface. The page displays a table of tokens with columns: Token, User, Expiration Date, Description, Allowed IPs, and Actions. There are seven records listed, each with a 'Delete' link in the Actions column. At the bottom left, there is a message '7 records - Refresh Print'. At the bottom right, there is a 'Rows' dropdown set to '10'.

Token	User	Expiration Date	Description	Allowed IPs	Actions
63ee8ed5-a8ce-4adb-ade9-45d40fcc8eb1	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete
cbd9436e-8c87-4bad-926f-fe0ba8ee1a58	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete
4c55082a-6beb-4dcc-a9e4-db41517f94ee	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete
03d0cada-f493-44f7-bad2-a45d03c5d80a	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete
19696f71-82cc-4d69-a860-de4eb5934010	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete
d474e3c5-fa4e-4a66-97ee-45f64f071cb7	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete
6096e4d4-02e3-4afc-9737-3b21510c878a	admin	12/31/2020, 12:00 PM		0.0.0.0/0	Delete

3. Click **Test Connection**.
4. Click **Close**, and then click **Save**.

In this unit, you viewed the integrations between the blueprint design server with both the OpenStack cloud and the IBM UrbanCode Deploy server. The integration with OpenStack provides resources that you can use to create a blueprint that describes an environment, such as images, networks, and storage volumes. This blueprint specifies the type of system to be deployed and the network for the system. The integration with the IBM UrbanCode Deploy server enables the blueprint to specify the application components to be installed.

In the next unit, you create a blueprint with the OpenStack resources. Then, you provision the blueprint and deploy the infrastructure.

Unit 3 Blueprints and Cloud Infrastructure

IBM Training



Blueprints and Cloud Infrastructure

**IBM UrbanCode Deploy Blueprint
Designer - v6.2.1.1**

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Explore the options of the Blueprint Designer that relate to images, networks, and storage
- Show how changes in the graphical editor manifest in the Heat Orchestration source code
- Create and configure a simple blueprint that includes the OpenStack artifacts
- Deploy a blueprint to a cloud environment and view the changes in OpenStack

Unit objectives

The IBM UrbanCode Deploy Blueprint Designer helps you to extend your cloud and virtualization technology and make that part of your complete DevOps delivery pipeline.

The screenshot shows the Blueprint Designer user interface with three main sections:

- Home:** Shows a list of blueprints and environments.
- Environments:** Shows a list of provisioned environments with their status and latest event.
- Repositories:** Shows a Git repository interface with a commit history and staged changes.

Three callout boxes point to specific features:

- Author and edit blueprints:** Points to the Home section.
- Check status of provisioned environments:** Points to the Environments section.
- Store blueprints and configuration files:** Points to the Repositories section.

Blueprints and Cloud Infrastructure © Copyright IBM Corporation 2017

Blueprint Designer user interface

From the Blueprint Designer Home page, you can open the blueprints and environments. You can also open the individual sections such as blueprints, environments, and repositories for more tasks:

- **Blueprints:** You create or edit blueprints in the editor. After you save a blueprint, you can provision it, where you name the environment and identify the properties required for the target cloud provider.
- **Environments:** After you author and provision the blueprints, you can view and manage the provisioned environments. The Environments page lists each provisioned environment, its status, and latest event (confirming success or failure of a provisioning action).
- **Repositories:** You store blueprints and configuration files that you create with the Blueprint Designer in Git repositories. By storing blueprints and configurations in the Git repository for a team, you can share the files with members of that team.

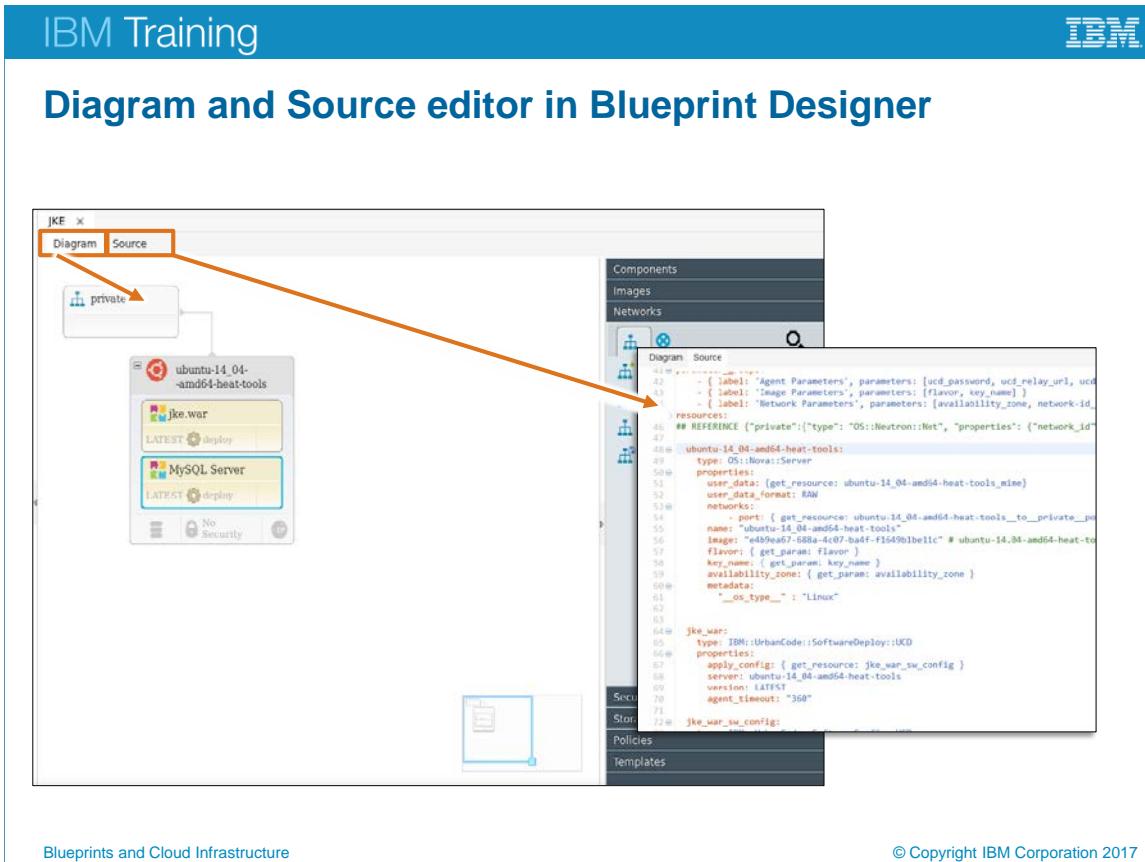


Diagram and Source editor in Blueprint Designer

The Blueprint Designer has a graphical Diagram editor where you can drag and drop blueprint components. You can directly edit blueprint document text in the source editor, which has a YAML-based syntax. Both editors have a Properties pane, where you can set input and output parameters, and include facilities to validate and test the created blueprints.

IBM Training

IBM

A graphical depiction of the modeled template

The screenshot shows the IBM UrbanCode Deploy Blueprint Designer application. On the left is a properties panel with fields like Agent Timeout, Component Process, and Version. The main area is a canvas with a network diagram showing nodes like 'private' and 'ubuntu-14-04-vm001 heat tools'. A yellow callout box says 'Drag resources to the canvas'. To the right is a palette divided into sections: Components, Images, Networks, Security, Storage, Policies, and Templates. A yellow callout box says 'Palette contains both cloud and IBM UrbanCode Deploy resources'. An orange bracket groups the 'Components', 'Images', and 'Networks' sections of the palette.

Drag resources to the canvas

Palette contains both cloud and IBM UrbanCode Deploy resources

Blueprints and Cloud Infrastructure

© Copyright IBM Corporation 2017

A graphical depiction of the modeled template

The palette at the right side of the editor shows resources that you can add to the blueprint. The palette is filtered based on the cloud system that you are connected to, and the palette is divided into multiple drawers:

- Components
- Images
- Networks
- Security
- Storage
- Policies
- Templates

In the Diagram editor, you can model the cloud landscape by dragging and dropping the various resources that are available in the editor palette, which translates the visual representation into the OpenStack HOT format.

A blueprint is a text document (HOT)

Palette works seamlessly for both editors

Blueprints and Cloud Infrastructure

© Copyright IBM Corporation 2017

Source code editor

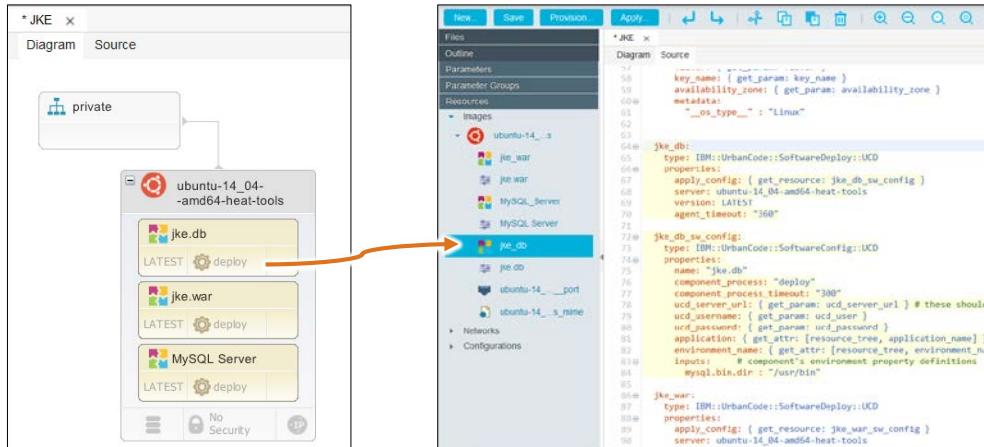
In the Source editor, you can click and drag resources from the palette just like when you edit the diagram. Be sure to drag your resources into the correct section of the source file.

The left side of the editor shows the following sections when the Source editor is open:

- **Outline:** Shows the structure of the blueprint.
- **Parameters:** Shows the parameters in the blueprint. Click a parameter to go to its definition, or drag it to the blueprint.
- **Parameter Groups:** Shows the parameter groups in the blueprint.
- **Resources:** Shows the resources that are in the template, such as virtual images.
- **Outputs:** Shows the output parameters, usually attributes, for the blueprint. The attributes in this section usually show relevant derived values, such as the URL of virtual instances.

Preserving object selection and state

- The editors are synchronized and preserve object selection and state



GUI

HOT source code

Preserving object selection and state

The editors are synchronized and preserve object selection and state.

The Source editor includes code validation

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. The left sidebar lists parameters: availability_zone, flavor, key_name, network_id_for_private, subnet_id_4bd4bcf89dc1, ucd_password, ucd_relay_url, ucd_server_url, and ucd_user. The main area displays a JSON-like configuration file with several error messages in the status bar:

```

1: Duplicate key: type
140: TODO remove this if using HEAT version Icehouse!
  
```

The configuration file contains the following code:

```

1: {
2:   "availability_zone": {
3:     "type": "string",
4:     "description": "Flavor to be used for compute instance"
5:   },
6:   "key_name": {
7:     "type": "string",
8:     "description": "For most clouds, the name of the key-pair to be used for the c"
9:   },
10:  "availability_zone": {
11:    "type": "string",
12:    "description": "Name of availability zone in which to create the instance"
13:  },
14:  "network_id_for_private": {
15:    "type": "string",
16:    "description": "Generated to reference 'private' network."
17:  },
18:  "subnet_id_ccde0dff-635f-49c5-b26b-4bd4bef89dc1": {
19:    "type": "string",
20:    "description": "Generated to reference subnet ''ccde0dff-635f-49c5-b26b-4bd4be"
21:  },
22:  "ucd_server_url": {
23:    "type": "string",
24:    "description": "The server URL for agent communication to UrbanCode Deploy. Do"
25:  },
26:  "ucd_user": {
27:    "type": "string",
28:    "description": "The user name for agent communication to UrbanCode Deploy. Do"
  
```

Blueprints and Cloud Infrastructure

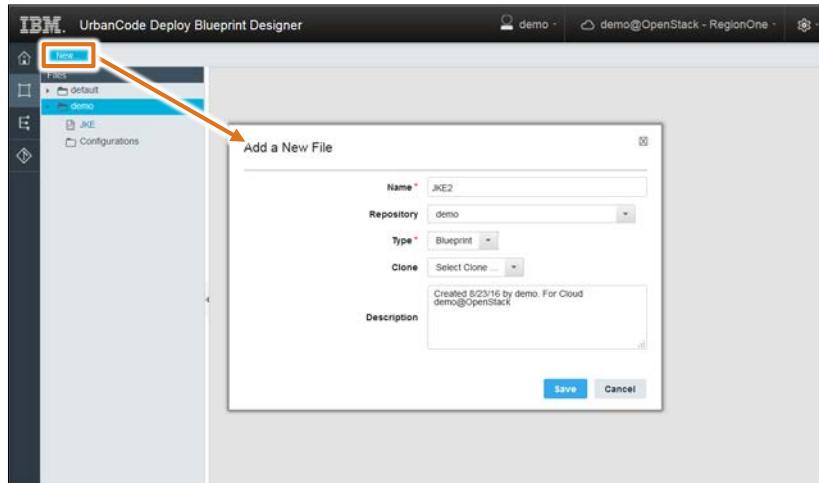
© Copyright IBM Corporation 2017

The Source editor includes code validation

If the blueprint contains an error, warning, or unfinished TODO item, you cannot provision from it. If at least one of these items is present in a blueprint, an icon that represents the most serious type of item is displayed to the right of **Toggle Comment**. For example, if the blueprint contains at least one error, the error icon is displayed. Hover over the icon to display a list of all items that you must correct or complete before you provision the blueprint.

Blueprints

- Blueprints store information that is used in provisioning cloud environments



Blueprints

The blueprint is meant to be cloud agnostic and not specifically designed for any one cloud environment. Properties that are specific to a cloud provider are kept in a configuration file. You author configuration files in the same Source editor where the blueprint is edited. The Configuration file defines the cloud-specific properties that qualify a blueprint when provisioned into a particular cloud.

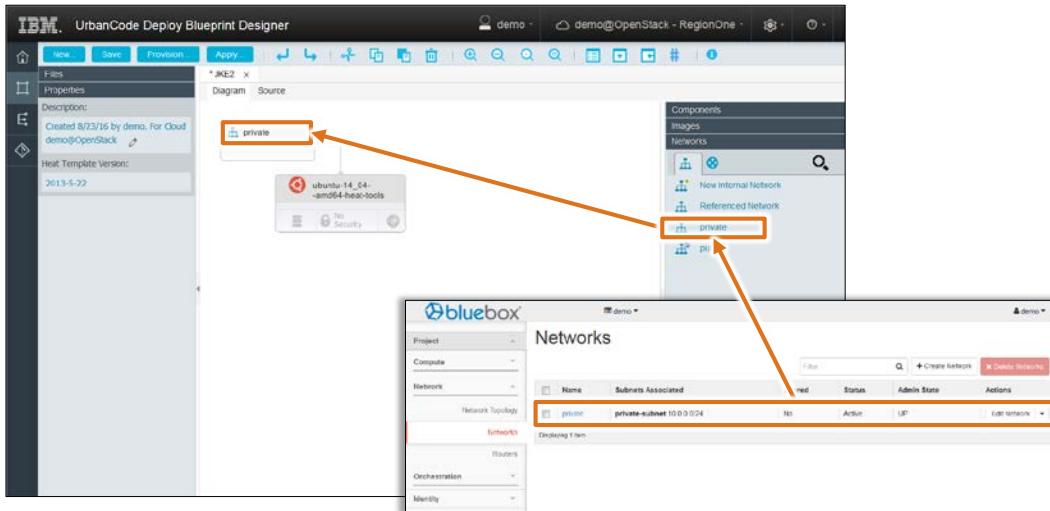
The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, the 'Blueprints and Cloud Infrastructure' sidebar includes sections for 'Key', 'Image', 'Availability Zone', 'Flavor', and 'Key Name'. The main workspace displays a 'Components' drawer with an 'Images' section containing a list of available images. Two specific images are highlighted with orange boxes: 'ubuntu-14_04-amd64-heat-tools' and 'ubuntu-14_04-amd64-heat-tools'. Arrows point from these highlighted images in the Components drawer to their corresponding entries in the 'Images' table below. The 'Images' table lists several images with columns for 'Image Name', 'Type', 'Status', 'Protected', 'Format', 'Size', and 'Actions'. The table shows entries like 'centos-6.5-amd64', 'ubuntu-14_04', and 'ubuntu-14_04-amd64-heat-tools'.

The Images drawer

The Images drawer shows virtual images that are available on the current cloud system. You can drag these images to the blueprint.

Note: Before you can use images on an OpenStack or OpenStack-based cloud with the Blueprint Designer, you must configure cloud-init on those images. Cloud-init is a set of scripts and utilities that cloud systems use to initialize and configure instances.

The Networks drawer



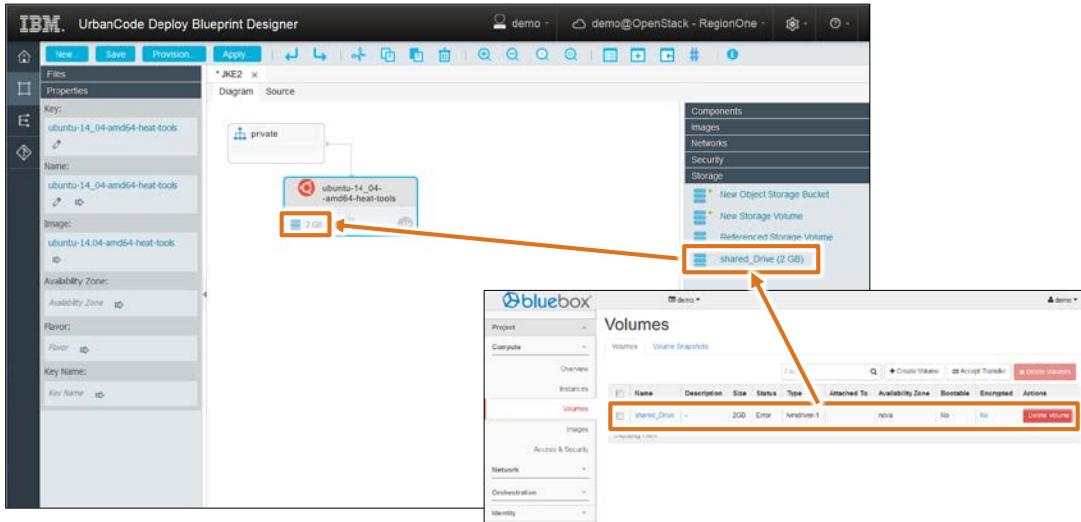
Blueprints and Cloud Infrastructure

© Copyright IBM Corporation 2017

The Networks drawer

If the connected cloud system supports routers, the Networks drawer also shows routers that are available. The New Internal Network resource represents a new network.

The Storage drawer



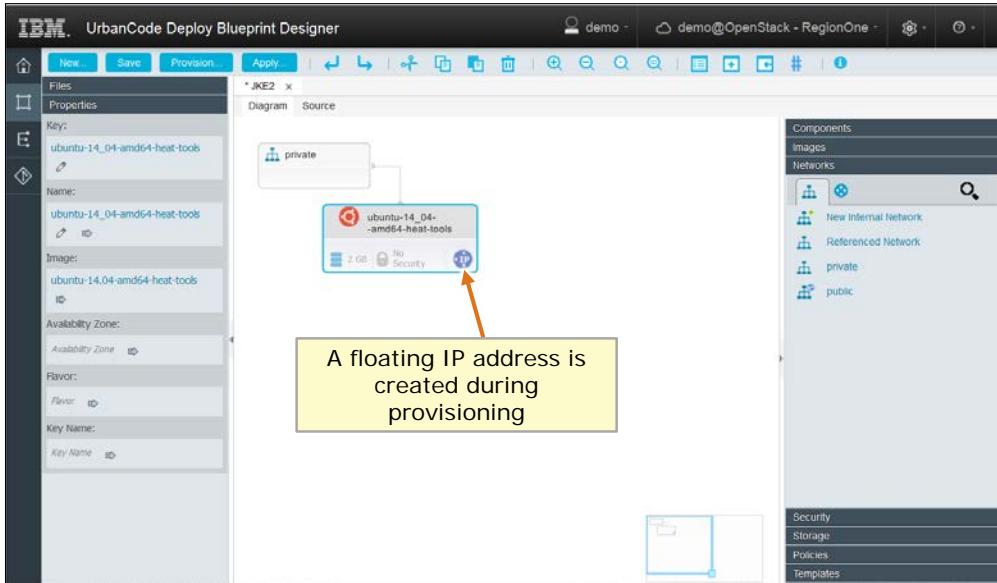
Blueprints and Cloud Infrastructure

© Copyright IBM Corporation 2017

The Storage drawer

To add a storage volume, drag a volume to a virtual image. The New Storage Volume resource represents a new storage volume.

Access network and resources



Access network and resources

The images can access the network and the resources that are on the network. For example, many virtual networks have an internal network and an external network. The external network provides public IP addresses to images that require IP addresses. In this case, you often have to connect a virtual image to an internal network and then retrieve a floating IP address from the external network. When you provision an environment from the blueprint, the images are connected to the networks.

The screenshot shows the UrbanCode Deploy Blueprint Designer application. On the left, there's a sidebar with options like 'New', 'Save', 'Properties', and 'Provision'. A red arrow points from the 'Provision' button in the top navigation bar down to the 'Provision Blueprint to new Environment' dialog box. The dialog box contains fields for 'Environment Name' (set to 'UAT'), 'Cloud Project' (set to 'demo@OpenStack'), and 'Configuration' (set to 'Select Configuration...'). It also includes sections for 'Agent Parameters', 'Image Parameters', 'Flavor' (set to 'm1.medium'), 'Key Name' (set to 'demo_key'), and 'Network Parameters'. At the bottom right of the dialog is a blue 'Provision' button.

Provisioning of environments

After a blueprint is created, you can provision a cloud environment and set aside resources on the specified cloud that you specified through a blueprint. The provisioning step involves a blueprint, naming the environment to be created, and, optionally, a configuration file. The blueprint is provisioned into an environment, creating the virtual compute resources and attaching them to a network.

Blueprint resources have attributes and properties. Attributes are values that describe the physical environment resources and are often available only after stack provisioning. Properties are input values that are specified at or before provisioning time, such as the size of a storage volume. The provisioning window also includes property fields whose values are filled automatically with the contents of an (optionally selected) configuration file. You can overwrite those default provisioning values.

When you provision environments through OpenStack Heat, you create Heat-based blueprints that identify the infrastructure of a cloud environment, including its virtual images, network, and storage.

Creation of the virtual environment

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. A table lists resources for a stack named 'UAT' under 'Applied Blueprint' JKE2. The table columns include Environment, Applied Blueprint, Cloud Project, Region, Last Update, Virtual Instances, Last Orchestration Event, and Actions. One row is highlighted with an orange border, showing a resource named 'ubuntu-14_04-amd64-heat-tools_to_private_port'. The 'Type' column shows 'OS:Neutron:Port'. The 'Details' column contains the configuration: name = ubuntu-14_04-amd64-heat-tools, image = ubuntu-14_04-amd64-heat-tools, power_type = PowerCLI, Active_ip_address = 10.0.0.12,172.24.4.9. The 'Status' column shows 'ACTIVE' with a green checkmark icon. The 'Stack Status' column shows 'Create Complete' with a green checkmark icon. The 'Last Orchestration Event' column shows 'Stack CREATE completed successfully' with a green checkmark icon.

Creation of the virtual environment

These virtual machines then have an IBM UrbanCode Deploy agent installed on them, and that agent is connected back to IBM UrbanCode Deploy. At the same time, all of the environment resources required to manage this virtual environment are created, including the resource tree, the agents, and the component placement rules required from an ongoing management perspective.

Environments

- The virtual resources for the environment are listed in the Environments tab

UrbanCode Deploy Blueprint Designer

Environments > UAT

Resources Instances

Volumes Network Component Outputs Links

Instances

Instance Name	IP Address	Size	Keypair	Status	Power State	Actions
ubuntu-14_04-amd64-heat-tools	10.0.0.12, 172.24.4.9	m1.medium	demo_key	ACTIVE	ACTIVE	

1 record - Refresh Print

IP address assigned to the system

Blueprints and Cloud Infrastructure

© Copyright IBM Corporation 2017

Environments

You can see these virtual images, which map to running virtual hosts in your OpenStack-based cloud.

OpenStack shows the deployed instance

The screenshot shows the bluebox cloud management interface. On the left, there's a sidebar with Project, Compute, Network, Orchestration, and Identity sections. Under Compute, the Instances section is selected, showing a table with columns: Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions. One row is selected, highlighting the IP address '172.24.4.9'. An orange arrow points from this highlighted IP address to a Command Prompt window titled 'Administrator: Command Prompt' running on Microsoft Windows. The window displays the output of a ping command to the IP address 172.24.4.9, showing four successful replies with round-trip times between 132ms and 207ms.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
ubuntu-14.04-amd64-heat-tools	ubuntu-14.04-amd64-heat-tools	10.0.0.12 Floating IP:	m1.medium	demo_key	Active					

```

Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\IBM_ADMIN>ping 172.24.4.9

Pinging 172.24.4.9 with 32 bytes of data:
Reply from 172.24.4.9: bytes=32 time=207ms TTL=63
Reply from 172.24.4.9: bytes=32 time=151ms TTL=63
Reply from 172.24.4.9: bytes=32 time=151ms TTL=63
Reply from 172.24.4.9: bytes=32 time=167ms TTL=63

Ping statistics for 172.24.4.9:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 132ms, Maximum = 207ms, Average = 132ms

C:\Users\IBM_ADMIN>

```

Blueprints and Cloud Infrastructure

© Copyright IBM Corporation 2017

OpenStack shows the deployed instance

The IP address is assigned to the public network, which is the same as the one listed in the Blueprint Designer. When the stack finishes provisioning, verify that the server is externally accessible by opening a command prompt and pinging the floating IP to check for a response.

The screenshot shows the IBM UrbanCode Deploy application interface. At the top, there's a blue header bar with the text "IBM Training" on the left and the "IBM" logo on the right. Below the header is a main content area with a title "IBM UrbanCode Deploy shows no activity". The main content area has a dark header bar with various navigation links: Dashboard, Components, Applications, Configuration (which is selected), Processes, Resources, Calendar, Work Items, Reports, and Settings. Underneath this is a sub-header with "Home > Applications > JKE" and the title "Application: JKE". Below the sub-header, there are sections for "Created By" (admin) and "Created On" (7/29/2016, 3:13 PM). To the right of these, there's a "Description:" field containing "Sample JKE Banking application". Below this, there's a navigation bar with tabs: Environments (selected), History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. A "Create Environment" button is located in the top-left corner of the main content area. Below the tabs, there are search fields for "Search by Name" and "Search by Blueprint", along with "Expand All" and "Collapse All" buttons. The main content area displays the message "No Environments Found!". At the bottom of the page, there's a footer bar with the text "Blueprints and Cloud Infrastructure" on the left and "© Copyright IBM Corporation 2017" on the right.

IBM UrbanCode Deploy shows no activity

The blueprint in this module specifies the type of system to be deployed, the network for the system, and the requirement of attaching to a public network. Because the IBM UrbanCode Deploy components were not added to the blueprint, an environment was not created in IBM UrbanCode Deploy.

Unit summary

- Explore the options of the Blueprint Designer that relate to images, networks, and storage
- Show how changes in the graphical editor manifest in the Heat Orchestration source code
- Create and configure a simple blueprint that includes the OpenStack artifacts
- Deploy a blueprint to a cloud environment and view the changes in OpenStack

Unit summary

The Diagram editor is graphical; the Source editor contains the HOT source code. The editors are synchronized and preserve object selection and state. Blueprint properties store information that is used in provisioning cloud environments.

You model the cloud landscape by dragging and dropping the various components that are available in the palette. When you provision a blueprint into an environment, virtual compute resources are created and attached to a network.

Exercises: Modeling a heat stack and provisioning a cloud infrastructure

- Create a blueprint with OpenStack resources
- Provision a blueprint to a cloud environment
- View the provisioned resources from the OpenStack dashboard

Exercises: Modeling a heat stack and provisioning a cloud infrastructure

Exercise 1

Create a simple blueprint

Exercise 1: Create a simple blueprint

Exercise 1: Create a simple blueprint

Now that the blueprint design server is configured to communicate with OpenStack, it can access the OpenStack resources. In this exercise, you use the Blueprint Designer to build a cloud infrastructure (Heat stack), which consists of the OpenStack resources that you analyzed in the previous lab.

In this exercise, you will:

- Create a simple blueprint.
- Add virtual images to the blueprint.
- Add a private network to the blueprint.

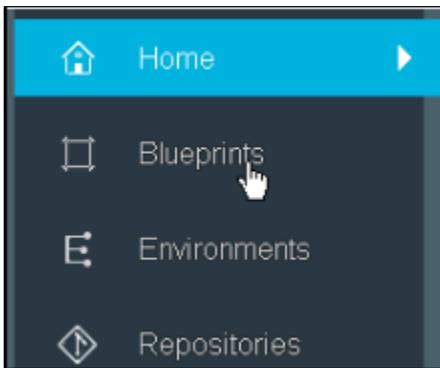
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 1:

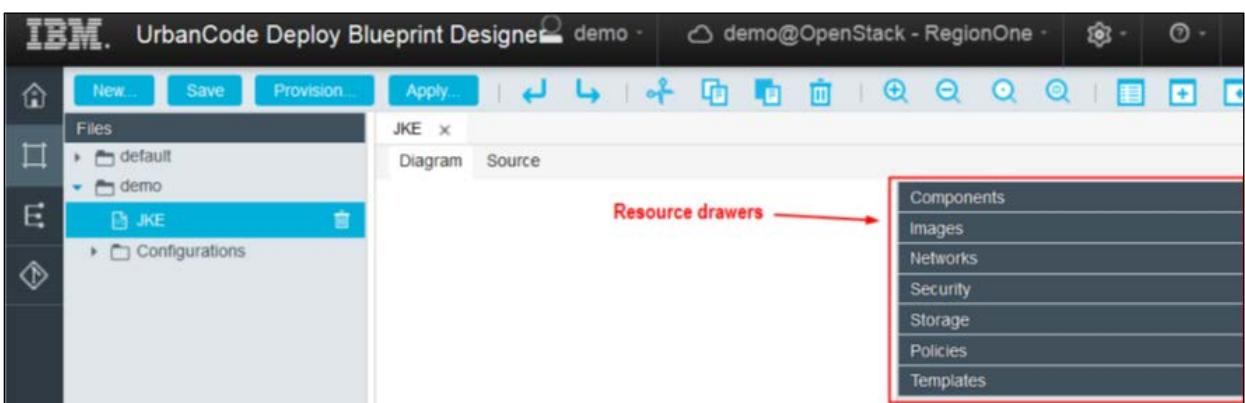
Tasks and results

Task 1. Create a blueprint.

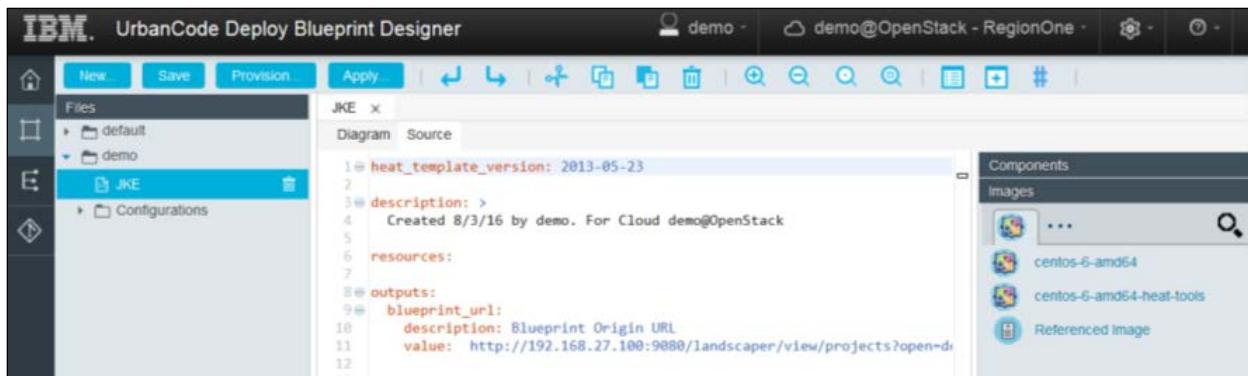
1. From the browser, switch to the **Blueprint Designer** tab.
2. Verify that you are logged in as the **demo** user with the user name of **demo** and a password of **labstack**.
3. From the main menu, click **Blueprints**.



4. In the Files panel, verify that the **demo** folder is selected, and click **New**.
5. Specify the following information for the new blueprint (Heat stack):
 - In the **Name** field, type **JKE**.
 - In the **Repository** field, select **demo**.
 - In the **Type** field, select **Blueprint**.
6. Click **Save**.



7. Next to the **Diagram** tab, click the **Source** tab. An empty skeleton of a Heat Orchestration Template (HOT) is displayed. As you add resources to the canvas, the Blueprint Designer adds the code.
- Note: A Heat Orchestration Template (HOT) is a particular template format that is native to Heat. Heat Orchestration Templates are expressed in YAML. YAML is a simple, text/human-readable annotation format that can be used to store data.



```

heat_template_version: 2013-05-23
description: >
    Created 8/3/16 by demo. For Cloud demo@OpenStack
resources:
outputs:
    blueprint_url:
        description: Blueprint Origin URL
        value: http://192.168.27.100:9080/landscaper/view/projects?open=di

```

The screenshot shows the Blueprint Designer interface. The left sidebar has 'Files' expanded, showing 'default', 'demo' (which is selected), and 'Configurations'. The main area shows the 'Source' tab with the above YAML code. To the right is a 'Components' drawer with 'Images' tab selected, showing three items: 'centos-6-amd64', 'centos-6-amd64-heat-tools', and 'Referenced Image'. There is also a search icon in the drawer.

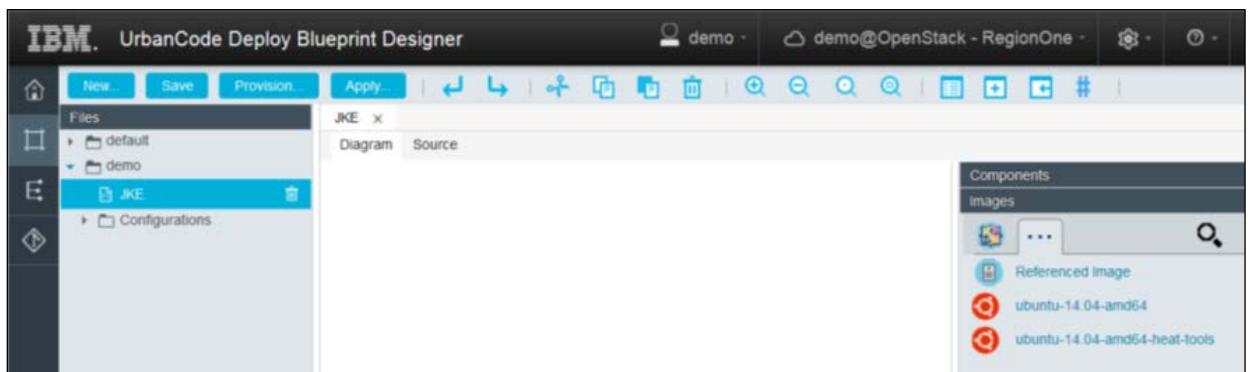
8. Switch back to the **Diagram** tab.

Task 2. Add virtual images to the blueprint.

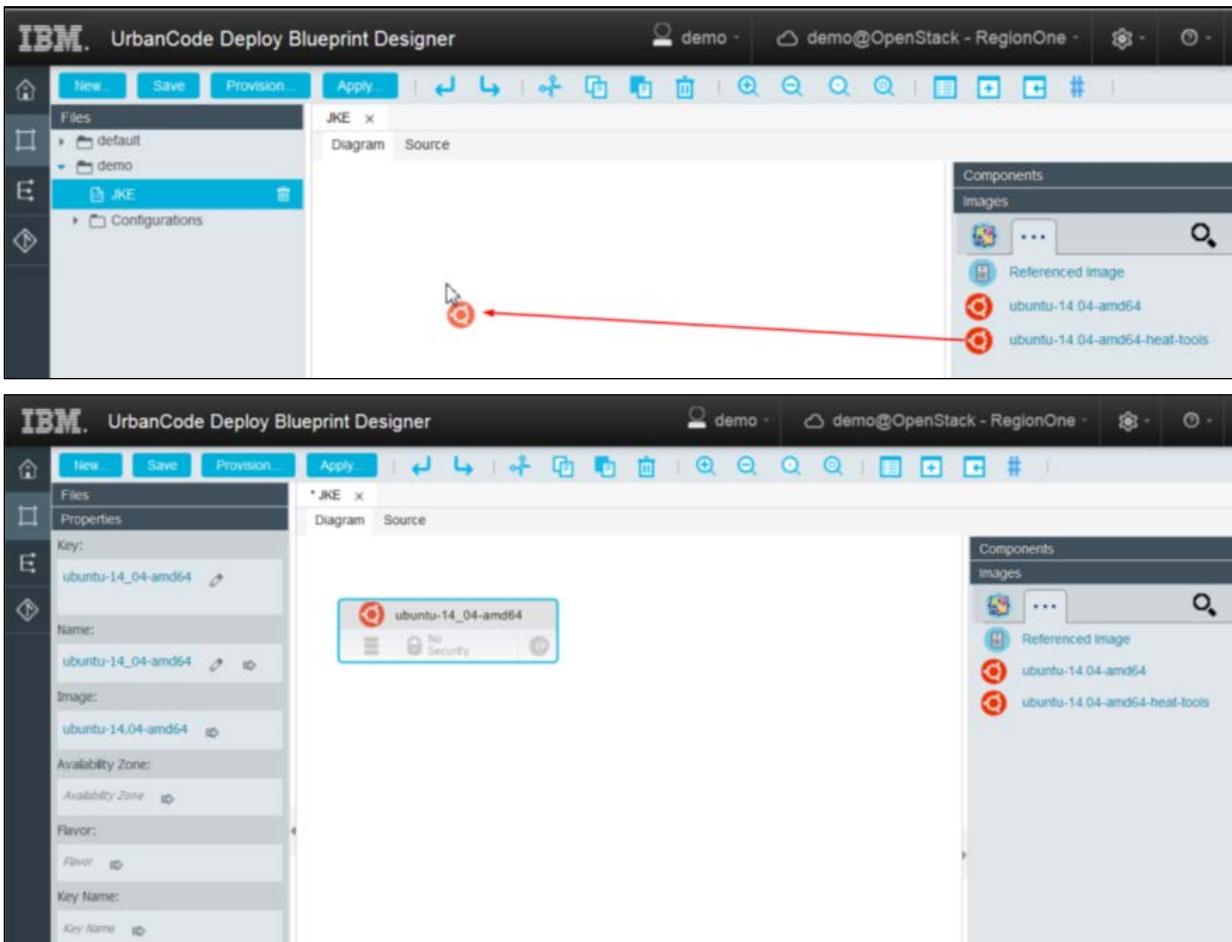
The Images drawer shows the virtual images that are available on the current cloud system, which is the OpenStack cloud that you connected to and explored in the previous lesson.

1. In the palette to the right, click the **Images** drawer. You can see that the two Centos images are the same ones that you saw in the OpenStack Images section.
2. Click the second tab with the ellipses (...). You can see the two Ubuntu images that you viewed in the OpenStack Images section in the previous lab.

The Images drawer also includes a resource that is named **Referenced Image**. This resource represents a generic virtual image. You can use the referenced image to represent a virtual image that you specify later.

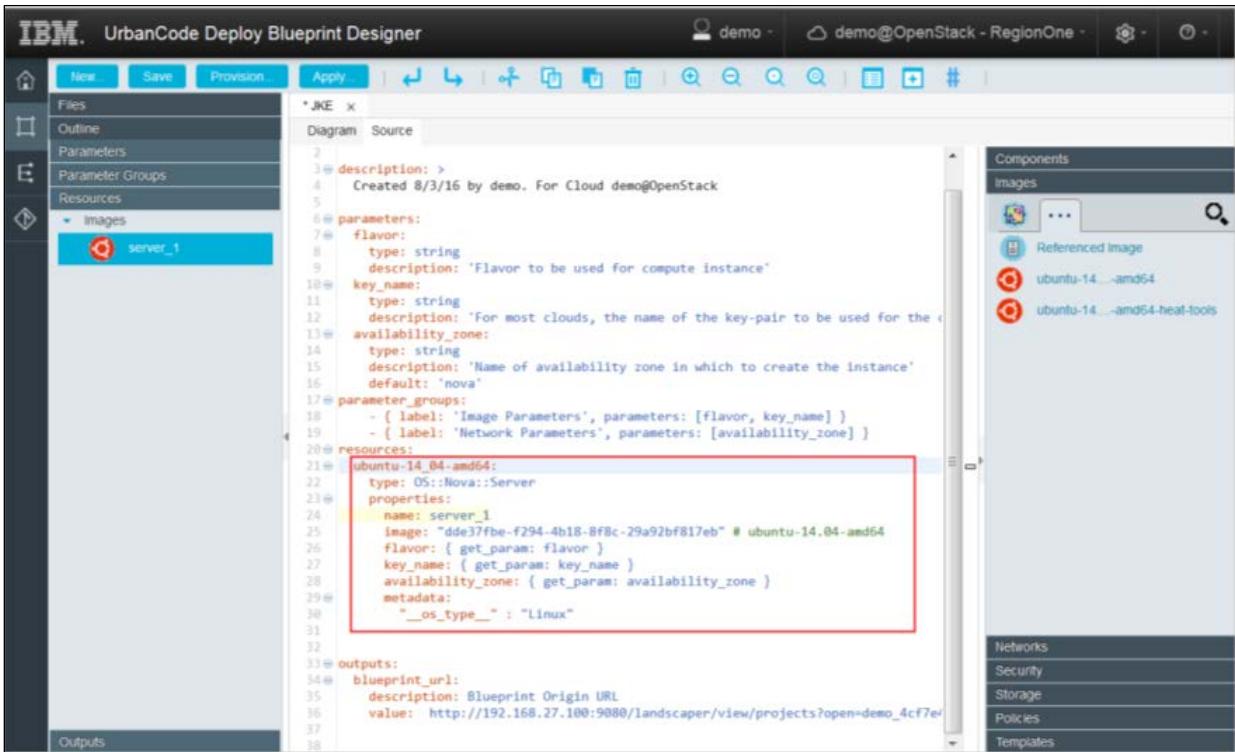


3. Drag the **ubuntu-14.04-amd64-heat-tools** image to the canvas.



4. In the **Name** field, click the **Edit** icon , type `server_1`, and press **Enter**.

5. Click the **Source** tab to observe the changes to the Heat template.
- Notice that there is a resource for the server (OS::Nova::Server), which has the name of server_1.



```

 1
 2
 3④ description: >
 4   Created 8/3/16 by demo. For Cloud demo@OpenStack
 5
 6④ parameters:
 7④   flavor:
 8     type: string
 9     description: 'Flavor to be used for compute instance'
10④   key_name:
11     type: string
12     description: 'For most clouds, the name of the key-pair to be used for the i
13④   availability_zone:
14     type: string
15     description: 'Name of availability zone in which to create the instance'
16     default: 'nova'
17④   parameter_groups:
18     - { label: 'Image Parameters', parameters: [flavor, key_name] }
19     - { label: 'Network Parameters', parameters: [availability_zone] }
20④ resources:
21④   ubuntu-14_04_amd64:
22     type: OS::Nova::Server
23     properties:
24       name: server_1
25       image: "dde37fbe-f294-4b18-8f8c-29a92bf817eb" # ubuntu-14.04-amd64
26       flavor: { get_param: flavor }
27       key_name: { get_param: key_name }
28       availability_zone: { get_param: availability_zone }
29       metadata:
30         __os_type__ : "Linux"
31
32
33④ outputs:
34④   blueprint_url:
35     description: Blueprint Origin URL
36     value: http://192.168.27.100:9080/landscaper/view/projects?open=demo_4cf7e1
37
38

```

There are also parameters to configure this resource. Those parameters will be queried from the deployment requester.

Note: Virtual machine resources (type OS::Nova::Server) include information about the virtual image for the machine and the networks to which the virtual machine is connected.

6. Click the **Diagram** tab.

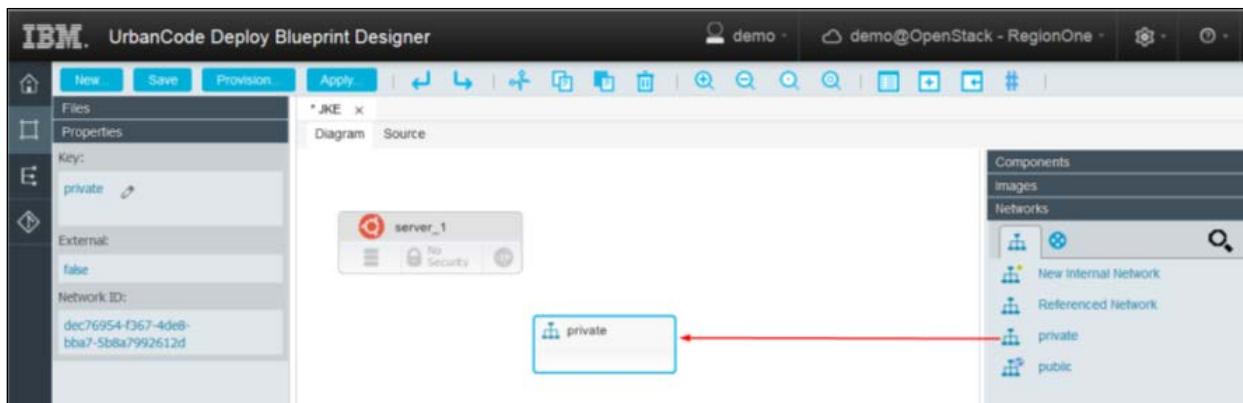
Notice the name of your file has an asterisk (*) next to it, which indicates that you have unsaved work in the file.

7. Click **Save**.

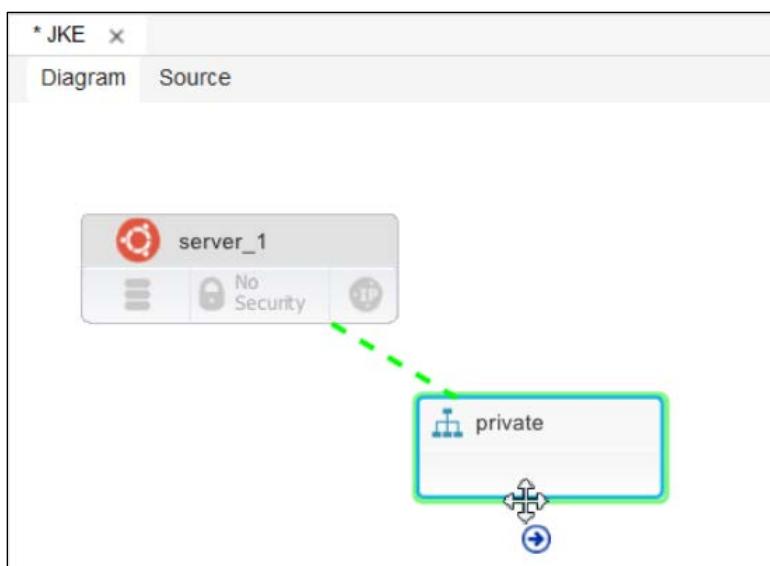
Task 3. Add a private network to the blueprint.

The Network drawer shows networks that are available on the OpenStack cloud.

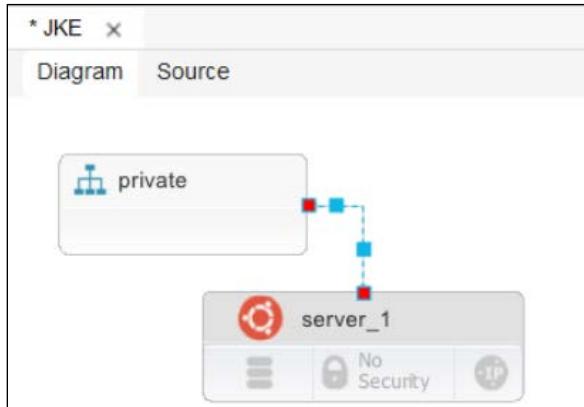
1. Click the **Network** drawer, which is located under the Images drawer.
The New Internal Network resource represents a new network. The Referenced Network resource represents a network for which you provide the information later.
2. Drag the **private** network to the canvas.



3. To connect server_1 to the private network, hover over either component until a small arrow is displayed. Click and drag this arrow to the other component to connect them.



After the connection is complete, a line connects the two resources. This compute resource will be the host for the MySQL database and the WebSphere Liberty Profile web server components (from IBM UrbanCode Deploy) in subsequent labs.



When you add a network to a blueprint, the editor adds a resource of the type OS::Neutron::Port, which represents a port on the specified network.

4. Switch to the **Source** tab to see this addition to the template:

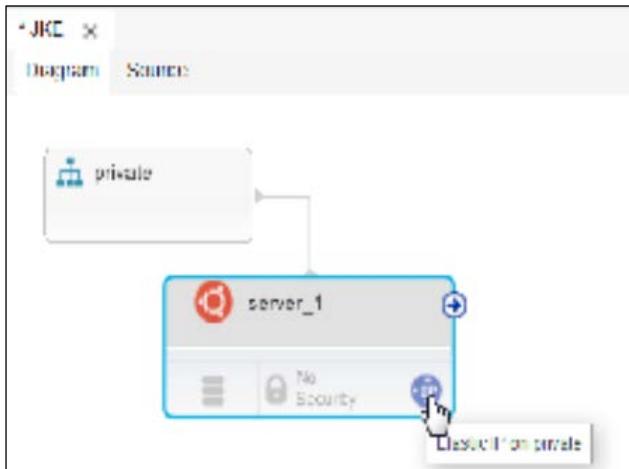
```
* JKE X
Diagram Source
22 type: string
23 description: 'Generated to reference subnet ''ccde0dff-635f-49c5-b26b-4bd4b
24 default: 'ccde0dff-635f-49c5-b26b-4bd4bef89dc1'
25 parameter_groups:
26 - { label: 'Image Parameters', parameters: [flavor, key_name] }
27 - { label: 'Network Parameters', parameters: [availability_zone, network-id]
28 resources:
29 ## REFERENCE {"private":{"type": "OS::Neutron::Net", "properties": {"network_id"
30
31 ubuntu-14_04-amd64:
32     type: OS::Nova::Server
33 properties:
34 networks:
35     - port: { get_resource: ubuntu-14_04-amd64_to_private_port }
36     name: server_1
37     image: "dde37fbe-f294-4b18-8f8c-29a92bf817eb" # ubuntu-14.04-amd64
38     flavor: { get_param: flavor }
39     key_name: { get_param: key_name }
40     availability_zone: { get_param: availability_zone }
41 metadata:
42     "__os_type__" : "Linux"
43
44
45 ubuntu-14_04-amd64_to_private_port:
46     type: OS::Neutron::Port
47 properties:
48     network_id: { get_param: network_id_for_private }
49     replacement_policy: AUTO #TODO remove this if using HEAT version Icehouse
50     fixed_ips:
51         - subnet: { get_param: subnet_id_ccde0dff-635f-49c5-b26b-4bd4bef89dc1 }
52
53 outputs:
54 blueprint_url:
55     description: Blueprint Origin URL
56     value: http://192.168.27.100:9080/landscaper/view/projects?open=demo_4cf7e
```

5. Assigning a floating IP to the private network

An OpenStack environment might have two networks: an internal network, with IP addresses for use within the network, and an external network, with externally accessible floating IP addresses. Each server resource has an **IP**

 icon for assigning a floating IP address. Earlier in this exercise, you connected the compute resource (server) to the *private* network, which means it is on a network, but it is not accessible outside of the OpenStack server. Now, you instruct OpenStack to assign a floating IP address to the server that can be externally accessed.

6. Click the **IP** icon  in the lower-right corner of the `server_1` image. The icon color turns blue.



7. Click the **Source** tab. Notice the addition of the floating IP address.

```

35@  ubuntu-14_04-amd64:
36    type: OS::Nova::Server
37    properties:
38      networks:
39        - port: { get_resource: ubuntu-14_04-amd64_to_private_port }
40    name: server_1
41    image: "dde37fbe-f294-4b18-8f8c-29a92bf817eb" # ubuntu-14.04-amd64
42    flavor: { get_param: flavor }
43    key_name: { get_param: key_name }
44    availability_zone: { get_param: availability_zone }
45    metadata:
46      "__os_type__" : "Linux"
47
48
49@  ubuntu-14_04-amd64_to_private_port_floating_ip:
50    type: OS::Neutron::FloatingIP
51    properties:
52      floating_network_id: { get_param: public_network_id }
53      port_id: { get_resource: ubuntu-14_04-amd64_to_private_port }
54

```

8. Click **Save**.

Exercise 2

Provision a blueprint to a cloud environment

Exercise 2: Provision a blueprint to a cloud environment

Exercise 2: Provision a blueprint to a cloud environment

You have focused on the connection between the blueprint design server and OpenStack. This allowed you to use the resources from OpenStack in a blueprint to build the Heat stack infrastructure. Although you have not yet added the components from IBM UrbanCode Deploy, you can provision an environment to the cloud to see how the changes that you made surface in the OpenStack dashboard.

In this exercise, you will:

- Provision the blueprint to a cloud environment.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

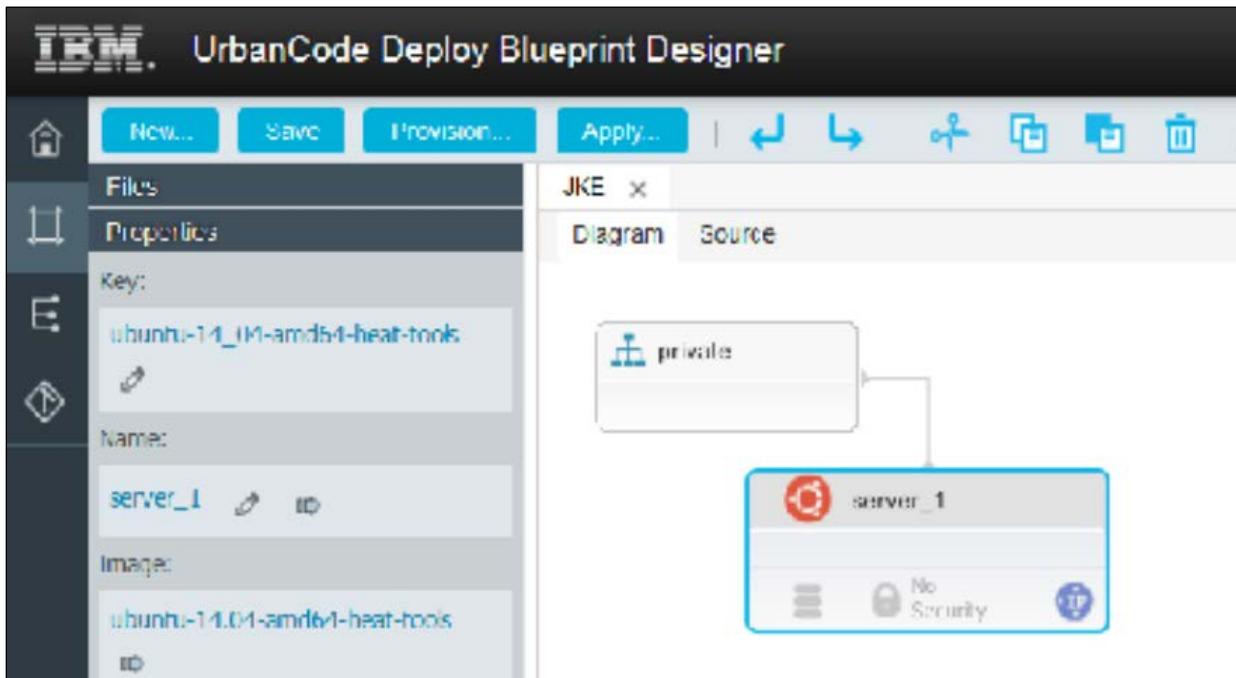
Exercise 2:

Tasks and results

Task 1. Provision the blueprint to a cloud environment.

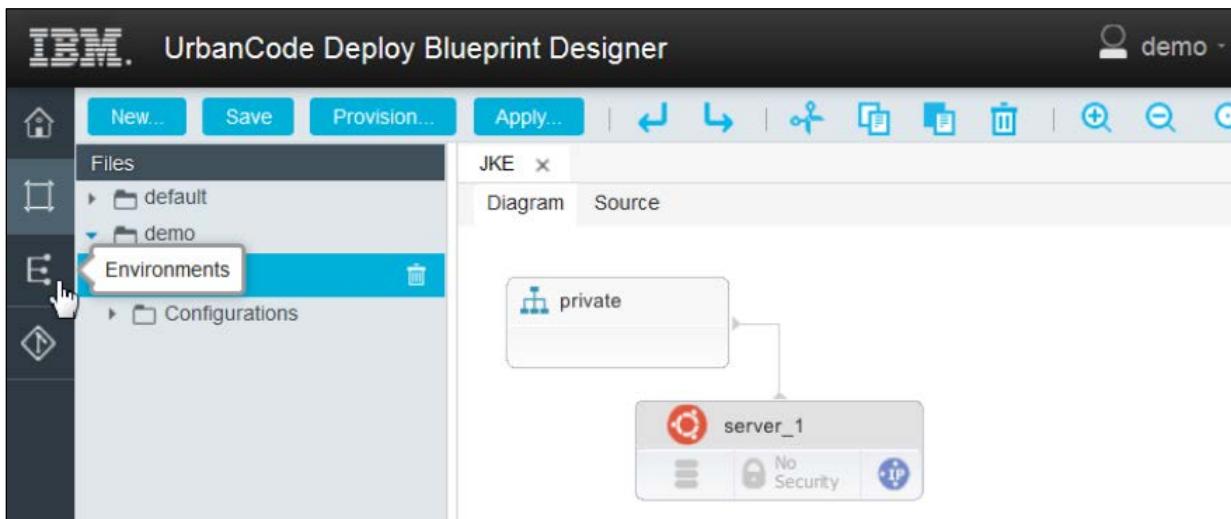
When you provision a cloud environment, you set aside resources on the cloud that you specified through a blueprint.

- Click **Provision**.



- In the “Provision Blueprint to new Environment” window, configure as follows:
 - Environment Name:** JKE-DEV
 - Click **Image Parameters**.
 - In the **Flavor** field, select **m1.medium**. The flavor represents the size of a provisioned image.
 - In the **Key Name** field, select **demo-key**. The key name represents the security key for a provisioned image.
 - Click **Network Parameters**, and in the **Public Network ID** field, select **public**.
- Click **Provision**.

4. On the left side of the page, click the **Environments** tab.



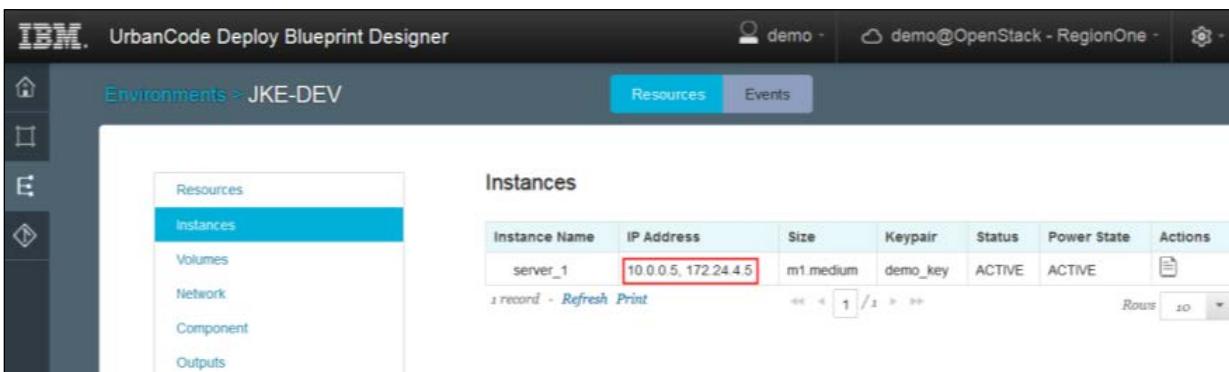
5. Click the twisty to expand the list of resources and observe the progress of the deployment.

The screenshot shows the UrbanCode Deploy Blueprint Designer interface with the 'Environments' tab selected. A table lists environments, applied blueprints, cloud projects, regions, last update times, virtual instances, last orchestration events, and actions. One row is expanded for 'JKE-DEV', showing three resources: 'ubuntu-14_04-amd64_to_private_port', 'ubuntu-14_04_amd64_to_private_port_floating_ip', and 'ubuntu-14_04-amd64'. The 'ubuntu-14_04-amd64' resource is detailed with its type as 'OS::Nova::Server', status as 'ACTIVE', and IP address as '10.0.0.5,172.24.4.5'. The 'Status' column indicates '100% Active' and the 'Stack Status' column shows 'Stack CREATE completed successfully' with green checkmarks for 'Create Complete'.

6. In the Environment column, click **JKE-DEV** to open the resources.

The screenshot shows the UrbanCode Deploy Blueprint Designer interface with the 'Environments' tab selected. A table lists environments, applied blueprints, cloud projects, and regions. The 'JKE-DEV' environment row is highlighted with a blue background and a white cursor icon pointing to the 'Environment' column. Below the table, a detailed view of the 'ubuntu-14_04-amd64' resource is shown, listing its name, type ('OS::Neutron::Port'), and details including its power state as 'Active' and IP address as '10.0.0.5,172.24.4.5'.

7. Click **Instances** in the left navigation.
Observe the IP addresses assigned to the image.



The screenshot shows the UrbanCode Deploy Blueprint Designer interface. The top navigation bar includes the IBM logo, the title "UrbanCode Deploy Blueprint Designer", and user information "demo" and "demo@OpenStack - RegionOne". Below the header, the left sidebar has a "Resources" section with "Instances" selected, and other options like "Volumes", "Network", "Component", and "Outputs". The main content area is titled "Instances" and displays a table with one record. The table columns are "Instance Name", "IP Address", "Size", "Keypair", "Status", "Power State", and "Actions". The single row shows "server_1" as the instance name, "10.0.0.5, 172.24.4.5" as the IP address, "m1.medium" as the size, "demo_key" as the keypair, "ACTIVE" as both status and power state, and a "Details" icon in the Actions column. Navigation controls at the bottom show "1 record", "Refresh", "Print", and a page number "1 / 1". A "Rows" dropdown is set to 10.

Instance Name	IP Address	Size	Keypair	Status	Power State	Actions
server_1	10.0.0.5, 172.24.4.5	m1.medium	demo_key	ACTIVE	ACTIVE	

When the stack is finished provisioning, ensure the server is externally accessible by opening a command prompt and pinging the floating IP to check for a response.

8. Click the **Konsole** icon on the desktop, and then type *ping* and the floating IP address. In this exercise, the floating IP address is 172.24.4.5.
9. Press **Ctrl+C** to stop receiving packets, and ensure that the packets were transmitted successfully.

Exercise 3

Observe the provisioned environment in OpenStack

Exercise 3: Observe the provisioned environment in OpenStack

Exercise 3: Observe the provisioned environment in OpenStack

In this exercise, you observe the provisioned environment in OpenStack.

To do this, you will:

- Observe the provisioned environment in OpenStack

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 3:

Tasks and results

Task 1. Observe the provisioned environment in OpenStack.

1. In the browser, switch to the **OpenStack** tab to see the created images.
2. In the panel on the left, click **Project > Compute > Instances**.

You can see the deployed instance of server_1. Again, you can see the IP address that is assigned to the public network, which is the same as the one listed in the Blueprint Designer.

The screenshot shows the Bluebox OpenStack Instances dashboard. On the left, there is a navigation sidebar with sections for Project (Compute selected), Overview, Instances (selected), Volumes, Images, Access & Security, Network, and Orchestration. The main area is titled 'Instances' and contains a table with the following data:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
server_1	ubuntu-14.04-amd64	10.0.0.5 Floating IPs: 172.24.4.5	m1.medium	demo_key	Active	nova	None	Running	4 minutes	<button>Create Snapshot</button>

Below the table, a message says 'Displaying 1 item'.

Exercise 4

Verify no activity in IBM UrbanCode Deploy

Exercise 4: Verify no activity in IBM UrbanCode Deploy

Exercise 4: Verify no activity in IBM UrbanCode Deploy

In this exercise, you verify that there is no activity in IBM UrbanCode Deploy.

To do this, you will:

- Verify that there is no activity in IBM UrbanCode Deploy

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 4:

Tasks and results

Task 1. Verify no activity in IBM UrbanCode Deploy.

1. In the browser, switch to **UrbanCode Deploy** tab.
2. From the top navigation, click the **Applications** tab.
3. In the **Name** column, click **JKE**.

The screenshot shows the 'Applications' tab selected in the top navigation bar. A single application entry for 'JKE' is listed in the table below. The 'Actions...' button next to the JKE entry is highlighted with a mouse cursor. The table columns include Name, Template, Description, Created, and By.

Name	Template	Description	Created	By
JKE		Sample JKE Banking application	7/29/2016, 3:13 PM	admin

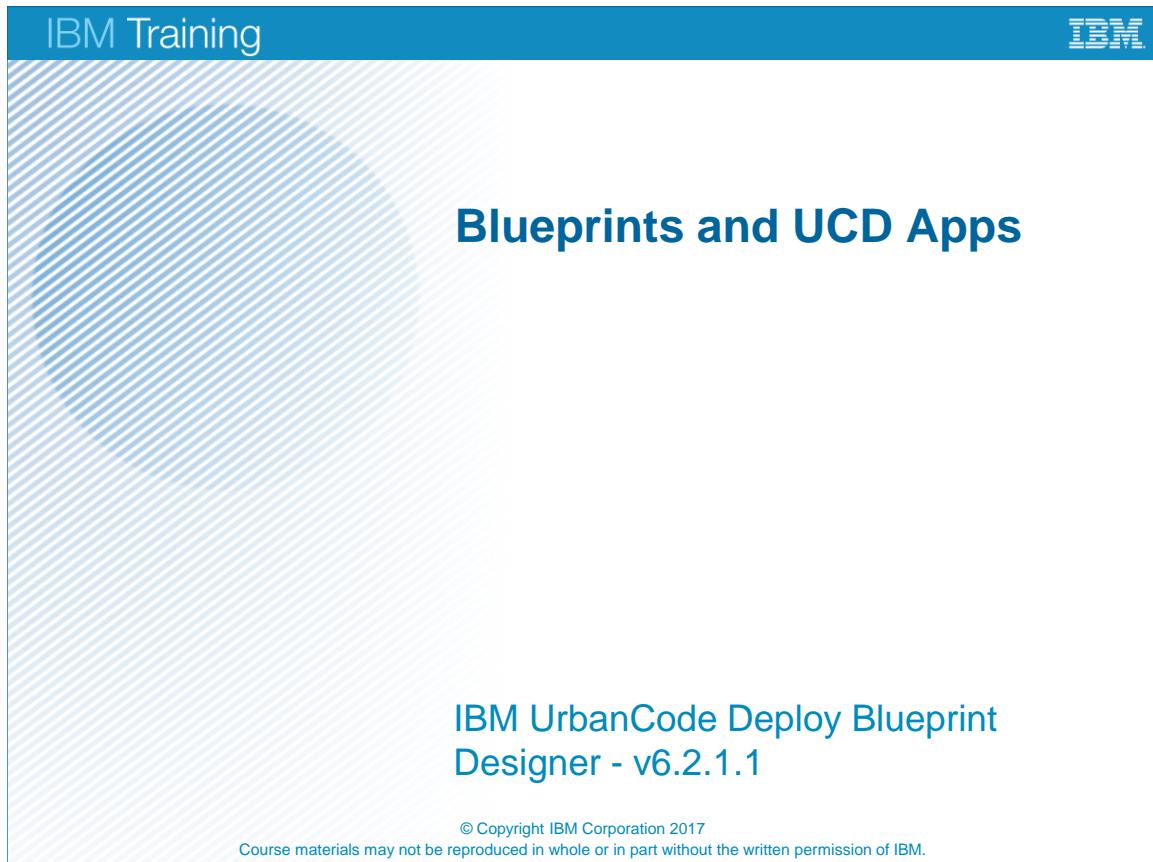
Observe that no environments exist for the application.

The screenshot shows the details for the 'JKE' application. The 'Environments' tab is selected in the navigation bar. A message at the bottom states 'No Environments Found!'. The 'Create Environment' button is visible at the top of the environments section.

In the next unit, you install components to the OpenStack cloud. You will come back to this page to see the running environment.

In this unit, you created a blueprint that describes an environment. This blueprint specifies the type of system to be deployed, the network for the system, and that it should also be attached to a public network. In the next lab, you add components to the blueprint from IBM UrbanCode Deploy and observe the provisioning.

Unit 4 Blueprints and UCD Apps



The slide has a blue header bar with 'IBM Training' on the left and the IBM logo on the right. The main title 'Blueprints and UCD Apps' is centered in large blue text. Below it, a subtitle 'IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1' is also in blue. At the bottom, there is a small copyright notice: '© Copyright IBM Corporation 2017' and 'Course materials may not be reproduced in whole or in part without the written permission of IBM.'

IBM Training

Blueprints and UCD Apps

IBM UrbanCode Deploy Blueprint
Designer - v6.2.1.1

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the written permission of IBM.

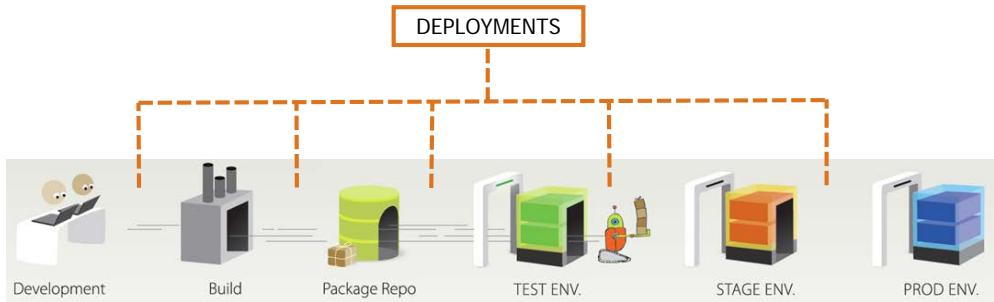
Unit objectives

- Review the application, component, and deployment processes in IBM UrbanCode Deploy
- Explore the options of the Blueprint Designer that relate to application components
- Modify the existing blueprint to include application components
- Explain how the IBM UrbanCode Deploy processes are triggered in the source code
- Deploy a blueprint to a cloud environment and view the changes in IBM UrbanCode Deploy and OpenStack
- Delete cloud environments and confirm that the resources are removed

Unit objectives

When you provision a cloud environment, you set aside resources on the specified cloud that you assigned through a blueprint. By using the Blueprint Designer, you can model a full-stack environment, including the infrastructure and application, in a single model and deploy it to the cloud in a single step.

Deployment process



Blueprints and UCD Apps

© Copyright IBM Corporation 2017

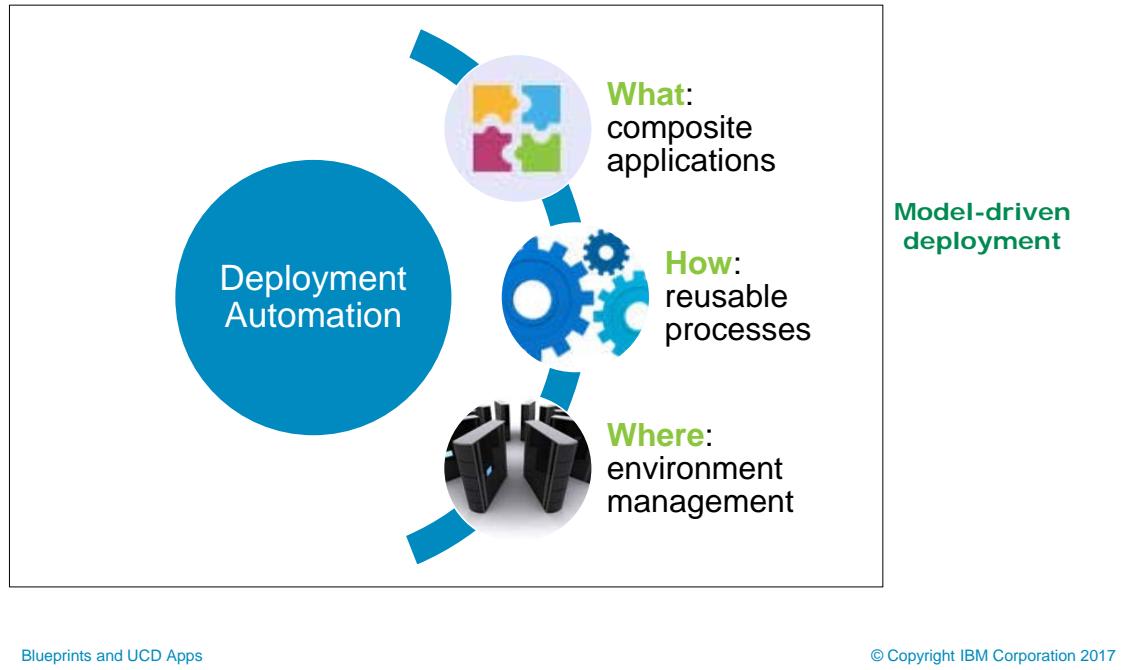
Deployment process

In deployment promotions, environments build on one another to increase the quality of an application before it reaches its intended user.

Applications do not need to go through a set number of environments, but these are typical:

- **Development (DEV):** In the *Development* environment, developers build and deploy code in a test lab, and the development team tests the application at the most basic level. When the application meets certain criteria for advancement, it moves to the next environment.
- **System Integration Testing (SIT):** In the *System Integration Testing* environment, the application is tested to ensure that it works with existing applications and systems. When the application meets the criteria of this environment, it's deployed to the next environment.
- **User Acceptance Testing (UAT):** In the *User Acceptance Testing* environment, the application is tested to ensure that it provides the required features for end users. This environment usually is production-like. When the application passes these requirements, it's promoted to the final environment.
- **Production (PROD):** In the *Production* environment, the application is made available to users. Feedback is captured by monitoring the application's availability and functionality. Any updates or patches are introduced.

Application models include components, processes, and environments



Blueprints and UCD Apps

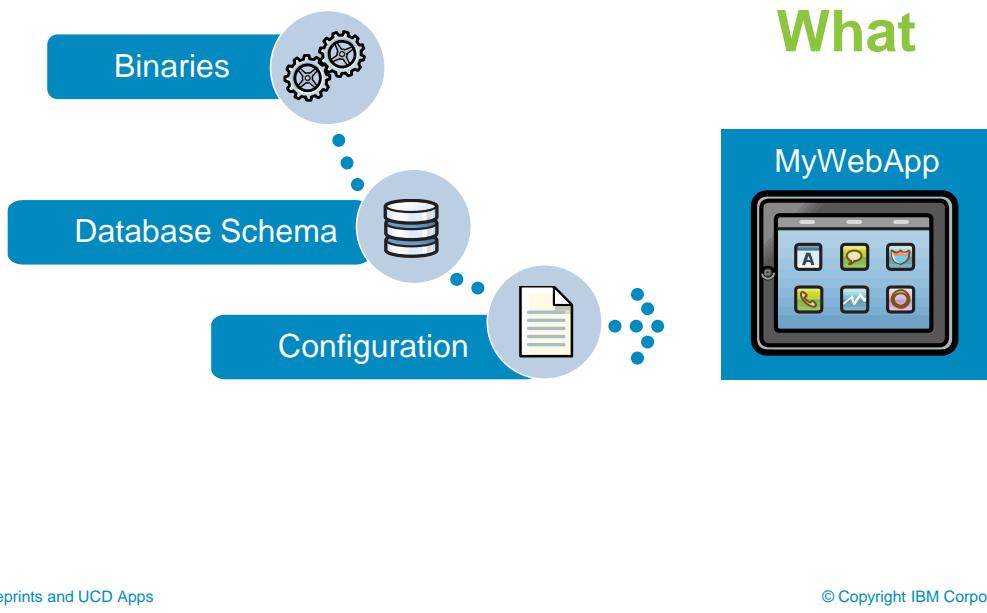
© Copyright IBM Corporation 2017

Application models include components, processes, and environments

IBM UrbanCode Deploy focuses on model-driven deployment. You model a plan to run software deployments, and then you reuse these processes to create automation.

An application brings together the pieces of your deployment and includes three different parts. First, it houses the components that you want to deploy. Second, it contains the application process that conducts deployment flow. And third, it contains information about the environment that you want to deploy to. This application model answers the questions of what gets deployed (the components), how it gets deployed (the processes), and where it gets deployed (the environments).

What is a Component?



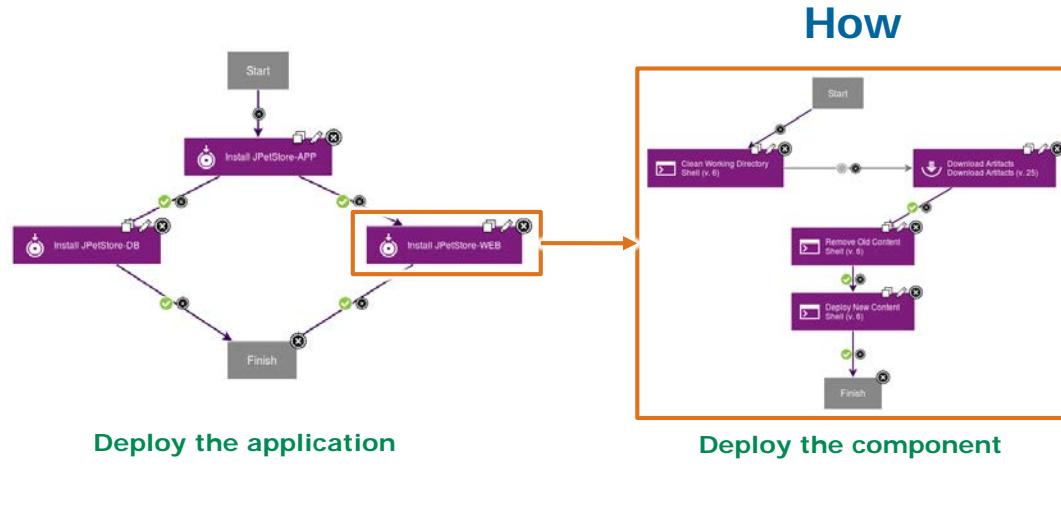
Blueprints and UCD Apps

© Copyright IBM Corporation 2017

What is a Component?

A component is the smallest functional unit of an application. Components are containers for deployable artifacts and identify the "what" of a deployment. This includes any files, images, and databases that are associated with a software project.

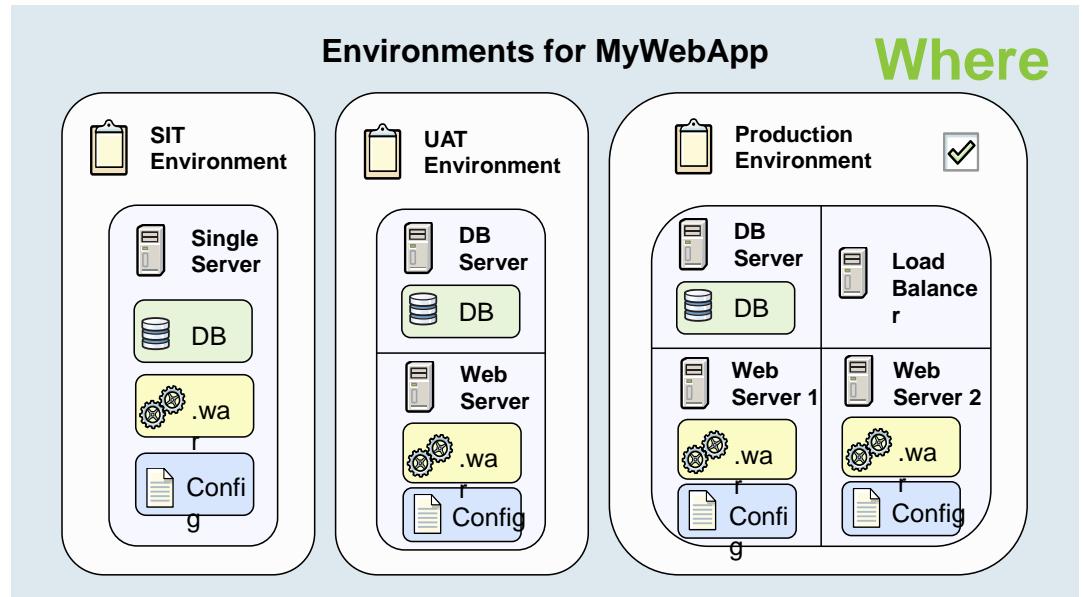
What are Processes?



What are Processes?

Processes are lists of steps and connections between those steps. Processes describe the "how" of a deployment and perform actions on components. The component process includes instructions, or steps, that operate on a component. The application process determines what components are deployed together and in what order.

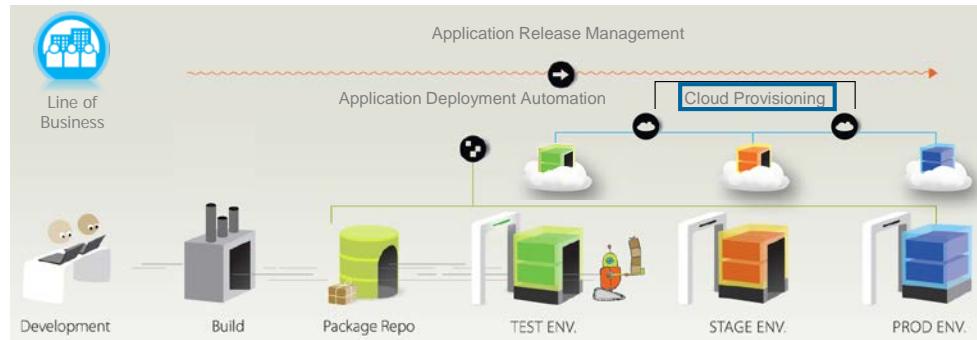
Environment



Environment

An environment is a collection of resources that hosts an application. The "where" part of the model refers to the target's hosts and environments. You can define where components are deployed and capture configuration settings for each deployment environment for an application.

Provisioning infrastructure is the current bottleneck in the delivery pipeline



Blueprints and UCD Apps

© Copyright IBM Corporation 2017

Provisioning infrastructure is the current bottleneck in the delivery pipeline

Development and testing teams are now optimized with agile, service virtualization, and deployment automation. Developers are increasingly becoming DevOps practitioners, demanding access to the full-environment from application-layer and down. This shift puts pressure on full-stack provisioning, which is the next bottleneck for the continuous delivery of applications. Addressing this bottleneck is driving the need for the Blueprint Designer.

Issues that relate to deploying cloud environments include the following topics:

- Application workloads include infrastructure, middleware, and applications.
- Different people have expertise in each area.
- I have different automation tools for each layer.
- Let me choose when to re-provision and when to update.
- Other times, I want to deploy changes into an existing environment, especially production.
- Don't make me pick a cloud.
- I want to be able to test in one cloud and run production in another.

The screenshot shows the IBM UrbanCode Deploy interface. On the left, a sidebar titled 'Components' lists several components: Referenced Component, jke.db, jke.war, MySQL Server, ucd-agent-linux-x86_64, ucd-agent-win-x86_64, and WebSphere Liberty Profile. An orange arrow points from the 'Components' list in the sidebar to the 'Components' list in the main view. The main view shows a table of components with columns: Name, Template, Description, Created, and By. The data in the table is as follows:

Name	Template	Description	Created	By
jke.db		This component contains the process to setup and configure the JKE database on MySQL Server.	7/29/2016, 3:13 PM	admin
jke.war		JKE Banking Sample Application Web Application Resource Archive	7/29/2016, 3:13 PM	admin
MySQL Server		This component contains the process to install MySQL Server	7/29/2016, 3:12 PM	admin
ucd-agent-linux-x86_64		Agent package with embedded JRE for x86_64 Linux	6/13/2016, 4:56 AM	admin
ucd-agent-win-x86_64		Agent package with embedded JRE for x86_64 Windows	6/13/2016, 4:56 AM	admin
WebSphere Liberty Profile		This component contains the binaries and process to install WebSphere Liberty Profile	7/29/2016, 3:12 PM	admin

At the bottom of the main view, there are buttons for Refresh and Print, and a pagination control showing 1 / 1 of 10 records. The sidebar also includes sections for Images, Networks, Security, Storage, Policies, and Templates.

The Components drawer

You add your application components to the virtual images and select processes to run to deploy those components. When you provision the environment, the server deploys each component to the virtual image and runs the processes. In this way, you can use a single blueprint to model both your infrastructure and your applications.

The components that you can see in the blueprint editor depend on your permissions. If your blueprint design server account is connected to an account on the IBM UrbanCode Deploy server, you see the components that you have access to on that server.

IBM Training IBM

Components adding to a blueprint

Application process in IBM UrbanCode Deploy

```

graph TD
    Start([Start]) --> InstJenkinsWAR[Install Jenkins WAR]
    Start --> InstJenkinsOS[Install Jenkins OS]
    Start --> InstJenkinsDB[Install Jenkins DB]
    InstJenkinsWAR --> Restart([Restart])
    InstJenkinsOS --> Restart
    InstJenkinsDB --> Restart
  
```

[Blueprints and UCD Apps](#) © Copyright IBM Corporation 2017

Components adding to a blueprint

When you add components to a blueprint, you are modeling the application process and the engine automatically installs the components. You do not run an application process as you do in IBM UrbanCode Deploy, as seen on the right. Instead, the blueprint runs component processes directly.

IBM Training

IBM

Deploying sequence for components

The screenshot shows the IBM Cloud Foundry interface with the following details:

- Main Workspace:** Shows a component tree with nodes: private, ubuntu-14_04-amd64-heat-tools, jke.db, MySQL Server, jke.war, and WebSphere Liberty Profile.
- Policies Drawer:** Contains options like New Load Balancer, New AutoScaling Group, and Deployment Sequence.
- Deployment Sequence Modal:** A dialog box titled "Deployment Sequence" lists the deployment sequence for each component:
 - MySQL Server (deploy) | ubuntu-14_04-amd64-heat-tools: LATEST | deploy
 - jke.db (deploy) | ubuntu-14_04-amd64-heat-tools: LATEST | deploy
 - WebSphere Liberty Profile (deploy) | ubuntu-14_04-amd64-heat-tools: LATEST | deploy
 - jke.war (deploy) | ubuntu-14_04-amd64-heat-tools: LATEST | deploy
- Bottom Navigation:** Includes links for Blueprints and UCD Apps, and Copyright IBM Corporation 2017.

Deploying sequence for components

You specify the deployment order of components in an application by dragging the Deployment Sequence resource from the Policies drawer of the palette and selecting the number. Because the deployment sequence is modeled as a tree and not a sequential list, more than one component can have the same deployment sequence number. Subsequent components are not deployed until all previous components are deployed and components without deployment sequence numbers are deployed asynchronously.

The slide illustrates the component process in IBM UrbanCode Deploy. It features two screenshots of the application's interface and a detailed process flow diagram.

Screenshots:

- Top Screenshot:** Shows the 'Components' page with a list of components. One component, 'jke.war', is selected, and its details are shown in a modal below.
- Bottom Screenshot:** Shows the 'Processes' tab for the selected component 'jke.war'. It displays a single process named 'deploy'.

Component process diagram:

```

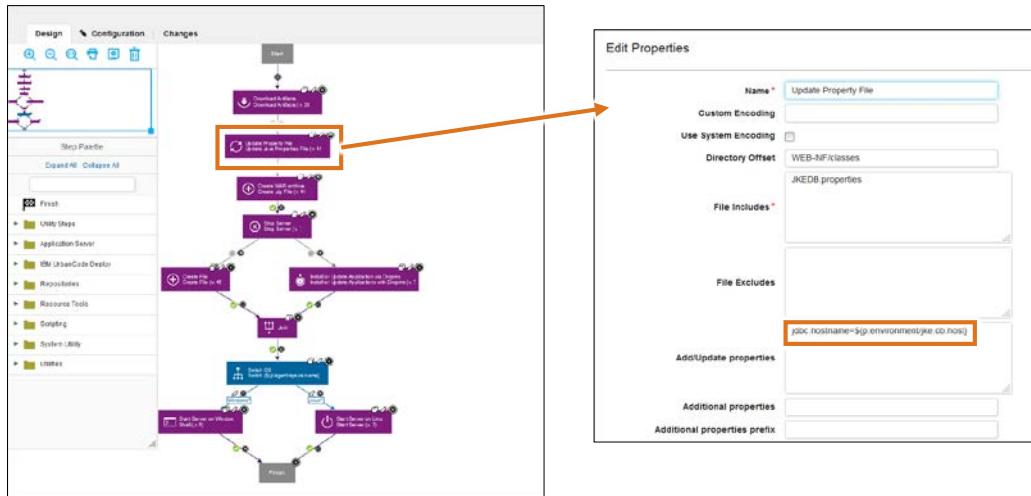
graph TD
    Start((Start)) --> Download[Download & Extract]
    Download --> CreateDB[Create DB]
    CreateDB --> StopServer1[Stop Server 1]
    StopServer1 --> CreateApp[Create App]
    CreateApp --> StopServer2[Stop Server 2]
    StopServer2 --> UpdateApp[Update Application]
    UpdateApp --> StartApp[Start Application]
    StartApp --> End((End))
    
```

The process starts with 'Download & Extract', followed by 'Create DB', 'Stop Server 1', 'Create App', 'Stop Server 2', 'Update Application', 'Start Application', and concludes with 'End'.

Component process in IBM UrbanCode Deploy

A component process is a series of user-defined steps that operate on a component's artifacts. Each component has at least one process that is defined for it and can have several.

Process steps and their environment properties



Process steps and their environment properties

Components can have component environment properties; these properties transfer to the environment when the component is deployed. An environment property overrides the value that is set on a component environment property that has the same name. For example, if you deploy a web application to three environments, where each environment has the application server in a different location, you can specify this location in an environment property on each environment.

In this example, the **File includes** field specifies a file, JKEDB.properties. In that file, this step searches for the string jdbc.hostname and replaces the value with the value of the IBM UrbanCode Deploy environment property called jke.db.host from the blueprint. The ability to tie this environment property to a HOT document parameter allows the blueprint design server to reuse blueprints without hardcoding particular values. Each environment can create a different value for this property.

IBM Training

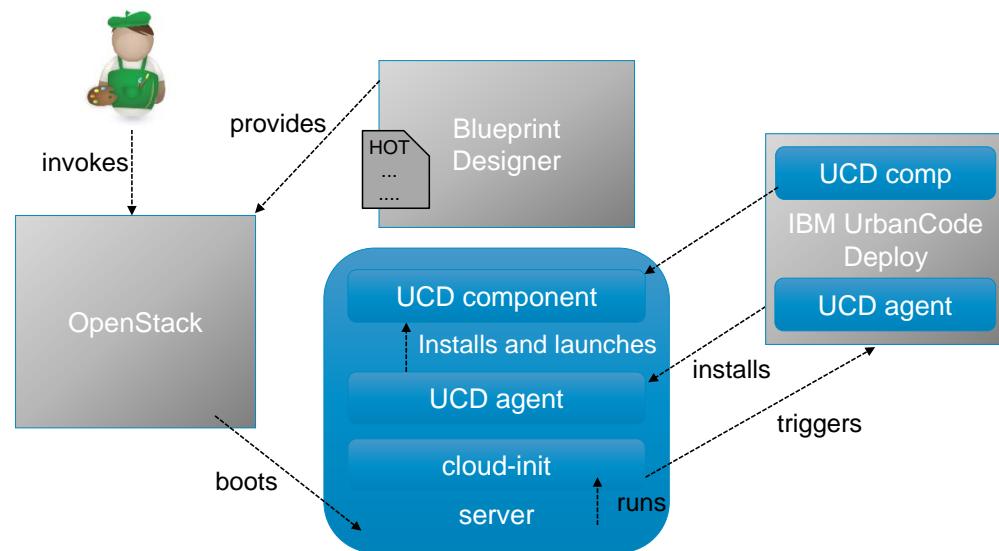
Setting properties in the Blueprint Designer

Blueprints and UCD Apps © Copyright IBM Corporation 2017

Setting properties in the Blueprint Designer

You can set properties that might vary depending on the environment in question. In this example, the jke.db.host property in the Blueprint Designer is set to **localhost** and replaces the **Default Value** field for jke.db.host in the IBM UrbanCode Deploy server during deployment.

Provisioning an environment: Behind the scenes



Blueprints and UCD Apps

© Copyright IBM Corporation 2017

Provisioning an environment: Behind the scenes

When you click the **Provision** button in the Blueprint Designer, processes run behind the scenes to prepare environments and deploy the applications.

During provisioning, Nova, which is responsible for provisioning the environment, starts the virtual machine on the server.

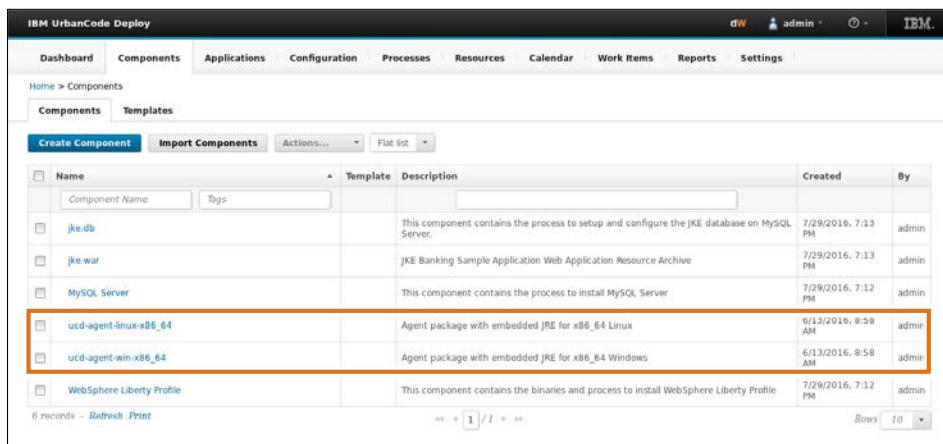
After the operating system is running, it runs cloud-init. Cloud-init is a set of scripts and utilities that cloud systems use to initialize and configure instances.

The agents reach out to the server, and the server registers them. Agents communicate with the IBM UrbanCode Deploy Server and complete the actual work of deploying the components of an application.

IBM UrbanCode Deploy can then push software to this virtual machine.

IBM Training 

Agents on the IBM UrbanCode Deploy server



Name	Template	Description	Created	By
jke.db		This component contains the process to setup and configure the JKE database on MySQL Server.	7/29/2016, 7:13 PM	admin
jke.war		JKE Banking Sample Application Web Application Resource Archive	7/29/2016, 7:13 PM	admin
MySQL Server		This component contains the process to install MySQL Server	7/29/2016, 7:12 PM	admin
ucd-agent-linux-x86_64		Agent package with embedded JRE for x86_64 Linux	6/13/2016, 8:58 AM	admin
ucd-agent-win-x86_64		Agent package with embedded JRE for x86_64 Windows	6/13/2016, 8:58 AM	admin
WebSphere Liberty Profile		This component contains the binaries and process to install WebSphere Liberty Profile	7/29/2016, 7:12 PM	admin

Blueprints and UCD Apps © Copyright IBM Corporation 2017

Agents on the IBM UrbanCode Deploy server

The agents that the cloud-init calls for are located with the components.

IBM Training IBM

Provisioning of blueprint from the Blueprint Designer

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, there's a sidebar with various configuration options like 'Heat Template Version', 'Resource Tree', and 'Team Mappings'. The main workspace shows a diagram with a 'private' node connected to a 'ubuntu-14-amd64-hd' instance. A modal dialog titled 'Provision Blueprint to new Environment' is open in the center. It contains fields for 'Environment Name' (set to 'JKE-DEV'), 'Cloud Project' (set to 'demo@OpenStack'), and 'Configuration' (set to 'Select Configuration...'). Below these are sections for 'Agent Parameters', 'Image Parameters', 'Flavor' (set to 'm1 medium'), and 'Key Name' (set to 'demo_key'). At the bottom of the dialog is a 'Provision' button, which is highlighted with a red box.

Blueprints and UCD Apps © Copyright IBM Corporation 2017

Provisioning of blueprint from the Blueprint Designer

When you provision the blueprint, the blueprint design server provisions the cloud resources and creates an application environment in IBM UrbanCode Deploy. Then, the IBM UrbanCode Deploy server runs the component processes that are listed in the blueprint. In the following slides, you examine the component processes in IBM UrbanCode Deploy and observe the actions associated with provisioning and deploying.

IBM Training IBM

Deployment of the components begins

Record the floating IP for checking the application later

Resource Name	Type	Details	Status	Stack Status
ubuntu-14_04-amd64-heat-tools__to__private__port_resource_tree	OS: Neutron: Port	IBM: UrbanCode: ResourceTree	-	Create Complete
jke_db_sw_config	IBM: UrbanCode: SoftwareConfig: UCD	-	-	Create Complete
WebSphere_Liberty_Profile_sw_config	IBM: UrbanCode: SoftwareConfig: UCD	-	-	Create Complete
vcd_openl_insta_linux	OS: Heat: SoftwareConfig	IBM: UrbanCode: SoftwareConfig: UCD	-	Create Complete
MySQL_Server_sw_config	IBM: UrbanCode: SoftwareConfig: UCD	-	-	Create Complete
jke_war_sw_config	IBM: UrbanCode: SoftwareConfig: UCD	-	-	Create Complete
ubuntu-14_04-amd64-heat-tools__to__private__port_floating_ip	OS: Neutron: FloatingIP	-	-	Create Complete
ubuntu-14_04-amd64-heat-tools_mime	OS: Heat: MultipartMime	-	-	Create Complete
ubuntu-14_04-amd64-heat-tools	OS: Nova: Server	name = ubuntu-14_04-amd64-heat-tools, instance state = ACTIVE, ip address = 10.0.0.10, port = 8080	ACTIVE	Create Complete
MySQL_Server	IBM: UrbanCode: SoftwareDeploy: UCD	-	-	Create Complete
jke_db	IBM: UrbanCode: SoftwareDeploy: UCD	-	-	Create In Progress
WebSphere_Liberty_Profile	IBM: UrbanCode: SoftwareDeploy: UCD	-	-	Inst Complete
jke_war	IBM: UrbanCode: SoftwareDeploy: UCD	-	-	Inst Complete

© Copyright IBM Corporation 2017

Deployment of the components begins

The components are now part of the deployment process, and you can see them listed as resources in the environment. Take note of the floating IP for the web server, as you can connect to this address after deployment completes to ensure the server is online.

IBM Training

Components in IBM UrbanCode Deploy on running of deployment process

The screenshot shows the IBM UrbanCode Deploy web interface. At the top, there's a navigation bar with links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The user is logged in as 'admin'. Below the navigation is a breadcrumb trail: Home > Applications > JKE. The main title is 'Application: JKE'. Underneath, it shows 'Created By' as admin and 'Created On' as 7/29/2016, 3:13 PM. A 'Description' field contains 'Sample JKE Banking application'. Below this is a navigation bar with tabs: Environments, History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. The 'Components' tab is selected. A sub-header says 'Create Environment' and includes search fields for 'Search by Name' and 'Search by Blueprint'. A note says 'Drag environments by their names to re-order them.' Below this is a table titled 'JKE-DEV' showing one component: MySQL Server, Version 5.5.49, Date Created 8/24/2016, 3:55 PM. The status is Active and the Compliancy is 1/1 (Compliant). At the bottom of the table are 'Refresh' and 'Print' buttons.

Blueprints and UCD Apps

© Copyright IBM Corporation 2017

Components in IBM UrbanCode Deploy on running of deployment process

After the agents come online, within two to three minutes of starting the operating system, you see them display in IBM UrbanCode Deploy. Because components are not displayed in the table until the blueprint design server begins the process of deploying them, it might take a few moments for you to see the components.

IBM Training

Component process

The screenshot shows the IBM UrbanCode Deploy interface. On the left, there's a sidebar with a tree view of steps: 'Download Attributes (physical - Server in env)', 'Install MySQL', 'Configure MySQL', 'Open Port in Firewall', and 'Start MySQL'. The 'Download Attributes' node is highlighted with a red box. An orange arrow points from this red box to the text below. The main area shows a flowchart of the component process, starting with 'Start', followed by 'Download Attributes (x 26)', 'Install MySQL (Server in env)', 'Configure MySQL (Server in env)', 'Open Port in Firewall (Server in env)', and finally 'Start MySQL (Server in env)'. The process is currently at the 'Configure MySQL' step. The bottom right corner of the screenshot contains the copyright notice: '© Copyright IBM Corporation 2017'.

Component process

You can view the request of a component process in IBM UrbanCode Deploy and see the steps as they run. These steps are the same ones that were modeled in the component process.

IBM Training IBM

Components deployed on the environment

Component	Version	Date Created	Snapshot	Properties	Status	Compliance	Actions
jke.war	1.0	8/24/2016, 3:56 PM			Active	Compliant (1/1)	
WebSphere Liberty Profile	8.5.5.5	8/24/2016, 3:57 PM			Active	Compliant (1/1)	
jke.db	1.0	8/24/2016, 3:57 PM			Active	Compliant (1/1)	
MySQL Server	5.5.49	8/24/2016, 3:55 PM			Active	Compliant (1/1)	

[Blueprints and UCD Apps](#) © Copyright IBM Corporation 2017

Components deployed on the environment

An environment is a user-defined collection of resources that hosts an application. An environment is the application's mechanism for bringing together components with the agent that deploys them. Usually, resources are found on the same host where the agent that manages them is located. A host can be a physical system, virtual machine, or cloud-based.

IBM Training IBM

Components are deployed to environments by agents

Name	Inventory	Status	Description
Resource Name			Base Resource for environment JKE-DEV
jke.db	1.0	Online	UCD jke.db
jke.war	1.0		UCD jke.war
MySQL Server	5.5.49 (+ 1 more)		UCD MySQL Server
WebSphere Liberty Profile	8.5.5.5		UCD WebSphere Liberty Profile

Blueprints and UCD Apps © Copyright IBM Corporation 2017

Components are deployed to environments by agents

Resources display in a hierarchical tree structure. A component-type resource can point directly to the agent that deploys it. A resource represents a deployment target, such as a physical server, virtual machine, or database.

IBM Training

The blueprint displays as an application process on the IBM UrbanCode Deploy server

The screenshot illustrates the integration between the Blueprint Designer and the Deploy server. On the left, the Blueprint Designer shows a 'private' blueprint containing components like MySQL Server, WebSphere Liberty Profile, and JBoss Seam. An orange arrow points from this blueprint to the Deploy server's 'Applications' screen on the right. On the Deploy server, a new application named 'Sample JDE Banking application' is listed. A second orange arrow points from the 'Components' tab of this application to the 'Components' section of the Deploy server's main interface, where three components are detailed: 'JDE DB', 'JDE EAR', and 'WebSphere Liberty Profile'. The 'WebSphere Liberty Profile' component is highlighted with a red box.

Blueprints and UCD Apps

© Copyright IBM Corporation 2017

The blueprint displays as an application process on the IBM UrbanCode Deploy server

When you include components in a blueprint, the blueprint shows as an application process on the IBM UrbanCode Deploy server. You can use this blueprint through the application process, just like you use any application process on the server. For example, you can apply the blueprint to an existing environment by running the application process.

IBM Training IBM

Components deployed shown on the Environment tab

The screenshot shows two main windows. The top window is the 'Environments' page in the Blueprint Designer, specifically for the 'JKE-DEV' environment. It displays a table of components with columns for Name, Version, Process, and Server. The components listed are 'jke-00' (WebSphere Liberty Profile), 'jke-war' (WebSphere Application), and 'MySQL_Server' (MySQL Server). The bottom window is the 'IBM UrbanCode Deploy' application details page for 'Application: JKE'. It shows the application was created by 'demo' on '2/29/2016, 3:15 PM' with a description 'Sample JKE Banking Application'. The 'Components' tab is selected, showing the same three components: 'jke-00', 'jke-war', and 'MySQL_Server'. An orange arrow points from the 'MySQL_Server' row in the top table to its corresponding entry in the bottom table.

Blueprints and UCD Apps © Copyright IBM Corporation 2017

Components deployed shown on the Environment tab

The new environment is displayed on the Environments page of the Blueprint Designer and in the list of application environments on the IBM UrbanCode Deploy server.

IBM Training IBM

Successfully provisioned environment

Confirms that the web server is online

Confirms that the database server is working

Auto Loans
Rates as low as 4.9%

Automatic Bill Pay

Totally Free Checking

Services

- Certificates of Deposit
- Open New Account
- Close Account
- Policy Change Request
- Policy Change Status

Financial Services

- Home Equity Lines
- Vacation Home Finance
- Today's Rates
- Refinancing

Account Overview

Account Type	Available Balance
Checking Account	\$2,000.00
Savings Account	\$12,500.00
IRA Retirement Account	\$25.00
Mutual Fund Savings Account	\$500.00

Blueprints and UCD Apps

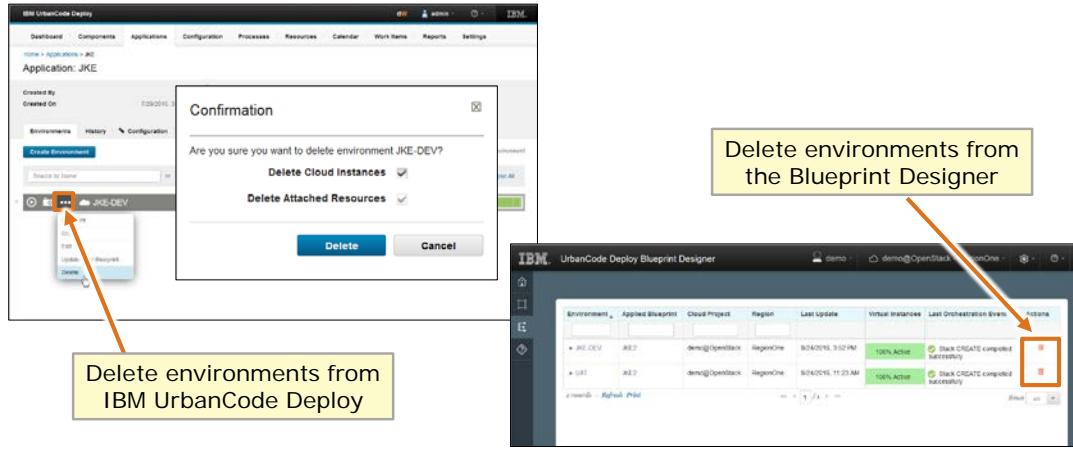
© Copyright IBM Corporation 2017

Successfully provisioned environment

Now that you have provisioned the environment and deployed applications to that environment, you can verify that the application is working by connecting to the external IP address.

When the JKE application home page opens, you know that the Blueprint Designer provisioned a full-application stack into the cloud. When the application displays the account balances, you know that the application is accessing the database.

Deleting an environment from either UrbanCode Deploy or Blueprint Designer



Deleting an environment from either UrbanCode Deploy or Blueprint Designer

Whether you need an environment for development, test, continuous integration, or production, you can create that environment and deploy the application to it automatically, directly from the IBM UrbanCode Deploy server. When you're done, you can delete the environment with a single click.

You can delete provisioned cloud environments from the Blueprint Designer. When you delete a cloud environment, the virtual nodes and all of the data on them are deleted. In most cases, you cannot undo this action.

IBM Training IBM

Environments are deleted across all tools

The screenshot shows two interface windows. The top window is the 'UrbanCode Deploy Blueprint Designer' showing a table of environments. One row for 'JKE-DEV' has a status message 'Stack DELETE started' with a delete icon. A green annotation 'Deleted environments' with an arrow points to this message. The bottom window is a 'bluebox' interface showing an 'Instances' table with no items displayed, indicated by the message 'No items to display.' Another green annotation with an arrow points to this message.

Deleted environments

No items to display.

Blueprints and UCD Apps © Copyright IBM Corporation 2017

Environments are deleted across all tools

The instances on the cloud are deleted, and the resources in the resource tree are deleted.

Unit summary

- Review the application, component, and deployment processes in IBM UrbanCode Deploy
- Explore the options of the Blueprint Designer that relate to application components
- Modify the existing blueprint to include application components
- Explain how the IBM UrbanCode Deploy processes are triggered in the source code
- Deploy a blueprint to a cloud environment and view the changes in IBM UrbanCode Deploy and OpenStack
- Delete cloud environments and confirm that the resources are removed

Unit summary

IBM UrbanCode Deploy's application model identifies what, how, and where in deployment automation. When you model applications in the Blueprint Designer, you replace the application process in the IBM UrbanCode Deploy server. A component process is a series of user-defined steps that operate on a component's artifacts.

You can use a single blueprint to model both your infrastructure and your applications. HOT document parameters allow the blueprint design server to reuse blueprints by passing environment properties to the IBM UrbanCode Deploy server without the need to hardcode particular values. Environments can be deleted from the Blueprint Designer or in IBM UrbanCode Deploy.

Exercises: Deploying applications to a cloud environment

- Add components from IBM UrbanCode Deploy to the server in the Blueprint Designer and set the deployment order
- Set the environment property in the Blueprint Designer that is passed to IBM UrbanCode Deploy during deployment
- View the provisioned resources in IBM UrbanCode Deploy, the Blueprint Designer, and OpenStack
- Verify the working application as a full-application stack
- Delete an environment

Exercises: Deploying applications to a cloud environment

When integrated with IBM UrbanCode Deploy, the blueprint can specify the application components to be installed. In these exercises, you build on the blueprint that you created in the previous lab by adding components from IBM UrbanCode Deploy to the blueprint. Then, you provision the environment to the cloud.

Exercise 1

Add IBM UrbanCode Deploy components to the blueprint

Exercise 1: Add IBM UrbanCode Deploy components to the blueprint

Exercise 1: Add IBM UrbanCode Deploy components to the blueprint

In this exercise, you add IBM UrbanCode Deploy components to the blueprint.

To do this, you will:

- View the components from the integration with IBM UrbanCode Deploy.
- Add IBM UrbanCode Deploy components to the blueprint.
- Setting the order to deploy the components.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 1:

Tasks and results

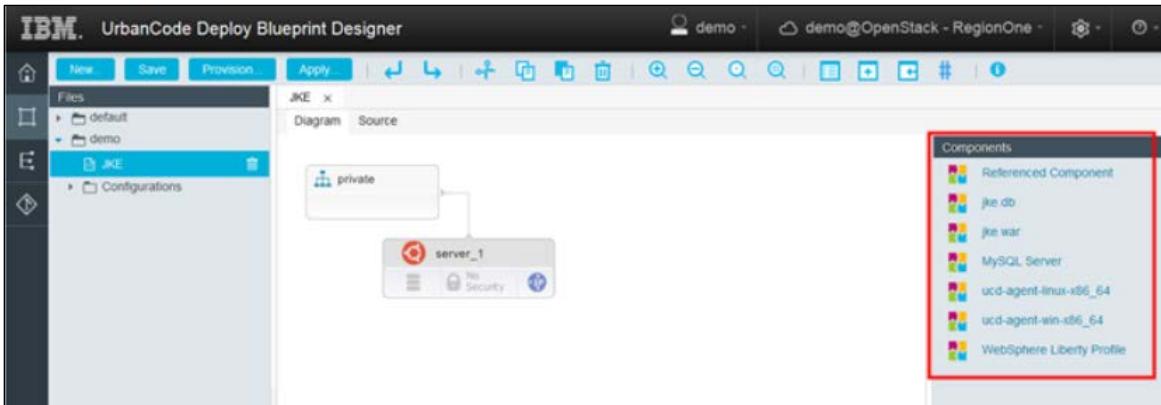
Task 1. View the components from the integration with IBM UrbanCode Deploy.

Components represent deployable items along with component processes. A component process is a series of user-defined steps that operate on the component or its artifacts.

1. From the browser, switch to the **Blueprint Designer** bookmark.
2. Ensure that you are logged in as the **demo** user by typing `demo` in the **User Name** field and `labstack` in the **Password** field.
3. From the home page, click the existing **JKE** blueprint.



In the previous unit, you granted access to both an OpenStack cloud and the IBM UrbanCode Deploy server to the **demo** user. With this access, the palette automatically has resources from this cloud. Notice that the components from IBM UrbanCode Deploy are in the Components drawer.



Task 2. Add IBM UrbanCode Deploy components to the blueprint.

You can now instruct the blueprint design server to make use of the predefined components in IBM UrbanCode Deploy by configuring the server with MySQL and WebSphere Liberty.

- From the Components drawer, select the **MySQL Server** component and drop it on to the compute resource. This component runs the steps that are necessary to install MySQL on the Linux image that is provisioned.



- In the Choose UCD Application window, ensure that **JKE** is selected, and click **OK**.

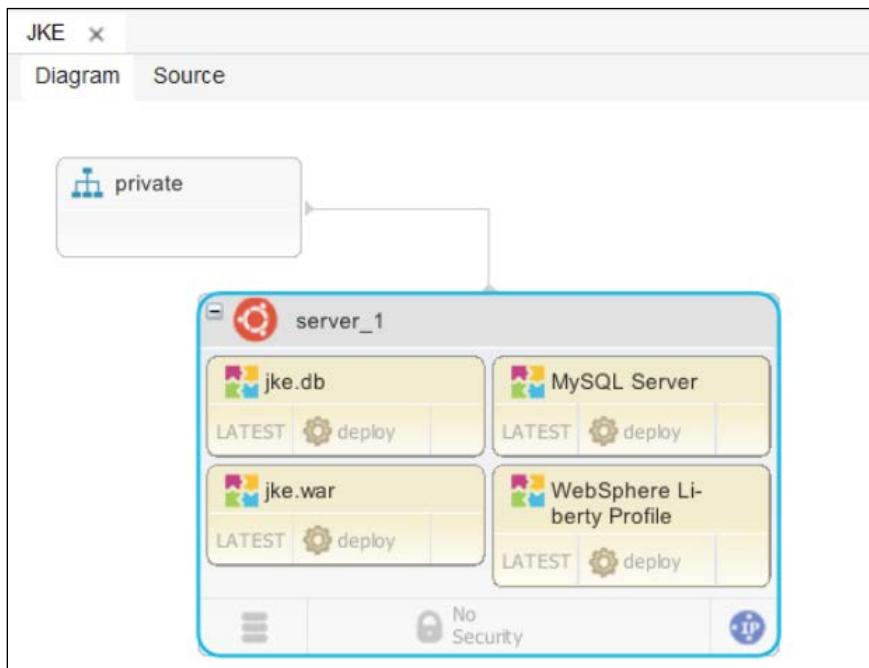
The MySQL Server component is added to the image:



- In the MySQL Server component, click **CHOOSE**, select **deploy** as the component process, and click **OK**.
- Drag the **jke.db** component to the same compute resource. This component creates the database for the JKE application to use when it is running.

5. Add the **WebSphere Liberty Profile** and **jke.war** components to the compute resource. The WebSphere Liberty Profile component installs the WebSphere Liberty Profile components on to this server. The jke.war component installs the EAR file on the web server.
6. In the WebSphere Liberty Profile component, click **CHOOSE**, select **deploy** as the component process, and click **OK**.

The blueprint now has the following resources:



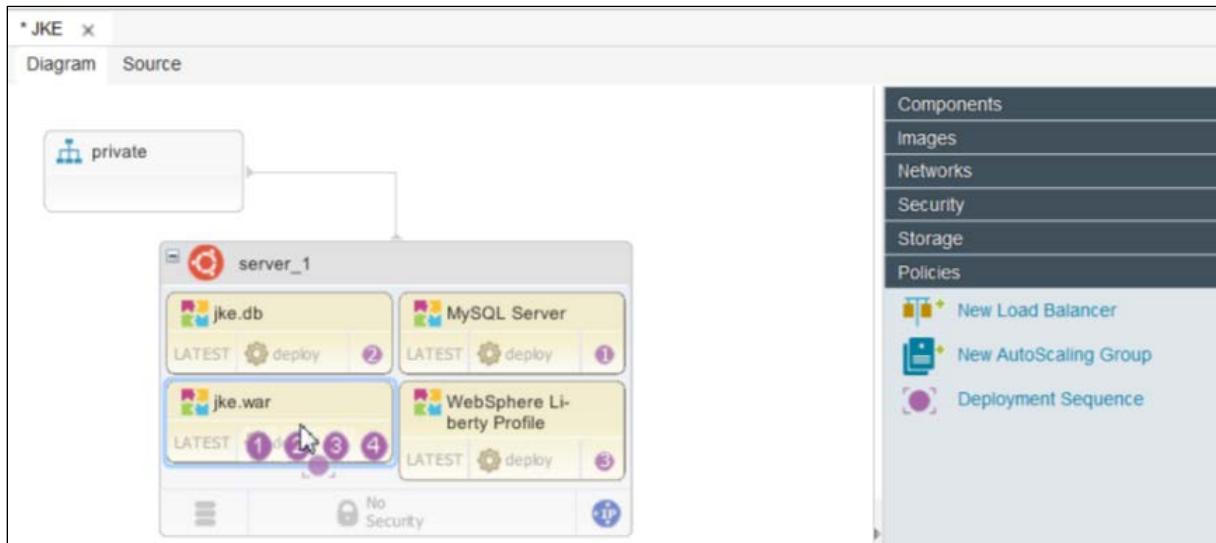
7. Save the blueprint.

Task 3. Set the order to deploy the components.

You can indicate the order in which the IBM UrbanCode Deploy components are deployed by using the Deployment Sequence resource.

Note: Because the deployment sequence is modeled as a tree, and not a sequential list, more than one component can have the same deployment sequence number. Subsequent components are not deployed until all previous components are deployed. Components without deployment sequence numbers are deployed asynchronously. Numbers are displayed on the components to indicate the deployment order.

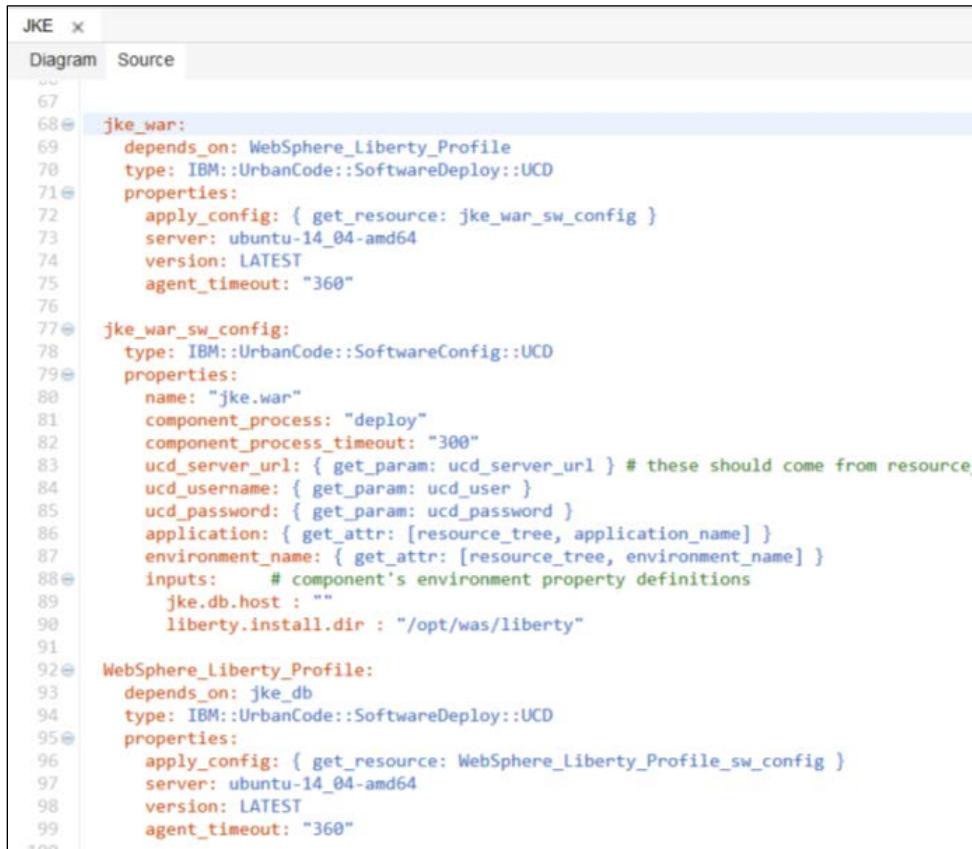
1. Open the **Policies** drawer, and drag the **Deployment Sequence** resource on to each of the components, on at a time, in the following order:
 - a. MySQL_Server
 - b. jke.db
 - c. WebSphere Liberty Profile
 - d. jke.war



Note: If you set the deployment sequence incorrectly when dragging, you can click the **Deployment Sequence** icon of any of the components and change the sequence.

2. Save the blueprint.

3. Click the **Source** tab, and view the text of the blueprint that you created. Scroll through the blueprint and observe some of the syntax and constructs.



```

JKE X
Diagram Source
67
68 @ jke_war:
69   depends_on: WebSphere_Liberty_Profile
70   type: IBM::UrbanCode::SoftwareDeploy::UCD
71 @ properties:
72   apply_config: { get_resource: jke_war_sw_config }
73   server: ubuntu-14_04-amd64
74   version: LATEST
75   agent_timeout: "360"
76
77 @ jke_war_sw_config:
78   type: IBM::UrbanCode::SoftwareConfig::UCD
79 @ properties:
80   name: "jke.war"
81   component_process: "deploy"
82   component_process_timeout: "300"
83   ucd_server_url: { get_param: ucd_server_url } # these should come from resource_
84   ucd_username: { get_param: ucd_user }
85   ucd_password: { get_param: ucd_password }
86   application: { get_attr: [resource_tree, application_name] }
87   environment_name: { get_attr: [resource_tree, environment_name] }
88 @ inputs: # component's environment property definitions
89   jke.db.host : ""
90   liberty.install.dir : "/opt/was/liberty"
91
92 @ WebSphere_Liberty_Profile:
93   depends_on: jke_db
94   type: IBM::UrbanCode::SoftwareDeploy::UCD
95 @ properties:
96   apply_config: { get_resource: WebSphere_Liberty_Profile_sw_config }
97   server: ubuntu-14_04-amd64
98   version: LATEST
99   agent_timeout: "360"
...

```

Exercise 2

Explore the application in IBM UrbanCode Deploy

Exercise 2: Explore the application in IBM UrbanCode Deploy

Exercise 2: Explore the application in IBM UrbanCode Deploy

In the previous exercise, you added components from IBM UrbanCode Deploy to the blueprint. Now you examine the component processes in IBM UrbanCode Deploy.

To do this, you will:

- Explore the component processes.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 2:

Tasks and results

Task 1. Explore the component processes.

- From the browser, switch to the **UrbanCode Deploy** tab.
- From the navigation at the top of the page, click **Components**.

You can see the four components that you used in the blueprint: jke.db, jke.war, MySQL Server, and WebSphere Liberty Profile. Each of these components contain user-defined processes that operate on them, usually by deploying them.

Note: In addition to the components, the agents ucd-agent-linux-x86_64 and ucd-agent-win-x86_64 are also listed. An agent is a lightweight process that runs on a deployment-target host and communicates with the IBM UrbanCode Deploy server. Agents do the actual work of deploying components and so relieve the server from the task, making large deployments that involve thousands of targets possible.

- In the **Name** column, click the WAR file, **jke.war**.

Name	Template	Description	Created	By
jke.db		This component contains the process to setup and configure the JKE database on MySQL Server.	7/29/2016, 3:13 PM	admin
jke.war	Actions...	JKE Banking Sample Application Web Application Resource Archive	7/29/2016, 3:13 PM	admin
MySQL Server		This component contains the process to install MySQL Server	7/29/2016, 3:12 PM	admin
ucd-agent-linux-x86_64		Agent package with embedded JRE for x86_64 Linux	6/13/2016, 4:58 AM	admin
ucd-agent-win-x86_64		Agent package with embedded JRE for x86_64 Windows	6/13/2016, 4:58 AM	admin
WebSphere Liberty Profile		This component contains the binaries and process to install WebSphere Liberty Profile	7/29/2016, 3:12 PM	admin

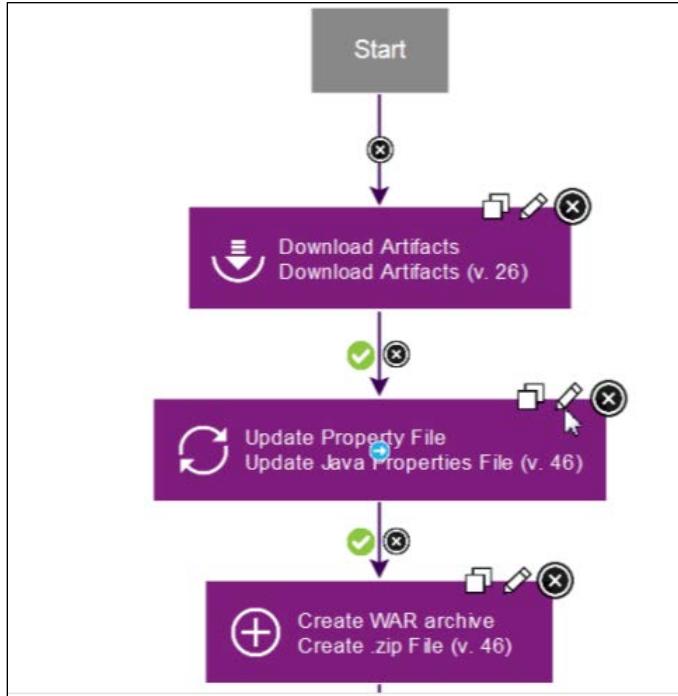
4. From the navigation for this component, click **Processes**.

The screenshot shows the 'IBM UrbanCode Deploy' interface. At the top, there's a navigation bar with links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, and Work Items. Below the navigation bar, the path 'Home > Components > jke.war' is displayed. The main title is 'Component: jke.war'. On the left, there's a sidebar with 'Created By' (admin), 'Created On' (7/29/2016, 3:13 PM), and 'Used By' (JKE). To the right, there's a 'Description' section with the text 'JKE Banking Sample Application Web Application Resource Archive'. Below this, a horizontal menu bar includes 'Dashboard', 'Usage', 'Configuration', 'Calendar', 'Versions', 'Processes' (which is highlighted with a mouse cursor), and 'Changes'. Underneath this menu, there's a 'Create Process' button. A table lists a single process entry: 'deploy' under 'Process', 'Description' (JKE), and 'Actions' (Edit, Copy, Delete). At the bottom of the table, it says '1 record - Refresh Print' and shows a page number '1 / 1'.

5. In the Process column, click **deploy** to open the process.

The Process editor displays the series of steps that run when the *deploy* process is started.

6. In the second step of the process, *Update Property File*, click the **Edit** icon.



Observe that the field for **File includes** specifies one file, **JKEDB.properties**. In that file, this step searches for the string `jdbc.hostname=` and replaces the trailing value with the value of the IBM UrbanCode Deploy environment property called `jke.db.host`. In IBM UrbanCode Deploy, each environment can create a different value for this property. The ability to tie this environment property to a HOT document parameter allows the blueprint design server to reuse blueprints without hardcoding particular values.

Edit Properties

Name *	Update Property File
Custom Encoding	
Use System Encoding	<input type="checkbox"/>
Directory Offset	WEB-INF/classes
File Includes *	JKEDB.properties
File Excludes	
Add/Update properties	jdbc.hostname=\${p:environment/jke.db.host}
Additional properties	
Additional properties prefix	

7. Do not make any changes, and close the edit window.

- At the top of the page, click the **jke.war** link.

The screenshot shows the IBM UrbanCode Deploy web interface. At the top, there's a navigation bar with tabs: Dashboard, Components, Applications, Configuration, and Processes. Below the navigation bar, the URL path is displayed as Home > Components > jke.war > Processes > Process: deploy. The main content area has a title 'Process: deploy' with a hand cursor icon above it. Below the title, there's a table with one row labeled 'Version' and '1 of 1'. Navigation arrows are at the bottom of the table.

- Click the **Configuration** tab, and click **Environment Property Definitions**. Components can have component environment properties; these properties transfer to the environment when the component is deployed. An environment property overrides the value that is set on a component environment property that has the same name. For example, if you deploy a web application to three environments where each environment has the application server in a different location, you can specify this location in an environment property on each environment.

- Observe that the **Default Value** field is empty for the `jke.db.host` property. In the next exercise, you set this property in your blueprint.

The screenshot shows the IBM UrbanCode Deploy interface with the Configuration tab selected. On the left, there's a sidebar with links like Basic Settings, Component Properties, Environment Property Definitions (which is currently selected), Resource Property Definitions, Version Property Definitions, Configuration File Templates, and Version Import History. The main content area shows a table titled 'Environment Property Definitions' with a note: 'Define properties here to be given values on each environment the component is used in.' There's a 'Add Property' button. The table lists two properties: 'jke.db.host' and 'liberty.install.dir'. The 'jke.db.host' row has its 'Default Value' field highlighted with a red box. The table includes columns for Name, Label, Pattern, Required, Default Value, Description, and Actions (Edit, Delete). Pagination at the bottom shows '2 records - Refresh Print' and page number '1 / 1'.

Exercise 3

Update the environment property in the Blueprint Designer

Exercise 3: Update the environment property in the Blueprint Designer

Exercise 3: Update the environment property in the Blueprint Designer

In this exercise, you update the environment property in the Blueprint Designer.

To do this, you will:

- Update the environment property

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

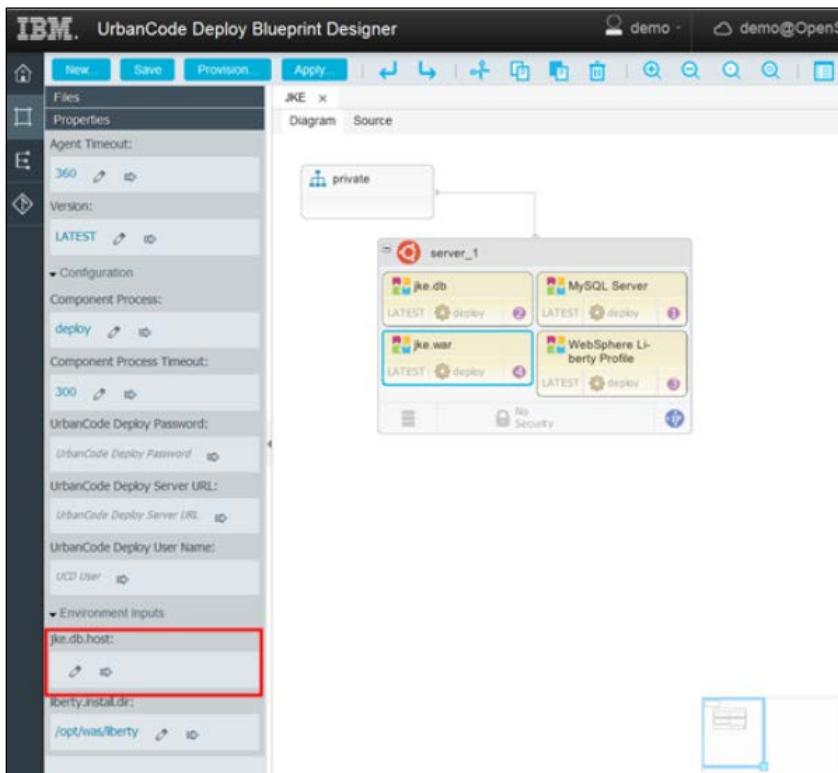
Exercise 3:

Tasks and results

Task 1. Update the environment property.

IBM UrbanCode Deploy is able to set properties that might vary depending on the environment in question. You can set these properties in the Blueprint Designer.

1. From the browser, switch to the **Blueprint Designer** tab. If necessary, open the JKE blueprint to edit.
2. In the server_1 image, click the **jke.war** component. Observe the properties in the panel to the left of the canvas and notice the values of the selected resource.
3. In the Properties panel, expand the **Environment Inputs** section. The property value for the **jke.db.host** field is empty. This property is the same one that you viewed in IBM UrbanCode Deploy.
4. Click the **Edit** icon , and then type localhost.



5. Save the blueprint.

Exercise 4

Provision the environment and observing the deployment

Exercise 4: Provision the environment and observing the deployment

Exercise 4: Provision the environment and observing the deployment

The blueprint design server provisions the cloud resources and creates an application environment in IBM UrbanCode Deploy. Then, the IBM UrbanCode Deploy server runs the component processes that are listed in the blueprint.

In this exercise, you provision blueprints to a cloud and examine the component processes in IBM UrbanCode Deploy. You also observe the actions associated with provisioning and deploying.

To do this, you will:

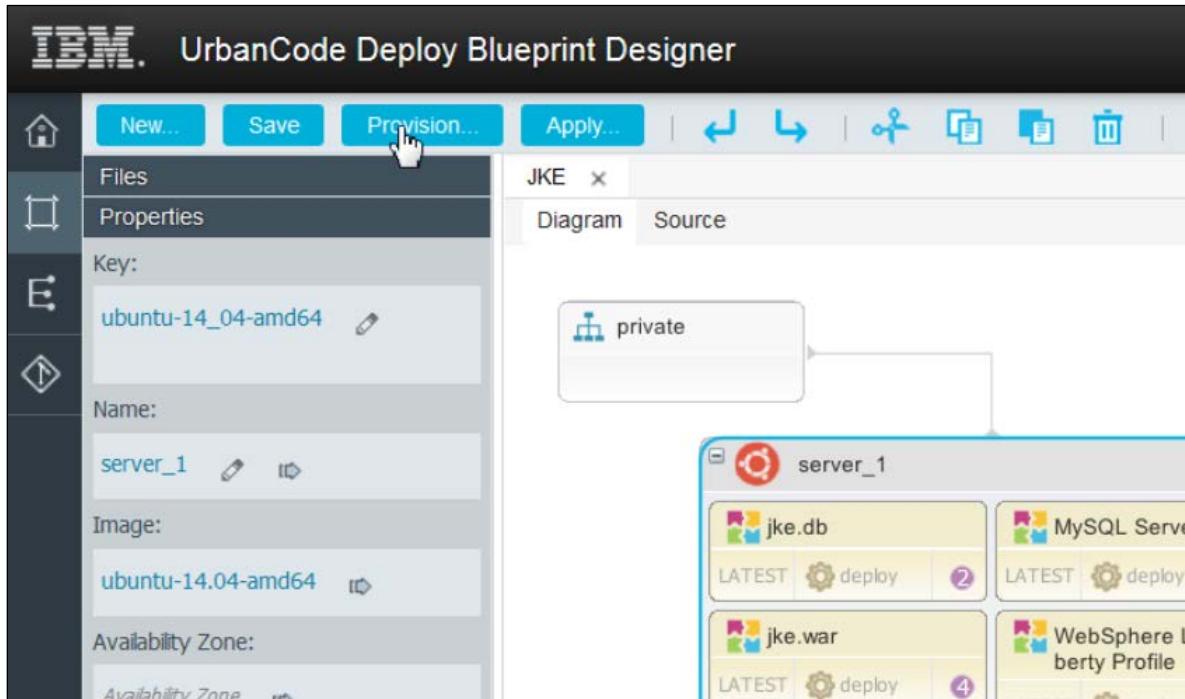
- Provision the blueprint.
- Observe the OpenStack provision.
- Observe the provision and deployment in IBM UrbanCode Deploy.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 4: Tasks and results

Task 1. Provision the blueprint.

1. To start the provision process, click **Provision**.



2. In the “Provision Blueprint to new Environment” window, do the following steps:
- In the **Environment Name** field, type JKE-OPENSTACK-UCD.
 - Click **Image Parameters**.
 - In the **Flavor** field, select **m1.medium**.
 - In the **Key name** field, select **demo-key**.
 - Click **Network Parameters**.
 - In the **Public Network ID** field, select **public**.

Provision Blueprint to new Environment

Environment Name *	JKE-OPENSTACK-UCD
Cloud Project	demo@OpenStack
Configuration	Select Configuration...
Set the parameter values for this environment	
Agent Parameters	
Image Parameters	
Network Parameters	
Availability Zone	nova
Network Id For Private	private
Public Network ID	public
Generated to reference subnet 'ccde0dff-635f-49c5- b26b-4bd4bef89dc1'.	
<input style="background-color: #0070C0; color: white; font-weight: bold; padding: 5px 10px; margin-right: 10px;" type="button" value="Provision"/> <input style="font-weight: bold; padding: 5px 10px;" type="button" value="Cancel"/>	

3. Click **Provision**.

4. Click the **Environments** tab.

Environment	Applied Blueprint	Cloud Project	Region	Last Update	Virtual Instances	Last Orchestration Event	Actions
JKE-OPENSTACK-UCD	JKE	demo@OpenStack	RegionOne	8/5/2016, 11:12 AM	100% Active	Stack CREATE started	

5. Expand the list of resources by clicking the twistie next to **JKE-OPENSTACK_UCD**. Observe the progress of the deployment.
6. In the Environment column, click **JKE-OPENSTACK-UCD** to see the resource details.

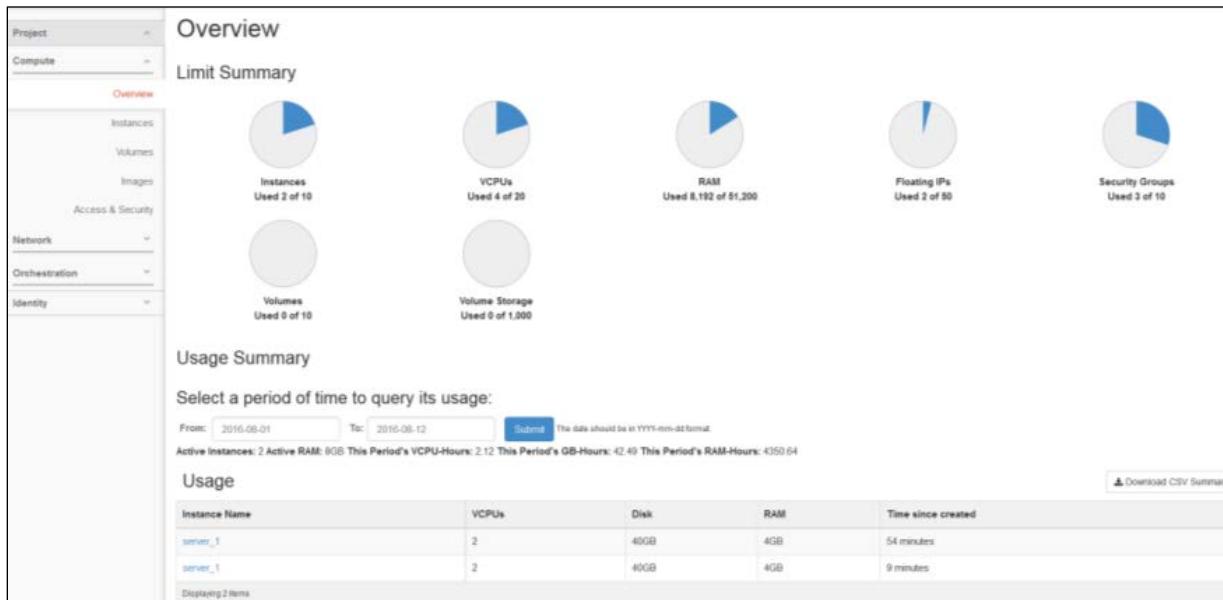
Resource Name	Type	Details	Status	Stack Status
resource_tree	IBM UrbanCode: ResourceTree	-	-	
ubuntu-14_04-amd64-heat-tools_to_private_port	OS: Neutron: Port	-	-	
jke_db_sw_config	IBM UrbanCode: SoftwareConfig: UCD	-	-	
WebSphere_Liberty_Profile_sw_config	IBM UrbanCode: SoftwareConfig: UCD	-	-	
ucd_agent_install_inox	OS: Heat: SoftwareConfig	-	-	
MySQL_Server_se_config	IBM UrbanCode: SoftwareConfig: UCD	-	-	
ubuntu-14_04-amd64-heat-tools_to_private_port_floating_ip	OS: Neutron: FloatingIP	-	-	
jke_war_sw_config	IBM UrbanCode: SoftwareConfig: UCD	-	-	
ubuntu-14_04-amd64-heat-tools_mime	OS: Heat: MultipartMime	-	-	
ubuntu-14_04-amd64-heat-tools	OS: Nova: Server	name = server_1, image = ubuntu-14_04-amd64-heat-tools, power_state = Build, ip_address =	BUILD	
jke_db	IBM UrbanCode: SoftwareDeploy: UCD	-	-	
WebSphere_Liberty_Profile	IBM UrbanCode: SoftwareDeploy: UCD	-	-	
jke_war	IBM UrbanCode: SoftwareDeploy: UCD	-	-	
MySQL_Server	IBM UrbanCode: SoftwareDeploy: UCD	-	-	

7. Click **Instances** to view the floating IP address that is assigned to the image.

Instance Name	IP Address	Size	Keypair	Status	Power State	Actions
server_1	10.0.0.6, 172.24.4.7	m1 medium	demo_key	ACTIVE	ACTIVE	

Task 2. Observe the OpenStack provision.

- From the browser, switch to the **OpenStack** tab to observe the image that is being created.
- Click **Overview** to see the allocated servers.



- Click **Instances** to see the image coming up and to see the IP addresses assigned to the system.

The screenshot shows the Bluebox Instances page. The sidebar on the left lists categories: Project, Compute, Network, Orchestration, and Identity. Under Compute, 'Instances' is selected. The main area is titled 'Instances' and displays a table of two server instances. The table columns are Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions. Both instances are named 'server_1', use the 'ubuntu-14.04-amd64-heat-tools' image, are in the 'm1.medium' size, have 'demo_key' as their key pair, are active, are in the 'nova' availability zone, and are running. Their IP addresses are 10.0.0.4 and 10.0.0.3 respectively. The 'Actions' column for each instance contains a 'Create Snapshot' button.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
server_1	ubuntu-14.04-amd64-heat-tools	10.0.0.4 Floating IPs: 172.24.4.7	m1.medium	demo_key	Active	nova	None	Running	13 minutes	Create Snapshot
server_1	ubuntu-14.04-amd64-heat-tools	10.0.0.3 Floating IPs: 172.24.4.5	m1.medium	demo_key	Active	nova	None	Running	1 hour	Create Snapshot

Task 3. Observe the provision and deployment in IBM UrbanCode Deploy.

1. In the browser, switch to the **IBM UrbanCode Deploy** tab to observe the environment.
2. Click **Applications**, and click **JKE**.

The screenshot shows the 'Applications' tab selected in the IBM UrbanCode Deploy interface. A single application entry is listed:

Name	Template	Description	Created	By
JKE		Sample JKE Banking application	7/29/2016, 3:13 PM	admin

At the bottom of the table, there is a row indicator '1 record' and a refresh/print button.

3. Click the twistie to the left of the created environment, **JKE-OPENSTACK-UCD**. Occasionally refresh the page until the **jke.war** component is displayed.
 4. Observe the components as they progress through the deployment steps.
- Note: Components are not displayed in the table until the blueprint design server is in the process of deploying them; so it might take a few moments for all of the components to display in the table.

5. In the Component column, click **jke.war** after it is displayed.

The screenshot shows the IBM UrbanCode Deploy interface. The top navigation bar includes 'Dashboard', 'Components', 'Applications', 'Configuration', and 'Processes'. Below this, there are links for 'Resources', 'Calendar', 'Work Items', 'Reports', and 'Settings'. The main content area is titled 'Application: JKE'. It displays 'Created By' (admin) and 'Created On' (7/29/2016, 3:13 PM). A 'Description' field contains 'Sample JKE Banking application'. Below this is a navigation bar with tabs: 'Environments', 'History', 'Configuration', 'Components' (which is selected), 'Blueprints', 'Snapshots', 'Processes', 'Calendar', and 'Changes'. A 'Create Environment' button is visible. A search bar allows searching by 'Name' or 'Blueprint'. A message says 'Drag environments by their names to re-order them.' Below this is a table titled 'JKE-OPENSTACK-UCD' with columns: Component, Version, Date Created, Snapshot, Properties, Status, Compliancy, and Actions. The table contains four rows: 'jke.war' (version 1.0, created 8/5/2016, 11:16 AM), 'WebSphere Liberty Profile' (version 8.5.5.5, created 8/5/2016, 11:15 AM), 'jke.db' (version 1.0, created 8/5/2016, 11:15 AM), and 'MySQL Server' (version 5.5.49, created 8/5/2016, 11:14 AM). All components are marked as 'Active' and 'Compliant (1/1)'. A green progress bar at the top right indicates 'Compliance 4 / 4'.

6. In the **Actions** column, click **View Request** and observe the steps as they complete.

The screenshot shows the IBM UrbanCode Deploy interface for the 'jke.war' component. The top navigation bar includes 'Dashboard', 'Usage', 'Configuration', 'Calendar', 'Versions', 'Processes', and 'Changes'. Below this is a 'Description' field: 'JKE Banking Sample Application Web Application Resource Archive'. A 'Used By' section shows 'JKE'. Below this is a table with columns: Resource, Version, Date, Status, and Actions. One row is listed: '/ Base Resource for environment JKE-OPENSTACK-UCD / JKE.JKE-OPENSTACK-UCD.10.0.0.6 / jke.war' (Version 1.0, Date 8/5/2016, 11:17 AM, Status Active). At the bottom left, it says '1 record - Refresh Print'. At the bottom right, there are pagination controls and a 'Rows' dropdown set to 20. The 'Actions' column for the resource row has a 'View Request' button, which is highlighted with a red box. Below this is a 'Component Request History' table with columns: Process, Resource, Version, Application, Environment, Scheduled For, By, Status, and Actions. One entry is shown: 'deploy / Base Resource for environment JKE-OPENSTACK-UCD / JKE.JKE-OPENSTACK-UCD.10.0.0.6 / jke.war' (Version 1.0, Application JKE, Environment JKE-OPENSTACK-UCD, Scheduled For 8/5/2016, 11:16 AM, By admin, Status Success). The 'Actions' column for this entry also has a 'View Request' button, which is highlighted with a red box. At the bottom left, it says '3 records - Refresh Print'. At the bottom right, there are pagination controls and a 'Rows' dropdown set to 20.

7. Point to **Update Property File** (the second step).

This step is the same one that you observed in the earlier exercise, which performs the value substitution.

8. Click the **Output Log** icon  that is displayed.

9. Observe the substitution of the value localhost for the jdbc.hostname property.

Output Log

Working Directory /opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/var/work/jke.war

```

1 =====
2 plugin: File Utils, id: com.urbancode.air.plugin.FileUtils, version: 46
3 plugin command: '/opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/opt/groovy-1.8.8/bin/groovy' '-c'
4 working directory: /opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/var/work/jke.war
5 properties:
6   PLUGIN_INPUT_PROPS=/opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/var/temp/logs612062491122171
7   PLUGIN_OUTPUT_PROPS=/opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/var/temp/logs61206249112217
8   additionalProps=
9   additionalPropsPrefix=
10  customEncoding=
11  deleteProps=
12  dir=WEB-INF/classes
13  excludes=
14  includes=JKEDB.properties
15  updateProps=jdbc.hostname=localhost
16  useSystemEncoding=false
17 environment:
18  AGENT_HOME=/opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4
19  AH_AUTH_TOKEN=*****
20  AH_WEB_URL=http://192.168.27.100:8080
21  AUTH_TOKEN=*****
22  DS_AUTH_TOKEN=*****
23  DS_SYSTEM_ENCODING=ANSI_X3.4-1968
24  JAVA_OPTS=-Dfile.encoding=ANSI_X3.4-1968 -Dconsole.encoding=ANSI_X3.4-1968
25  PLUGIN_HOME=/opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/var/plugins/com.urbancode.air.plugin
26  UD_DIALOGUE_ID=8be4e3d1-4c55-4a30-83e8-56ee76ad2a91
27  WE_ACTIVITY_ID=40af9f46-7a74-425d-bd1f-9ea6fc700053
28 =====
29 Using Input/Output Streams to read the properties file.
30 Working directory: /opt/ibm-ucd/JKE.JKE-OPENSTACK-UCD.10.0.0.4/var/work/jke.war/WEB-INF/class

```

Download Log 1 / 1

Exercise 5

Validate the application

Exercise 5: Validate the application

Exercise 5: Validate the application

Now that you have provisioned the environment and deployed applications to that environment, you can verify that the application is working by connecting to the external IP address.

In this exercise, you will:

- Validate the application.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

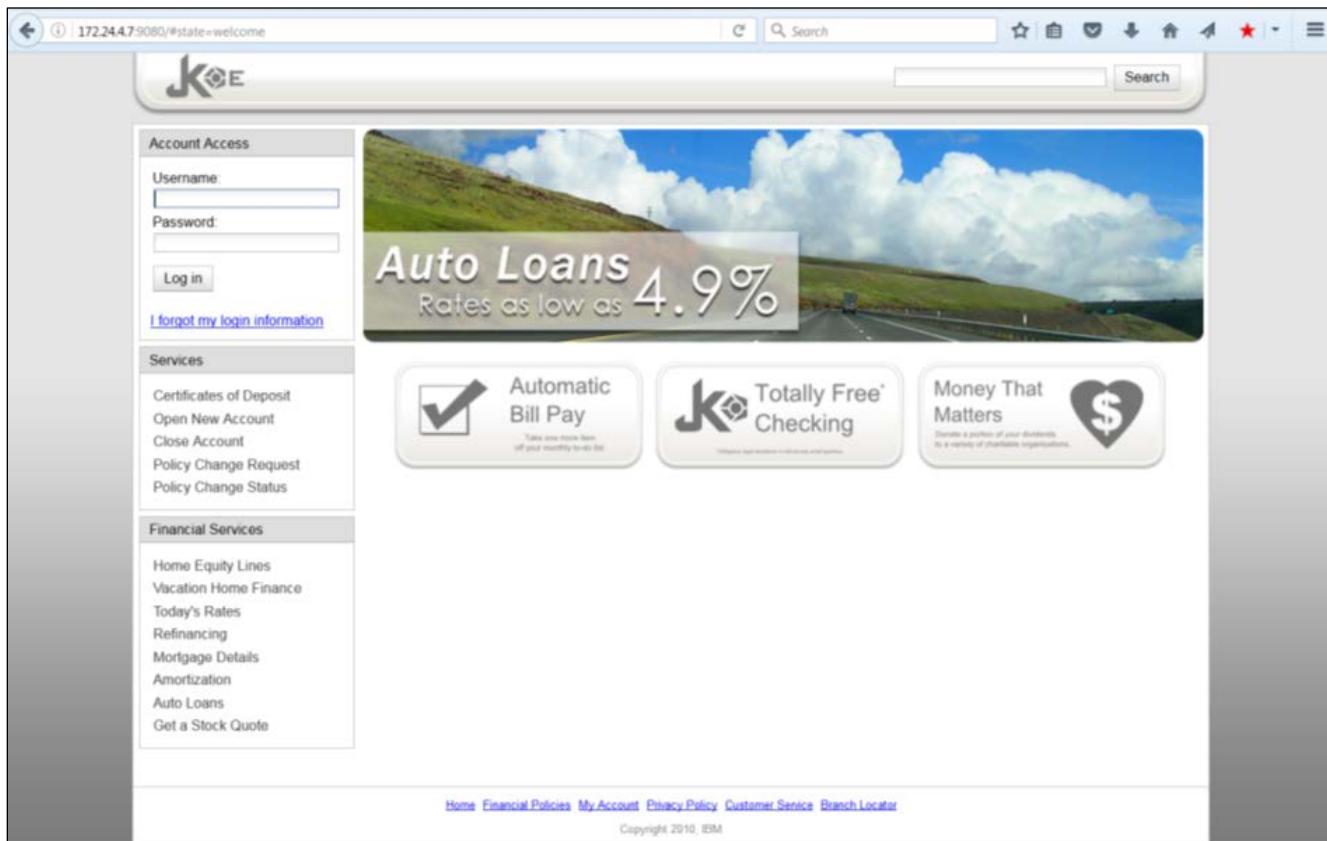
Exercise 5:

Tasks and results

Task 1. Validate the application.

1. Open a new tab in the browser.
2. Type the floating IP address that OpenStack assigned to your environment (172.24.4.x) and include port 9080. In this lab, the URL is <http://172.24.4.7:9080>.

When the JKE application home page opens, you know that the Blueprint Designer provisioned a full-application stack into the cloud.



3. In the Account Access section, log in to the JKE application by using the user name of **jbrown** and the password **jbrown**.
Observe that the user has balances. The balances confirm that the application is accessing the database.

Exercise 6

Deleting the cloud environment

Exercise 6: Delete the cloud environment

Exercise 6: Delete the cloud environment

You can delete provisioned cloud environments from the Blueprint Designer. When you delete a cloud environment, the virtual nodes and all of the data on them are deleted. In most cases, you cannot undo this action.

In this exercise, you will:

- Delete the cloud environment.

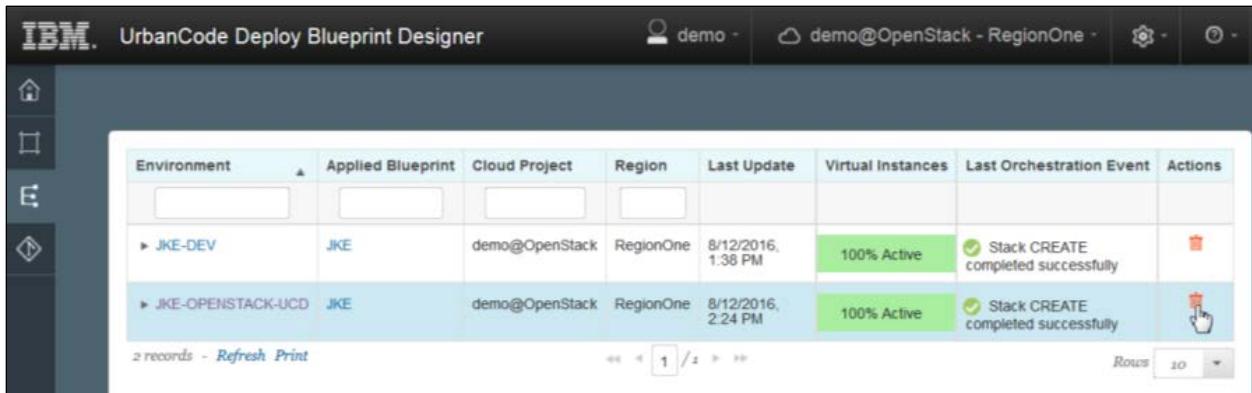
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 6:

Tasks and results

Task 1. Delete the cloud environment.

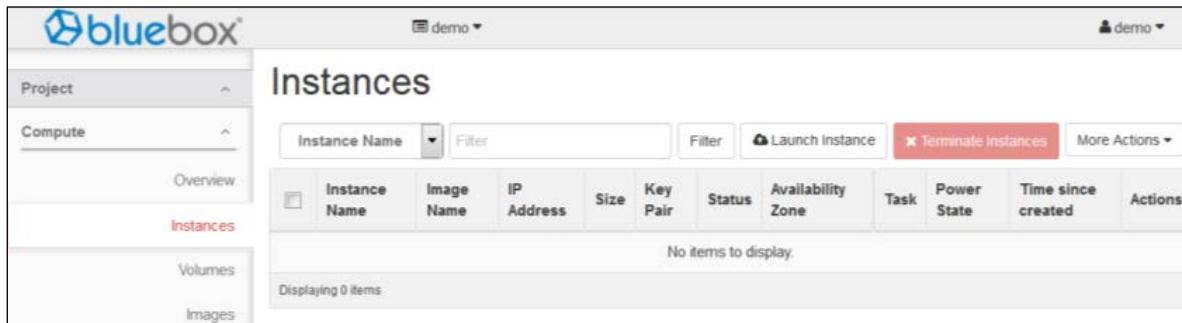
1. From the browser, switch to the **Blueprint Designer** tab.
2. Click the **Environments** tab.
3. Locate the environment you created, **JKE-OPENSTACK-UCD**, and click the **Delete** icon  under the Actions column.



Environment	Applied Blueprint	Cloud Project	Region	Last Update	Virtual Instances	Last Orchestration Event	Actions
JKE-DEV	JKE	demo@OpenStack	RegionOne	8/12/2016, 1:36 PM	100% Active	✓ Stack CREATE completed successfully	
JKE-OPENSTACK-UCD	JKE	demo@OpenStack	RegionOne	8/12/2016, 2:24 PM	100% Active	✓ Stack CREATE completed successfully	

You can also delete the environment from the first exercise. The instances on the cloud are deleted and the resources in the resource tree are deleted.

4. From the browser, click the **OpenStack** tab and verify that the environments are removed.

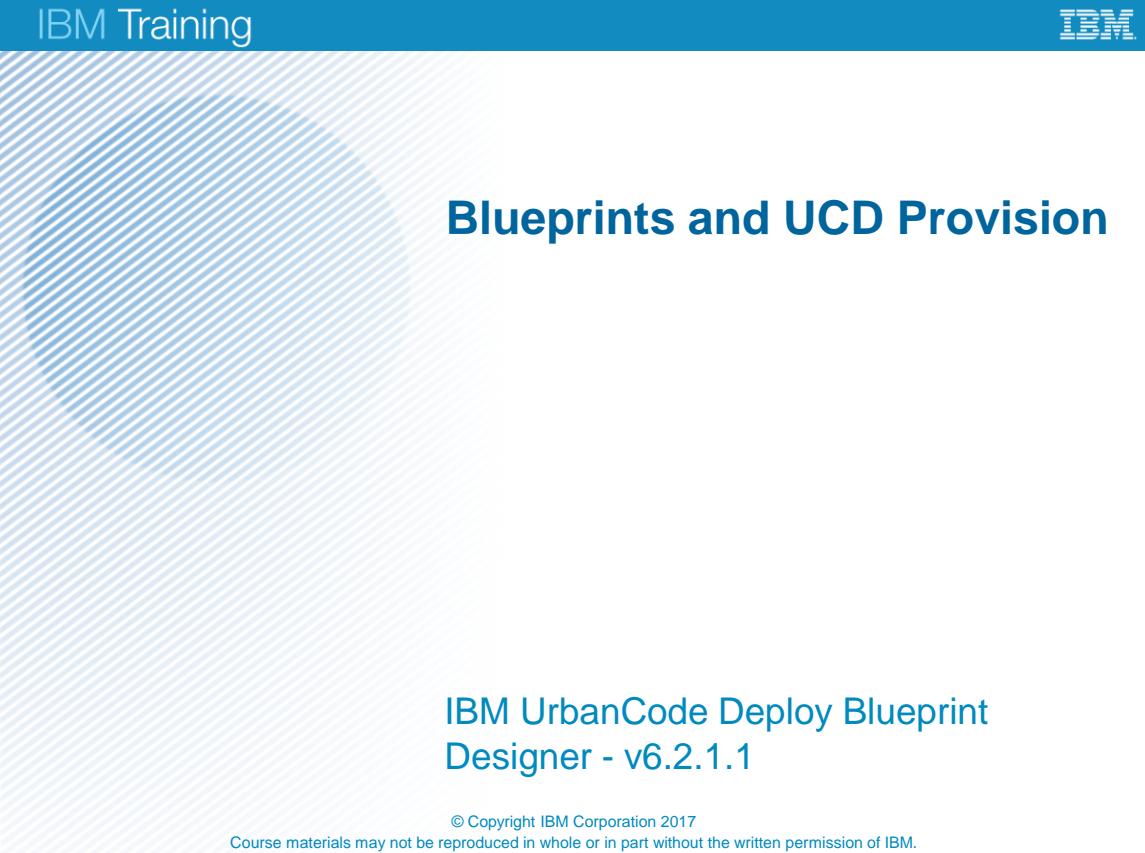


Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
No items to display.										
Displaying 0 items										

5. From the browser, click the tab with the JKE application, and refresh the browser to verify that the environment is removed.

In this unit, you provisioned your blueprint to the OpenStack cloud in your environment. Then, you tracked the progress of the environment provisioning in OpenStack, the component deployment to the environment, and the control of the environment properties. You connected to the newly provisioned environment and tested the application, which illustrated the ability of the Blueprint Designer server to provision a full-application stack into a cloud. Finally, you deleted cloud environments and verified that the resources were removed. You can re-create the environment later when it is needed by provisioning the blueprint again.

Unit 5 Blueprints and UCD Provision



The slide features a blue header bar with the text "IBM Training" on the left and the IBM logo on the right. The main content area has a light gray background with a subtle diagonal striped pattern. The title "Blueprints and UCD Provision" is centered at the top in a large, bold, dark blue font. Below the title, the subtitle "IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1" is displayed in a smaller, dark blue font. At the bottom of the slide, there is a thin horizontal footer bar containing the copyright notice: "© Copyright IBM Corporation 2017" and "Course materials may not be reproduced in whole or in part without the written permission of IBM."

Blueprints and UCD Provision

IBM UrbanCode Deploy Blueprint
Designer - v6.2.1.1

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

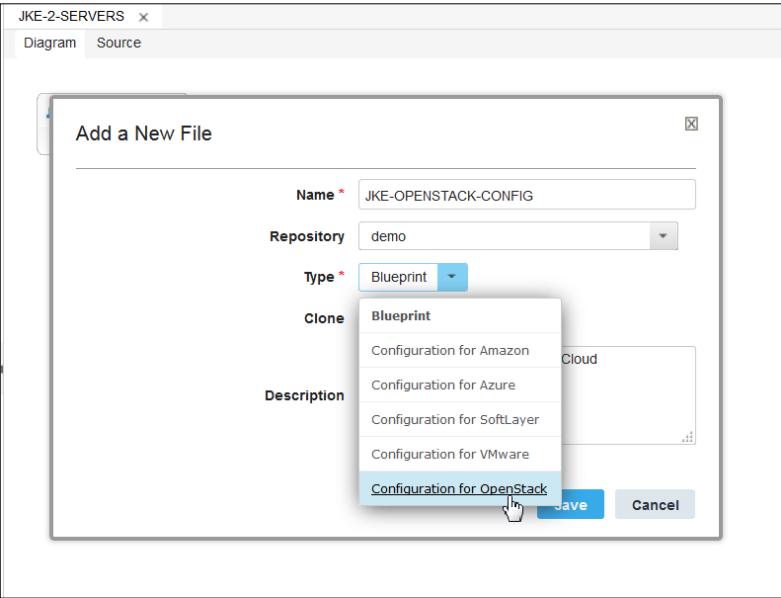
- Create a configuration file with values
- View the deployments in IBM UrbanCode Deploy and OpenStack
- Provision a blueprint from IBM UrbanCode Deploy

Unit objectives

You can divide the functional areas of the blueprint into separate servers, so you can assign the appropriate security controls. You also create a configuration file that contains the initial values for the input properties for the blueprint.

IBM Training IBM

Provisioning configuration files on different cloud systems



The screenshot shows the 'Blueprints and UCD Provision' section of the IBM Blueprint Designer. A modal dialog titled 'Add a New File' is open. Inside, there are fields for 'Name' (set to 'JKE-OPENSTACK-CONFIG'), 'Repository' (set to 'demo'), and 'Type' (set to 'Blueprint'). A dropdown menu under 'Type' lists several options: 'Blueprint', 'Configuration for Amazon', 'Configuration for Azure', 'Configuration for SoftLayer', 'Configuration for VMware', and 'Configuration for OpenStack'. The 'Configuration for OpenStack' option is currently selected. At the bottom of the dialog are 'Save' and 'Cancel' buttons.

Blueprints and UCD Provision © Copyright IBM Corporation 2017

Provisioning configuration files on different cloud systems

Configuration files are lists of properties and values. You can use configuration files to provision the same blueprint on multiple cloud systems. Configuration files apply only to blueprints that you create in the Blueprint Designer, but not for blueprints that you create with the IBM UrbanCode Deploy server.

You can specify these parameters in the blueprint, in a configuration file, or at provisioning time. To provide property values in blueprints, you can put the values in parameters and then externalize the parameters to a configuration file. Externalizing properties in this way is an important part of creating blueprints that can be provisioned on different cloud systems. You can keep the basic information in the blueprint and create configuration files for the specific details for different clouds or different situations.

The screenshot shows the UrbanCode Deploy Blueprint Designer application. The title bar says "IBM Training" and "IBM". The main window has a toolbar with "New", "Save", "Provision", "Apply", and other icons. Below the toolbar is a navigation bar with "demo" and "demo@OpenStack - RegionOne". The left sidebar shows a file tree with "Files" (containing "default", "demo" folder with "JKE", "JKE-2-SERVERS", "JKE2", and "Configurations" folder with "JKE-OPENSTACK-CONFIG"), "Applications" (with "Cloud Foundry" and "OpenShift" entries), and "Blueprints" (with "JKE-OPENSTACK-CONFIG"). The right pane shows a configuration file named "JKE-OPENSTACK-CONFIG" with the following content:

```

# Configuration file template
description: >
    Created 8/25/16 by demo. For Cloud demo@OpenStack
parameters:

```

A modal dialog titled "Choose Blueprint" is open in the center of the screen. It contains the instruction "Add types and parameters from the selected blueprint." and a dropdown menu labeled "Blueprint*" with the option "demo / JKE-2-SERVERS". At the bottom of the dialog are "OK" and "Cancel" buttons.

At the bottom of the application window, there are two copyright notices: "Blueprints and UCD Provision" on the left and "© Copyright IBM Corporation 2017" on the right.

Importing properties from existing blueprints

Each blueprint begins with a list of parameters. In most cases, you create a blueprint with a default set of properties. Then you use one or more configuration files to customize provisioning for specific clouds or specific situations. Because blueprints use only the properties in the configuration file that apply to it, you can also use a configuration file with more than one blueprint. You can also import properties from more than one blueprint. When you provision a blueprint, the blueprint uses only the properties in the configuration file that apply to it.

IBM Training IBM

Configuration files list parameters to provision a blueprint

The top screenshot shows the 'JKE-2-SERVERS' configuration file with the following code:

```

1 # Configuration file template
2
3 @description: >
4   Created 8/25/16 by demo. For Cloud demo@OpenStack
5
6 @parameters:
7   flavor: "T000"
8   key_name: "T000"
9   availability_zone: "nova"
10  network_id_for_private: "dec769"
11  subnet_id_ccde0dff-63f1-49c5-b26b
12  ucd_server_url: "http://192.168.7
13  ucd_user: "PasswordsAuthToken"
14  ucd_relay_url: "None"
15  public_network_id: "public_network"
16

```

The bottom screenshot shows the 'JDE-OPENSTACK-CONFIG' configuration file with the following code:

```

1 # Configuration file template
2
3 @description: >
4   Created 8/25/16 by demo. For Cloud demo@OpenStack
5
6 @parameters:
7   flavor: "x1-ec2large"
8   key_name: "T000"
9   availability_zone: "nova"
10  network_id_ccde0dff-63f1-49c5-b26b
11  subnet_id_ccde0dff-63f1-49c5-b26b
12  ucd_server_url: "http://192.168.7
13  ucd_user: "PasswordsAuthToken"
14  ucd_relay_url: "None"
15  public_network_id: "public_network"
16

```

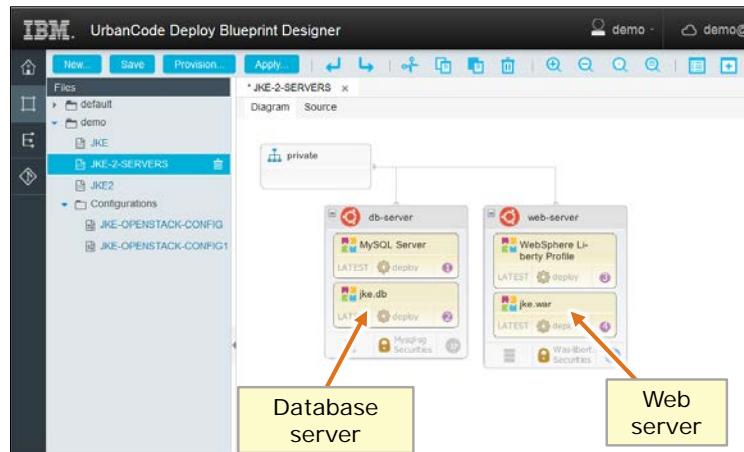
Blueprints and UCD Provision © Copyright IBM Corporation 2017

Configuration files list parameters to provision blueprint

The editor for configuration files is similar to the editor for blueprint source code. You can specify properties in a configuration file and substitute those properties when you provision the environment. For instance, the parameter "flavor" represents the size of a provisioned image, and the parameter "key_name" represents the security key for a provisioned image.

Now you can use this configuration file when you provision the environment. You can create multiple configuration files to customize blueprints for different situations or clouds. For example, suppose that your cloud systems provide different virtual images. Start by creating virtual images from one of the clouds in a blueprint, or use referenced images to represent generic images. Then externalize each of the properties in the virtual images. Finally, create a different configuration file for each cloud, import these properties, and specify values for a specific cloud system.

Separate functional areas for the web and database servers

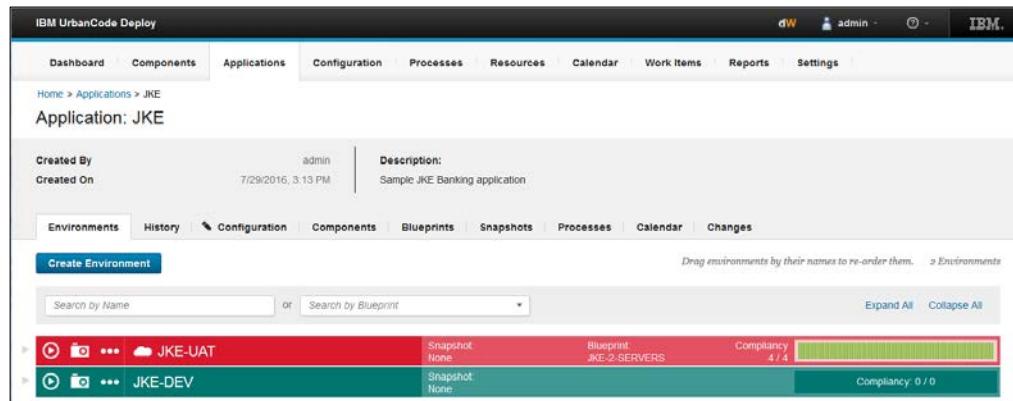


Separate functional areas for the web and database servers

This example shows the blueprint divided into two functional areas, the web server and the database server. This division is useful when you want to assign the appropriate security controls to each resource. In the next slides, you explore provisioning this blueprint from IBM UrbanCode Deploy.

IBM Training 

Environments created from blueprints



The screenshot shows the IBM UrbanCode Deploy application interface. The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Applications menu is currently selected. Below the navigation is a breadcrumb trail: Home > Applications > JKE. The main content area is titled "Application: JKE". It displays information about the application, such as "Created By: admin" and "Created On: 7/29/2016, 3:13 PM". A "Description:" field contains "Sample JKE Banking application". Below this, there is a navigation bar with tabs for Environments, History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. The "Environments" tab is selected. A search bar at the top of the environments list allows searching by name or blueprint. Two environments are listed: "JKE-UAT" and "JKE-DEV". Each environment entry includes icons for a cloud, a camera, and three dots, followed by the environment name, its current status (Snapshot: None), its blueprint (Blueprint: JKE-2-SERVERS), and its compliance level (Compliance: 4 / 4 for JKE-UAT, Compliance: 0 / 0 for JKE-DEV). A note at the bottom of the environments list says "Drag environments by their names to re-order them." and shows a count of "2 Environments". At the bottom of the page, there are copyright notices: "Blueprints and UCD Provision" and "© Copyright IBM Corporation 2017".

Environments created from blueprints

You can recognize environments generated from blueprints by the cloud icon before the environment name. Standard environments do not have this icon.

IBM Training IBM

Blueprints created in the Blueprint Designer are available in IBM UrbanCode Deploy

The screenshot shows the 'Create Environment' dialog box in the IBM UrbanCode Deploy interface. The 'Blueprint' dropdown is highlighted, showing a list of available blueprints. The 'Name' field is set to 'JKE-UAT'. The 'Color' section shows a color palette with 'Bright Cerulean (Blue)' selected. Other settings like 'Teams' and 'Exempt Processes' are also visible.

Blueprints and UCD Provision © Copyright IBM Corporation 2017

Blueprints created in the Blueprint Designer are available in IBM UrbanCode Deploy

After blueprints are created in the Blueprint Designer, they become automatically available in the IBM UrbanCode Deploy main user interface. When a user creates an environment (from **Applications > Environment**), the window includes a **Blueprint** field that contains the blueprints that were created in the Blueprint Designer. Those blueprints show a cloud icon here also, to distinguish them from the already supported ones.

Predefined properties and values to use

The screenshot shows two instances of the 'Create Environment' dialog side-by-side. Both instances have the following settings:

- Cloud Provisioning:**
 - Blueprint: JHE-2-SERVERS
 - Cloud Project: demo@OpenStack
 - Region: RegionOne
 - Blueprint Version: 1.0
 - Configuration: JHE-OPENSTACK-CONFIG (highlighted in the first instance)
 - Configuration Version: 1.0
- Set property values for nodes to be created for this environment:**

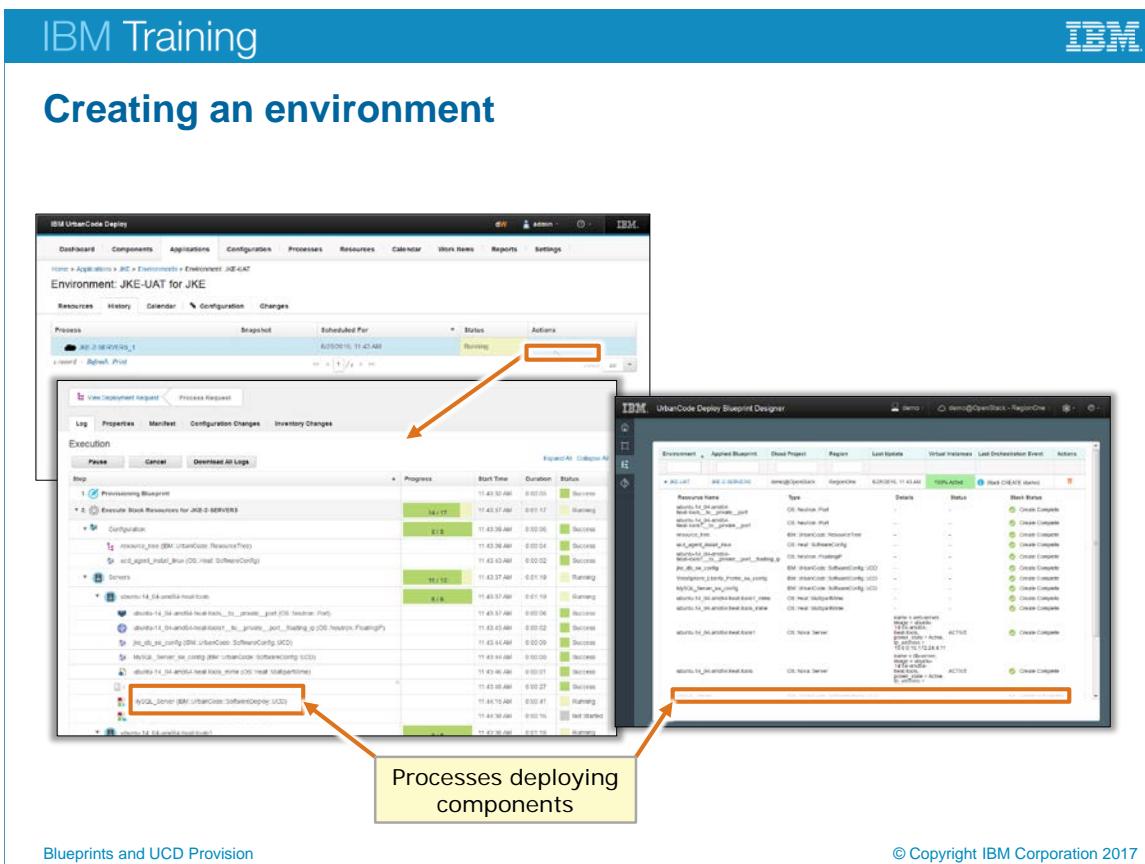
availability_zone	nova
flavor	
key_name	
network-id_for_private	private
public_network_id	public_network_id
subnet_id_code0dff435f48c5-b280-4d44e189d1	private >> private-subnet
ucd_relay_url	None
- Buttons:** Previous, Create, Cancel.

Blueprints and UCD Provision

© Copyright IBM Corporation 2017

Predefined properties and values to use

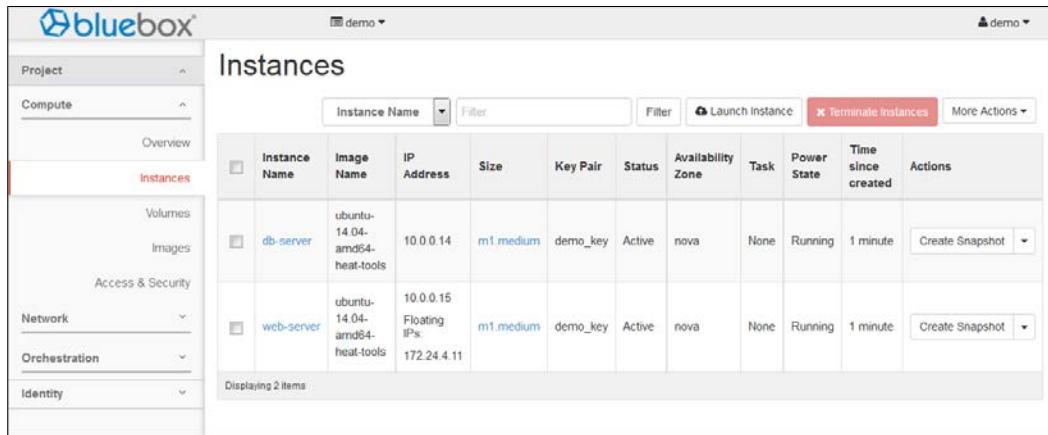
When you provision environments from the blueprint, the parameters are shown in the provisioning window. In that window, you can accept the values in the source code or specify new values. You can also specify a configuration file to use the values that are in that file.



Creating an environment

The creation of an environment from a blueprint not only instantiates the server and other infrastructure in the cloud, but it also deploys the components included in the blueprint. It is done by running the corresponding component deployment processes that are also included in the blueprint.

Instances of servers on OpenStack



The screenshot shows the bluebox OpenStack Instances dashboard. On the left, there's a sidebar with navigation links for Project, Compute (Overview, Instances), Volumes, Images, Access & Security, Network, Orchestration, and Identity. The main area is titled 'Instances' and displays a table with the following data:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
db-server	ubuntu-14.04-amd64-heat-tools	10.0.0.14	m1.medium	demo_key	Active	nova	None	Running	1 minute	<button>Create Snapshot</button>
web-server	ubuntu-14.04-amd64-heat-tools	10.0.0.15 Floating IP: 172.24.4.11	m1.medium	demo_key	Active	nova	None	Running	1 minute	<button>Create Snapshot</button>

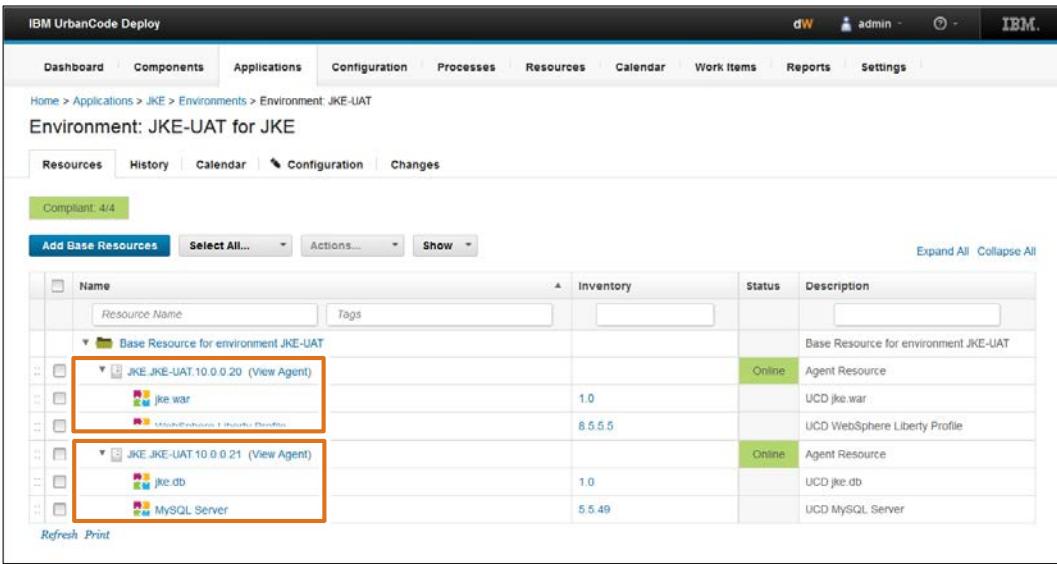
Displaying 2 items

Instances of servers on OpenStack

OpenStack shows two server instances: one for the database server and one for the web server.

IBM Training IBM

Agent processes are started separately for each resource



The screenshot shows the IBM UrbanCode Deploy interface. The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The user is logged in as 'admin'. The main content area displays the 'Environment: JKE-UAT for JKE' page. Under the 'Resources' tab, there is a table listing deployed components. Two specific rows are highlighted with orange boxes:

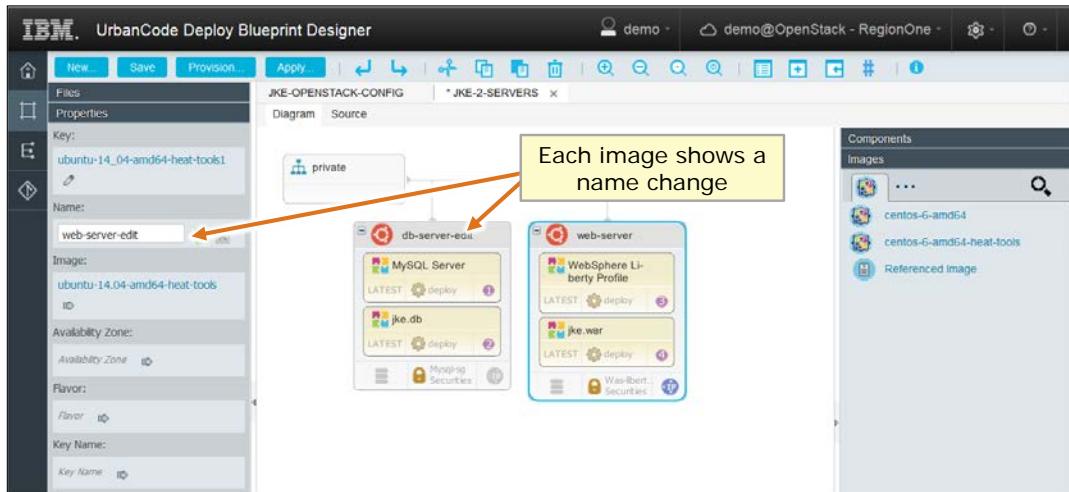
Name	Inventory	Status	Description
JKE JKE-UAT.10.0.0.20 (View Agent)	1.0	Online	Base Resource for environment JKE-UAT Agent Resource UCD jke.war UCD WebSphere Liberty Profile
JKE JKE-UAT.10.0.0.21 (View Agent)	1.0	Online	Agent Resource UCD jke.db UCD MySQL Server

Blueprints and UCD Provision © Copyright IBM Corporation 2017

Agent processes are started separately for each resource

IBM UrbanCode Deploy shows the deployed components, each with its own agent.

Updating environment in Blueprint Designer



Updating environment in Blueprint Designer

You can update an environment by updating the blueprints in the Blueprint Designer, such as adding a component or a virtual machine. Then you run the application process in IBM UrbanCode Deploy to deploy the new versions. When you update an environment in this way, the things that you can change depend on what parts of an environment the target cloud can update without reprovisioning the environment.

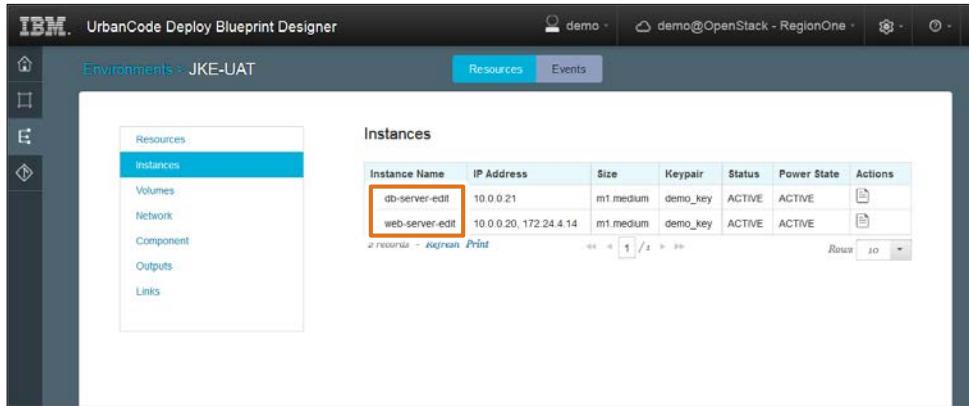
This example shows a simple update to the image names where *edit* was added to the names.

The screenshot shows the IBM UrbanCode Deploy interface. On the left, the 'Run Process on JKE-UAT' dialog is open, showing a dropdown menu with 'JKE' and 'JKE-2-SERVERS_1'. An orange arrow points from this dropdown to the 'Blueprint Properties' dialog on the right. The 'Blueprint Properties' dialog displays various configuration settings for a cloud provisioning project named 'JKE-2-SERVERS'. The 'Cloud Project' is set to 'demo@openstack', 'Region' to 'RegionOne', and 'Blueprint Version' to '1.0'. The 'Configuration' dropdown shows 'E-OPENSTACK-CONFIG'. Below these, there are fields for setting properties for nodes, including 'availability_zone', 'flavor', 'key_name', 'network_id_for_private', 'public_network_id', and 'subnet_id_ccce40ff-42d1-4e0b-b780-4bd4cafb9b01'. The 'wcd_relay_wif' field is also present. At the bottom of the dialog is an 'OK' button.

Blueprints can update a running environment

Changes are applied to the environment (from which the blueprint was created) by invoking a *Run Process* and selecting the blueprint from the process drop-down list. By treating blueprints as application processes in IBM UrbanCode Deploy, you can evolve the model captured in a blueprint by updating the environments with the changed blueprints by running the blueprint as an application process on that environment.

Instance names are updated in the Blueprint Designer



The screenshot shows the UrbanCode Deploy Blueprint Designer interface. The left sidebar has a 'Resources' section with 'Instances' selected, which is highlighted with a blue background. The main area is titled 'Instances' and contains a table with two rows. The table columns are 'Instance Name', 'IP Address', 'Size', 'Keypair', 'Status', 'Power State', and 'Actions'. The first row has 'db-server-edit' in the 'Instance Name' column, which is highlighted with a red box. The second row has 'web-server-edit' in the 'Instance Name' column. Both rows show '10.0.0.21' in the 'IP Address' column and 'm1 medium' in the 'Size' column. Both rows also have 'demo_key' in the 'Keypair' column, 'ACTIVE' in the 'Status' column, and 'ACTIVE' in the 'Power State' column. The 'Actions' column contains edit and delete icons. At the bottom of the table, there are buttons for '2 records', 'Z尋找', 'Print', and a page number '1 / 1'. There are also 'Rows' and a dropdown menu.

Instance Name	IP Address	Size	Keypair	Status	Power State	Actions
db-server-edit	10.0.0.21	m1 medium	demo_key	ACTIVE	ACTIVE	
web-server-edit	10.0.0.20, 172.24.4.14	m1 medium	demo_key	ACTIVE	ACTIVE	

Blueprints and UCD Provision

© Copyright IBM Corporation 2017

Instances name are updated in the Blueprint Designer

The Blueprint Designer shows the updates to the names.

Unit summary

- Create a configuration file with values
- View the deployments in IBM UrbanCode Deploy and OpenStack
- Provision a blueprint from IBM UrbanCode Deploy

Unit summary

The configuration file defines the cloud-specific properties that qualify a blueprint when provisioned into a particular cloud. Blueprints that are created with the Blueprint Designer are automatically available from IBM UrbanCode Deploy. Environments that are generated from blueprints act as application processes. Blueprints can update a running environment.

Exercises: Modeling a two-server blueprint and provisioning from IBM UrbanCode Deploy

- Create a blueprint with the separate functions for the web server and database server
- Create a configuration file to capture the blueprint parameter for reuse
- Provision a blueprint from IBM UrbanCode Deploy with the configuration file

Exercises: Modeling a two-server blueprint and provisioning form the IBM UrbanCode Deploy

In these exercises, you create a blueprint with two servers, so you can assign each with separate security controls. You also create a configuration file that contains the parameters required to provision a new environment. Finally, you use that configuration file to provision an environment directly from IBM UrbanCode Deploy.

Exercise 1

Create a blueprint with two servers

Exercise 1: Create a blueprint with two servers

Exercise 1: Create a blueprint with two servers

In this exercise, you create a blueprint that contains two servers. The functional areas of the blueprint, the web server and the database server, are separated into the two servers, so you can assign the appropriate security controls. You also create a configuration file that contains the initial values for the input properties for the blueprint.

To do this, you will:

- Create a blueprint.
- Create the database server.
- Create the web server.
- Define the deployment order.
- Define a floating IP to the web server.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 1:

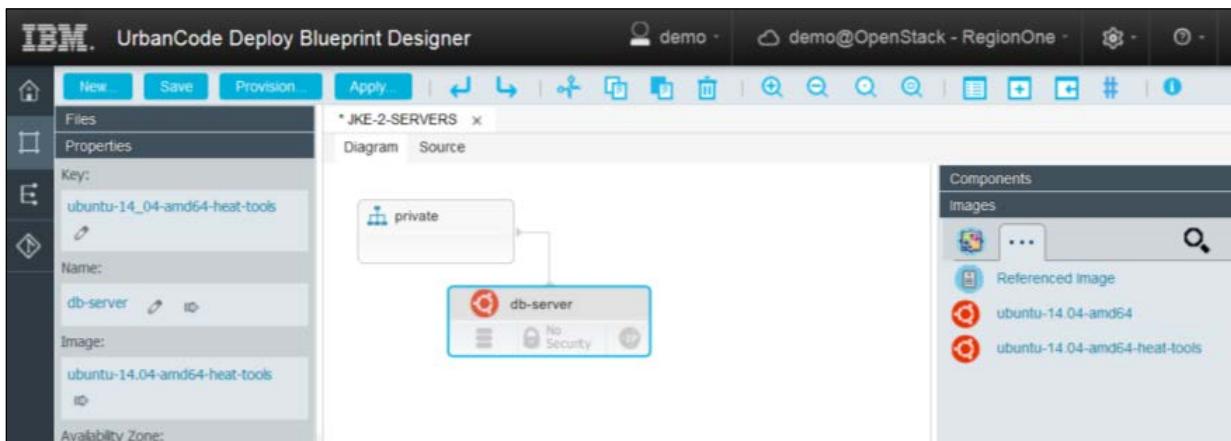
Tasks and results

Task 1. Create a blueprint.

1. From the browser, switch to the **Blueprint Designer** tab.
2. From the main menu, click **Blueprints**.
3. In the Files panel, ensure the demo folder is selected, and then click **New**.
4. Specify the following information for the new blueprint (Heat stack):
 - In the **Name** field, type **JKE-2-SERVERS**.
 - In the **Repository** field, select **demo**.
 - In the **Type** field, select **Blueprint**.
5. Click **Save**.

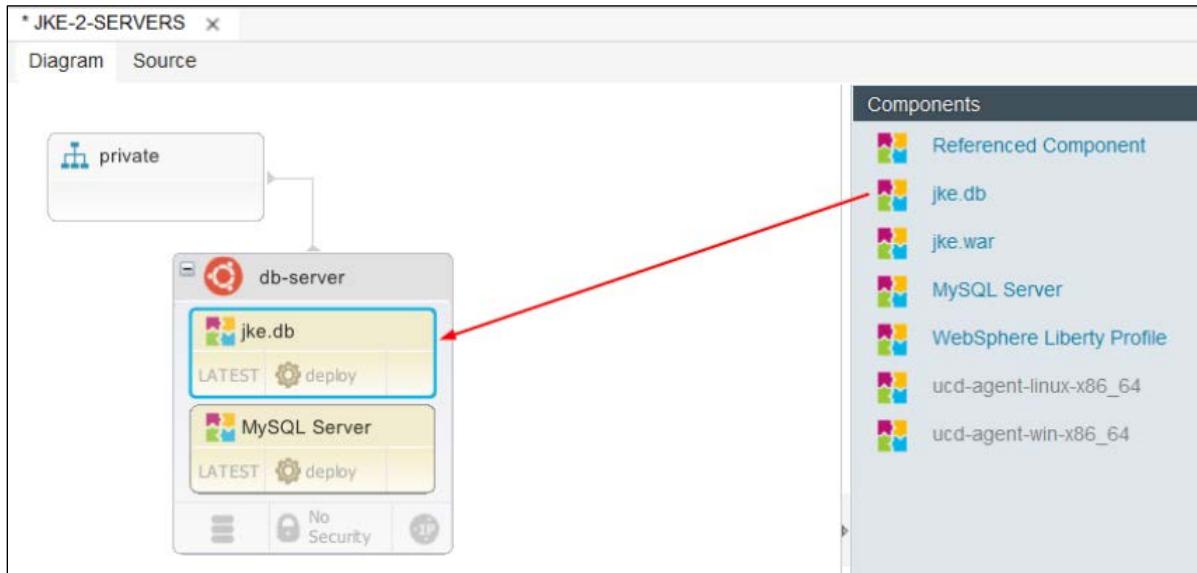
Task 2. Create the database server.

1. From the Networks drawer, drag the **private** network on to the canvas.
2. From the Images drawer, drag the **Ubuntu-14.04-amd64-heat-tools** image directly on to the private network.
3. In the Properties panel, edit the **Name** field of the image, type **db-server**, and press Enter.

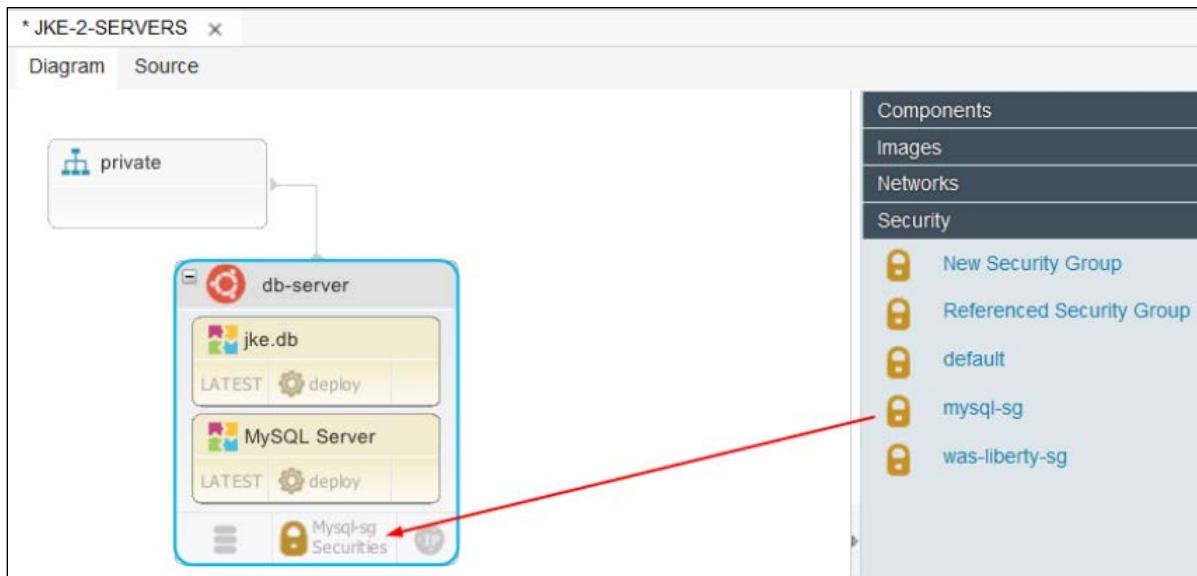


4. From the Components drawer, drag the **MySQL Server** component on to the db-server image, select **JKE** as the application, and click **OK**. Click **CHOOSE**, select **deploy** as the IBM UrbanCode Deploy component process, and click **OK**.

5. From the Components drawer, drag the **jke.db** component on to the db-server image.



6. Click the **db-server** image.
 7. From the Security drawer on the right, drag the **mysql-sg** security group definition on to the db-server.

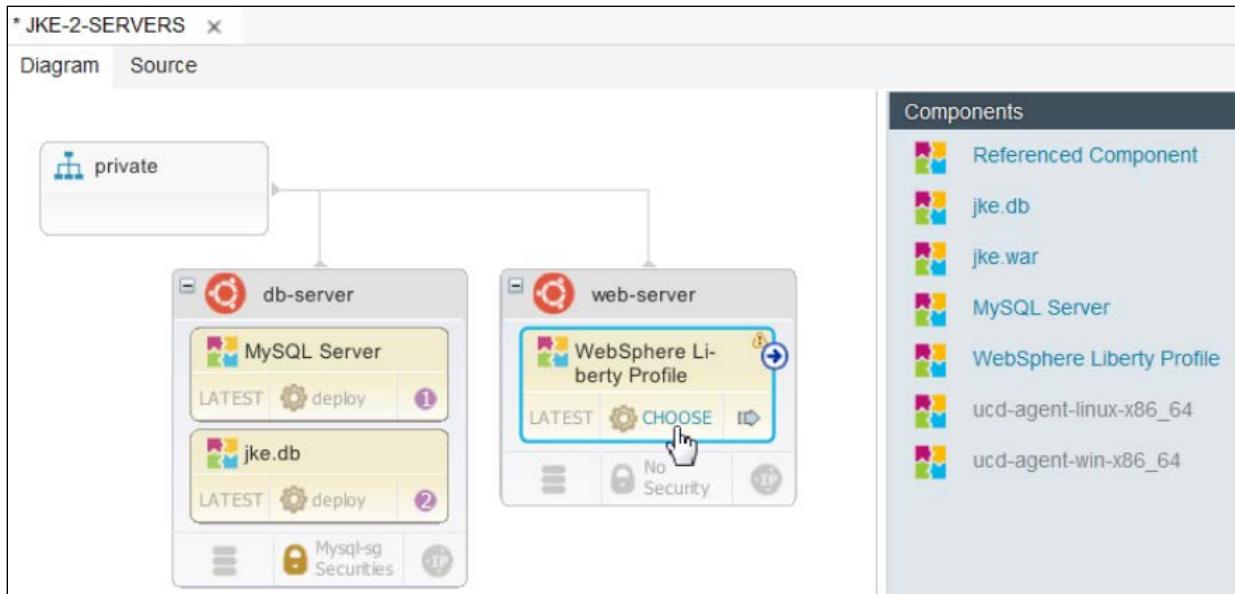


The integration with the cloud provider enables the blueprint to specify the security policies that are set for the provisioned server.

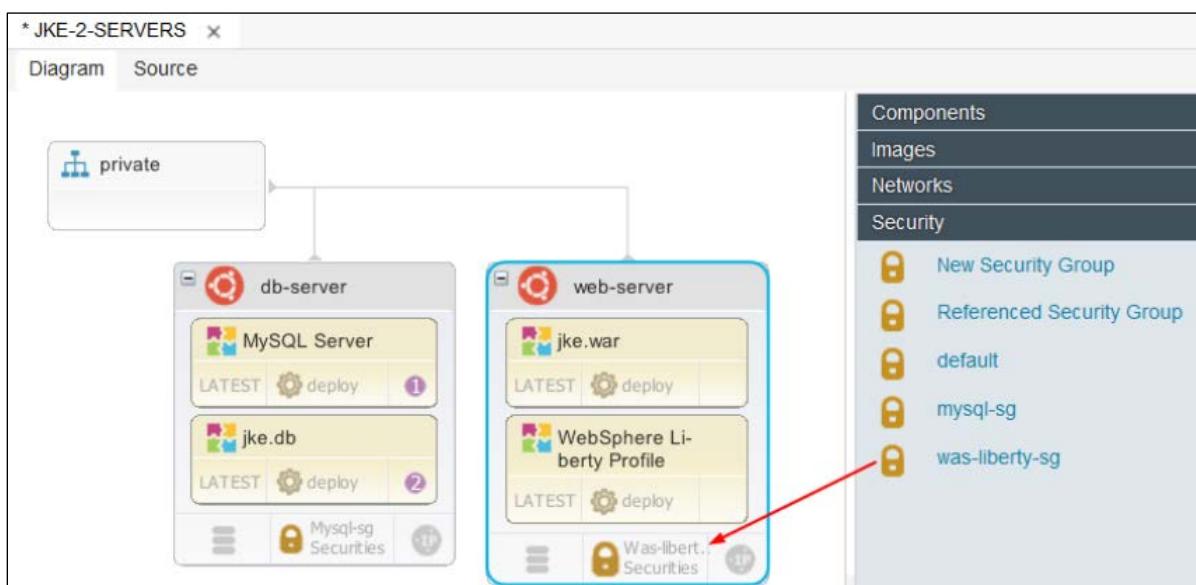
Note: The security policy for a database server opens port 3306 for MySQL access and port 22 for SSH access.

Task 3. Create the web server.

1. Drag another copy of the Ubuntu image on to the **private** network.
2. Change the name of this image to **web-server**.
3. From the Components drawer, drag the **WebSphere Liberty Profile**.
4. Click **CHOOSE**, select **deploy** as the IBM UrbanCode Deploy component process, and click **OK**.



5. Drag the **jke.war** component onto the web-server image.
6. Open the Security drawer and drag the **was-liberty-sg** security group definition on to the web-server.



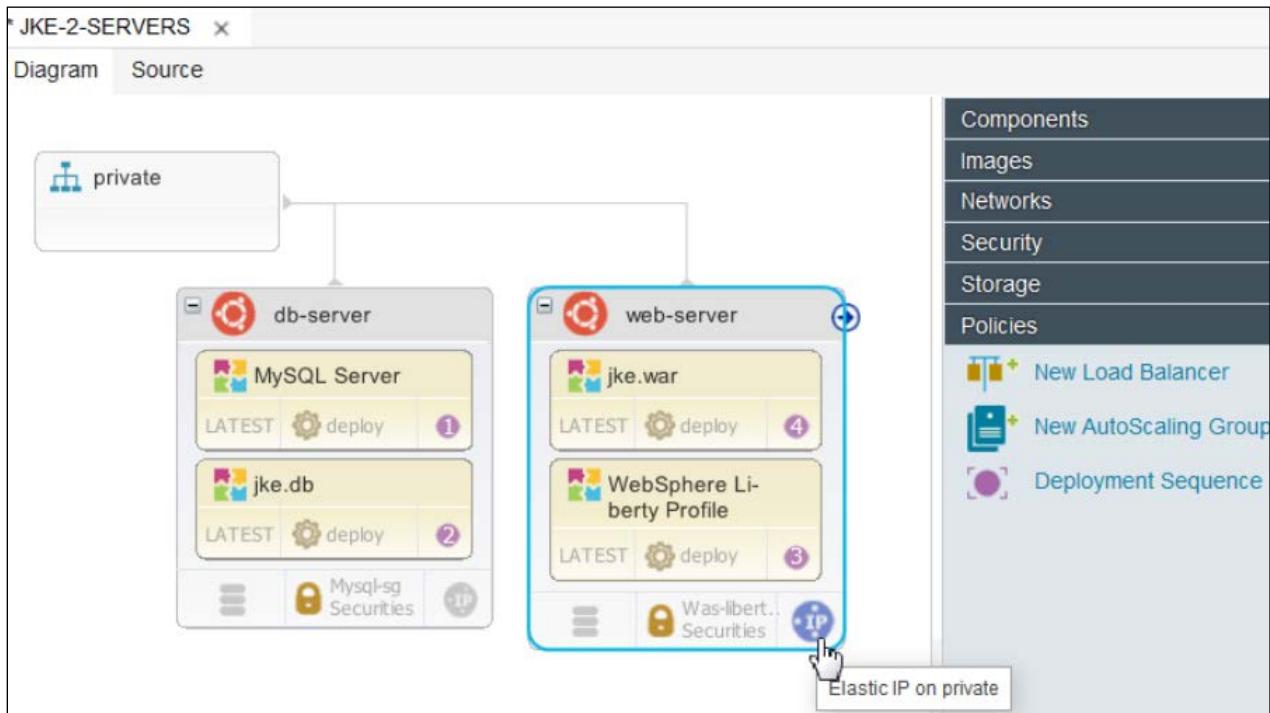
The security policy for web servers opens ports 80, 443, 9080, and 9443 for http(s) traffic and port 22 for SSH traffic.

Task 4. Define the deployment order.

1. Open the **Policies** drawer.
2. Assign the deployment sequence of the components by dragging the **Deployment Sequence** resource to each of the components in this order: MySQL Server, jke.db, WebSphere Liberty Profile, and jke.war.

Task 5. Assign a floating IP to the web server.

1. Click the **Floating IP** icon  on web-server image.



2. Save the blueprint.

Exercise 2

Assign the IP address to the database server

Exercise 2: Assign the IP address to the database server

Exercise 2: Assign the IP address to the database server

In this exercise, you assign the IP address to the database server.

To do this, you will:

- Assign the IP address to the database server.

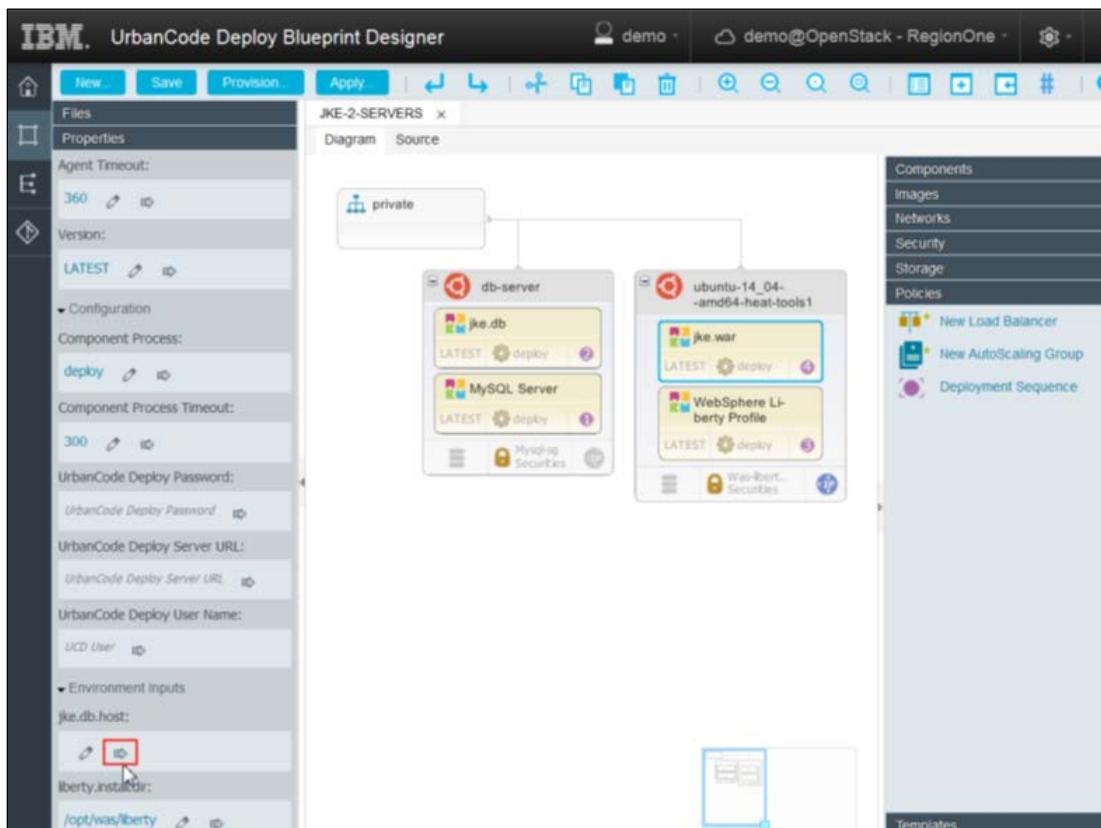
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 2:

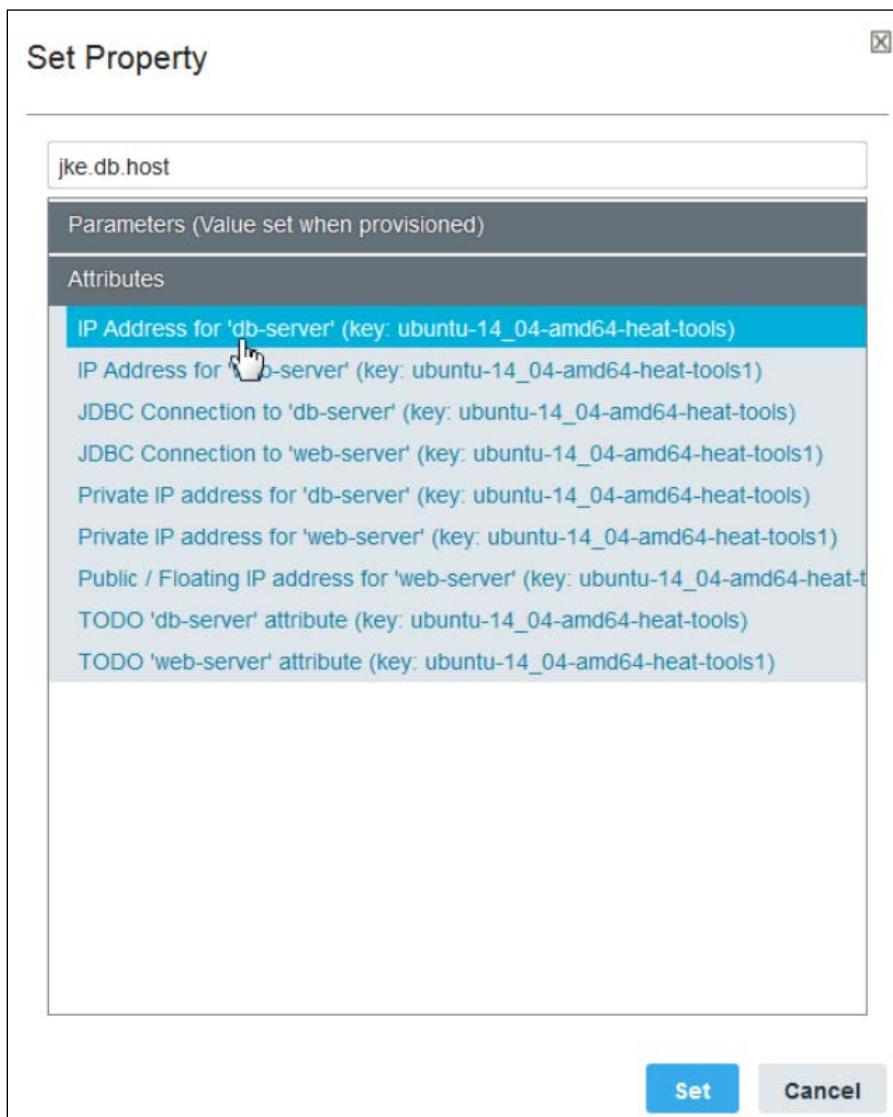
Tasks and results

Task 1. Assign the IP address to the database server.

1. Click the **jke.war** component on the **Diagram** tab.
2. In the Properties panel, expand the **Environment Inputs** section.
3. In the **jke.db.host** field, click the **Set Property** icon .



4. Click **Attributes** to expand the section, click **IP Address for db-server**, and click **Set**.



After this action, the orchestration engine can get an attribute associated with a resource in the blueprint and assign it to another resource. For this blueprint, you have instructed the orchestration engine to assign the IP address of the database server to the property `jke.db.host`. Until the blueprint is provisioned, you do not know what the IP address of the database server will be. Most likely, it will be different each time you provision this blueprint.

5. Save the blueprint.
6. Click the **Source** tab of the blueprint.
7. On the left side of the page, open the **Resources** drawer.
8. Expand **Images**, and expand **web-server**.

9. Click the second **jke.war** resource. Observe the entry in the blueprint that is getting this attribute.

```

IBM. UrbanCode Deploy Blueprint Designer
New Save Provision Apply
Files Outline Parameters Resources Diagram Source
JKE-2-SERVERS ×
Properties:
apply_config: { get_resource: jke_war_sw_config }
server: ubuntu-14_04-amd64-heat-tools1
version: LATEST
agent_timeout: "360"
jke_war_sw_config:
type: IBM::UrbanCode::SoftwareConfig::UCD
properties:
name: "jke.war"
component_process: "deploy"
component_process_timeout: "300"
ucd_server_url: { get_param: ucd_server_url } # these should come from resource_tree
ucd_username: { get_param: ucd_user }
ucd_password: { get_param: ucd_password }
application: { get_attr: [resource_tree, application_name] }
environment_name: { get_attr: [resource_tree, environment_name] }
inputs: # component's environment property definitions
jke.db.host: { get_attr: [ubuntu-14_04-amd64-heat-tools, first_address] } # IP Address for 'db-server'
liberty.install.dir : "/opt/was/liberty"
ubuntu-14_04-amd64-heat-tools1_mime:
type: OS::Heat::MultipartMime
properties:
parts:
- config: { get_resource: ucd_agent_install_linux }
subType: "x-shellscript"

```

Exercise 3

Save parameters to a configuration file

Exercise 3: Save parameters to a configuration file

Exercise 3: Save parameters to a configuration file

Configuration files are lists of properties and values. You can use configuration files to provision the same blueprint on multiple cloud systems.

In most cases, you create a blueprint with a default set of properties. Then, you use one or more configuration files to customize provisioning for specific clouds or specific situations. Because blueprints use only the properties in the configuration file that apply to it, you can also use a configuration file with more than one blueprint.

To do this, you will:

- Create the configuration file.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 3:

Tasks and results

Task 1. Create the configuration file.

1. At the top of the page, click **New** to create a file.
2. In the **Name** field, type **JKE-OPENSTACK-CONFIG**.
3. Verify that the **demo** is selected in the **Repository** field.
4. In the **Type** field, select **Configuration for OpenStack**.
5. Click **Save**.

Add a New File

Name *	JKE-OPENSTACK-CONFIG
Repository	demo
Type *	Configuration for OpenStack
Clone	Select Clone ...
Description	Created 8/5/16 by demo. For Cloud demo@OpenStack
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

IBM. UrbanCode Deploy Blueprint Designer

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. The top navigation bar includes 'demo', 'demo@OpenStack - RegionOne', and a settings gear icon. The main window has a toolbar with icons for New, Save, Provision, Apply, and various file operations. On the left is a sidebar with 'Files' and a tree view showing 'JKE-2-SERVERS' containing 'default', 'demo', 'JKE', and 'Configurations'. Under 'Configurations', 'JKE-OPENSTACK-CONFIG' is selected and highlighted in blue. The central workspace displays the configuration file content:

```

# Configuration file template
description: >
    Created 8/12/16 by demo. For Cloud demo@OpenStack
parameters:

```

A red box highlights the '+' icon in the toolbar, which typically represents a 'New' or 'Create' function.

6. To assign a value to the flavor parameter, click the word **TODO** to the right of flavor parameter, press Ctrl+Space bar at the same time, and select **m1.medium** from the options.

```

JKE-2-SERVERS * JKE-OPENSTACK-CONFIG x
Configuration
1 # Configuration file template
2
3 @description: >
4   Created 8/12/16 by demo. For Cloud demo@OpenStack
5
6 @parameters:
7   flavor: "TODO"
8   key_name: "TO m1.large"
9   availability_zone: "m1.medium"
10  network-id_for_private: "dec76954-f367-4de8-bba7-5b8a7992612d"
11  ucd_server_url: "http://192.168.27.100:8080"
12  ucd_user: "ad m1.tiny m1.medium"
13  ucd_relay_url: "m1.xlarge"
14  public_network_id: "public_network_id"
15

```

7. Assign values in the same way to the following parameters:

- **key_name: demo_key**
- **public_network_id: public**

Observe that the actual value placed in the configuration file is the UUID of the public network rather than the label **public**.

```

JKE-2-SERVERS * JKE-OPENSTACK-CONFIG x
Configuration
1 # Configuration file template
2
3 @description: >
4   Created 8/12/16 by demo. For Cloud demo@OpenStack
5
6 @parameters:
7   flavor: "m1.medium"
8   key_name: "demo_key"
9   availability_zone: "nova"
10  network-id_for_private: "dec76954-f367-4de8-bba7-5b8a7992612d"
11  ucd_server_url: "http://192.168.27.100:8080"
12  ucd_user: "admin"
13  ucd_relay_url: "None"
14  public_network_id: "4a4f7b4a-596b-4396-a7ad-efdcf26aa367" # public
15

```

8. Save the file.

Now you can use this configuration file when you provision the environment. You can create multiple configuration files to customize blueprints for different situations or clouds.

Exercise 4

Provision the two-server blueprint

Exercise 4: Provision the two-server blueprint

Exercise 4: Provision the two-server blueprint

In this exercise, you provision the two-server blueprint.

To do this, you will:

- Provision the two server blueprint.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 4: Tasks and results

Task 1. Provision the two server blueprint.

1. From the JKE-2-SERVERS blueprint, click **Provision**.
2. In the “Provision Blueprint to new Environment” window, do the following steps:
 - In the **Environment Name** field, type **JKE-QA**.
 - In the **Configuration** field, select **JKE_OPENSTACK_CONFIG**.

Note that the parameter fields now have values that are prepopulated.

The screenshot shows the 'Provision Blueprint to new Environment' dialog box. It contains the following fields:

- Environment Name:** JKE-QA
- Cloud Project:** demo@OpenStack
- Configuration:** JKE-OPENSTACK-CONFIG
- Additional Configuration:** Select Configuration...
- Flavor:** m1.medium
- Key Name:** demo_key

Below these fields, there is a section titled "Set the parameter values for this environment" with links for "Agent Parameters", "Image Parameters", and "Network Parameters". At the bottom right of the dialog are "Provision" and "Cancel" buttons.

- Click **Provision**.

3. In the browser, switch to the **OpenStack** tab, and click **Instances** to observe the deployment.

You can now see the two servers and the more descriptive names provided. Note that only one server has an IP address on the public network.

The screenshot shows the Bluebox OpenStack Instances dashboard. The left sidebar includes sections for Project, Compute (Overview, Instances), Volumes, Images, Access & Security (Network, Orchestration, Identity). The main area is titled 'Instances' with a table header: Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, Actions. Two instances are listed:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
db-server	ubuntu-14.04-amd64-heat-tools	10.0.0.5	m1.medium	demo_key	Active	nova	None	Running	0 minutes	<button>Create Snapshot</button>
web-server	ubuntu-14.04-amd64-heat-tools	10.0.0.6 Floating IPs: 172.24.4.6	m1.medium	demo_key	Active	nova	None	Running	0 minutes	<button>Create Snapshot</button>

Displaying 2 items

4. When the blueprint has completed provisioning, test the application by opening a new tab in the browser and typing the address for the web server with a port of 9080. In this example, the URL is <http://172.24.4.6:9080>.
5. Log in with the user name of **jbrown** and a password of **jbrown**.
6. Observe that the JKE application opens and the user has balances. These balances confirm that the application is accessing the database.

Exercise 5

Provision a blueprint from IBM UrbanCode Deploy

Exercise 5: Provision a blueprint from IBM UrbanCode Deploy

Exercise 5: Provision a blueprint from IBM UrbanCode Deploy

Although it is useful and appropriate for some users to provision blueprints the Blueprint Designer, other users, such as developers, might prefer IBM UrbanCode Deploy. In this lab, you provision a new cloud environment from IBM UrbanCode Deploy.

To do this, you will:

- Create the environment.
- Observe the provisioned environment.
- Verify the application.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 5:

Tasks and results

Task 1. Create an environment.

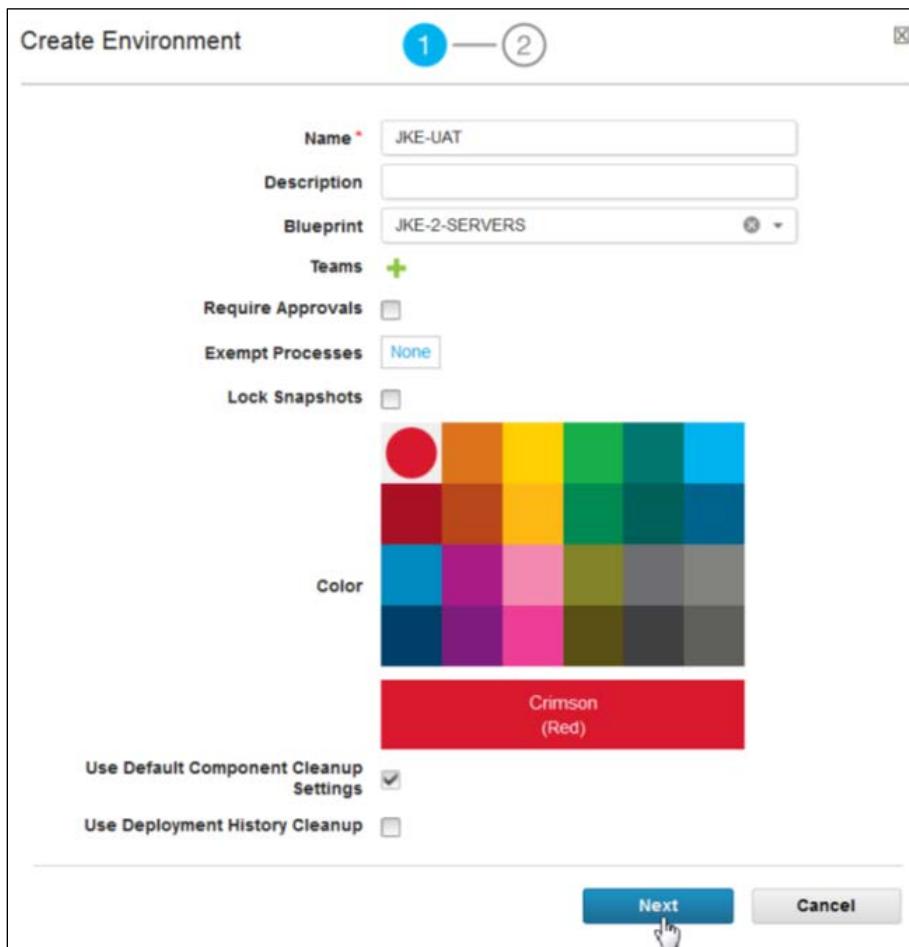
- From the browser, click the **UrbanCode Deploy** tab. If necessary, type admin for the user ID and admin for the password.
- From the top navigation, click **Applications**, and in the Name column, click **JKE**.

The screenshot shows the UrbanCode Deploy interface. The top navigation bar includes tabs for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Applications tab is selected. Below the navigation is a breadcrumb trail: Home > Applications. Under the Applications tab, there are buttons for Create Application, Import Applications, Actions..., and Flat list. A table lists applications, with one record named "JKE" selected. The table columns are Name, Template, Description, Created, and By. The JKE entry has "Sample JKE Banking application" in the Description field, was created on 7/29/2016 at 3:13 PM by admin, and has a status of "1 record". Buttons for Refresh and Print are at the bottom left, and a Rows dropdown is at the bottom right.

- On the **Environments** tab, click **Create Environment**.

The screenshot shows the UrbanCode Deploy interface with the Applications tab selected. The breadcrumb trail shows the path: Applications > JKE > Environment: JKE. The main area displays the JKE application details: Name: JKE, Type: Web, Admin: admin, Created: 7/29/2016, 3:13 PM, and Description: Sample JKE Banking application. Below this, the Environments tab is selected, showing a table with one row for "JKE-2-SERVERS". The table columns are Environment, Snapshot, Blueprint, and Actions. The "JKE-2-SERVERS" row has a Snapshot of "None" and a Blueprint of "JKE-2-SERVERS". A search bar at the top allows searching by Name or Blueprint. A "Drag and Drop" placeholder is visible above the table.

4. On the first page of the Create Environment window, do the following steps:
- In the **Name** field, type JKE-UAT as the environment name.
 - In the **Blueprint** field, select **JKE-2-SERVERS**.
 - Choose a color for this environment.
 - Click **Next**.



5. On the second page of the Create Environment window, do the following steps:
- Accept the default values the for **Cloud Project** and **Region** fields.
 - In the **Blueprint Version** field, select **1.0**. Note that the property values are populated from the configuration file.
 - In the **Configuration** field, select **JKE-OPENSTACK-CONFIG**.
 - In the **Configuration Version** field, select **1.0**.
 - Click **Create**.

Create Environment ① — ② ✖

Cloud Provisioning

Blueprint **JKE-2-SERVERS**

Cloud Project * **demo@OpenStack**

Region * **RegionOne**

Blueprint Version * **1.0**

Configuration **E-OPENSTACK-CONFIG**

Configuration Version * **1.0**

Set property values for nodes to be created for this environment

availability_zone **nova**

flavor **m1.medium**

key_name **demo_key**

network-id_for_private **private**

public_network_id **public**

subnet_id_ccde0dff-635f-49c5-b26b-4bd4bef89dc1 **private >> private-subnet**

ucd_relay_url **None**

Previous **Create** **Cancel**

The screenshot shows the 'Create Environment' dialog box. The title bar indicates it's the second step (②). The 'Cloud Provisioning' section includes fields for Blueprint (set to JKE-2-SERVERS), Cloud Project (set to demo@OpenStack), Region (set to RegionOne), Blueprint Version (set to 1.0), Configuration (set to E-OPENSTACK-CONFIG), and Configuration Version (set to 1.0). Below this, a section titled 'Set property values for nodes to be created for this environment' contains dropdown menus for availability_zone (nova), flavor (m1.medium), key_name (demo_key), network-id_for_private (private), public_network_id (public), and subnet_id_ccde0dff-635f-49c5-b26b-4bd4bef89dc1 (private >> private-subnet). The ucd_relay_url field is set to None. At the bottom, there are 'Previous', 'Create', and 'Cancel' buttons.

Task 2. Observe the provisioned environment.

- To see the status of the provisioned environment, in the Actions column, click **View Request**.

Process	Snapshot	Scheduled For	Status	Actions
JKE-2-SERVERS		8/15/2016, 8:29 AM	Running	View Request

- Expand the steps to see the progress of the deployment.

Step	Progress	Start Time	Duration	Status
1. Provisioning Blueprint		8:29:14 AM	0:00:05	Success
2. Execute Stack Resources for JKE-2-SERVERS	14 / 17	8:29:19 AM	0:01:51	Running
Configuration	2 / 2	8:29:19 AM	0:00:07	Success
resource_tree (IBM_UrbanCode_ResourceTree)		8:29:19 AM	0:00:05	Success
ucd_agent_install_linux (OS_Heat_SoftwareConfig)		8:29:24 AM	0:00:02	Success
Servers	11 / 13	8:29:22 AM	0:01:49	Running
ubuntu-14_04-amd64-heat-tools1	5 / 5	8:29:22 AM	0:01:49	Running
ubuntu-14_04-amd64-heat-tools1_to_private_port (OS_Neutron_Port)		8:29:22 AM	0:00:02	Success
WebSphere_Liberty_Profile_sw_config (IBM_UrbanCode_SoftwareConfig_UCD)		8:29:24 AM	0:00:00	Success
ubuntu-14_04-amd64-heat-tools1_mime (OS_Heat_MultipartMime)		8:29:26 AM	0:00:01	Success
ubuntu-14_04-amd64-heat-tools1 (OS_Nova_Server)		8:29:27 AM	0:00:27	Success
jke_war_sw_config (IBM_UrbanCode_SoftwareConfig_UCD)		8:29:54 AM	0:00:00	Success
WebSphere_Liberty_Profile (IBM_UrbanCode_SoftwareDeploy_UCD)		8:31:05 AM	0:00:06	Not Started
jke_war (IBM_UrbanCode_SoftwareDeploy_UCD)		8:31:05 AM	0:00:06	Not Started
ubuntu-14_04-amd64-heat-tools	6 / 6	8:29:22 AM	0:01:49	Running
ubuntu-14_04-amd64-heat-tools_to_private_port (OS_Neutron_Port)		8:29:22 AM	0:00:02	Success
jke_db_sw_config (IBM_UrbanCode_SoftwareConfig_UCD)		8:29:24 AM	0:00:01	Success

3. In the Blueprint Designer, click the **Environments** tab, and expand **JKE-UAT** to see the progress.

Environment	Applied Blueprint	Cloud Project	Region	Last Update	Virtual Instances	Last Orchestration Event	Actions
JKE-QA	JKE-2-SERVERS	demo@OpenStack	RegionOne	8/12/2016, 4:11 PM	100% Active	Stack CREATE completed successfully	✖
JKE-UAT	JKE-2-SERVERS	demo@OpenStack	RegionOne	8/15/2016, 8:29 AM	100% Active	Stack CREATE started	✖

Resource Name	Type	Details	Status	Stack Status
resource_tree	IBM:UrbanCode:ResourceTree	-	-	✓ Create Complete
ubuntu-14_04-amd64-heat-tools1_to_private_port	OS:Neutron:Port	-	-	✓ Create Complete
ubuntu-14_04-amd64-heat-tools1_to_private_port	OS:Neutron:Port	-	-	✓ Create Complete
jke_db_sw_config	IBM:UrbanCode:SoftwareConfig:UCD	-	-	✓ Create Complete
WebSphere_Liberty_Profile_sw_config	IBM:UrbanCode:SoftwareConfig:UCD	-	-	✓ Create Complete
ucd_agent_install_linux	OS:Heat:SoftwareConfig	-	-	✓ Create Complete
MySQL_Server_sw_config	IBM:UrbanCode:SoftwareConfig:UCD	-	-	✓ Create Complete
ubuntu-14_04-amd64-heat-tools1_to_private_port_floating_ip	OS:Neutron:FloatingIP	-	-	✓ Create Complete
ubuntu-14_04-amd64-heat-tools_mime	OS:Heat:MultipartMime	-	-	✓ Create Complete
ubuntu-14_04-amd64-heat-tools1_mime	OS:Heat:MultipartMime	-	-	✓ Create Complete
ubuntu-14_04-amd64-heat-tools1	OS:Nova:Server	name = web-server, image = ubuntu-14_04-amd64-heat-tools, power_state = Active, ip_address = 10.0.0.7, floating_ip = 172.24.4.7 name = db-server, image = ubuntu-14_04-amd64-	ACTIVE	✓ Create Complete

4. In OpenStack, click the **Instances** tab to see the progress of the images. Make note of the floating IP (172.24.4.7) for the web-server instance.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
db-server	ubuntu-14.04-amd64-heat-tools	10.0.0.8	m1.medium	demo_key	Active	nova	None	Running	5 minutes	Create Snapshot
web-server	ubuntu-14.04-amd64-heat-tools	10.0.0.7	Floating IP: 172.24.4.7	m1.medium	demo_key	Active	nova	Running	5 minutes	Create Snapshot
db-server	ubuntu-14.04-amd64-heat-tools	10.0.0.5	m1.medium	demo_key	Active	nova	None	Running	2 days, 16 hours	Create Snapshot
web-server	ubuntu-14.04-amd64-heat-tools	10.0.0.6	Floating IP: 172.24.4.6	m1.medium	demo_key	Active	nova	Running	2 days, 16 hours	Create Snapshot

5. To see the deployed components, in IBM UrbanCode Deploy, click **Applications**, and click **JKE**.

The screenshot shows the UrbanCode Deploy interface. At the top, there's a navigation bar with tabs: Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Applications tab is selected. Below the navigation bar, the title 'JKE' is displayed. On the left, there's a sidebar with sections for Overview, History, Configuration, Components, Snapshots, Snapshots, Processes, Calendar, and Changes. The Overview section shows a file icon, the size '128.000.1.71.1 mb', and a description 'Sample JKE Banking application'. The History section is currently selected and displays a table of deployment events. The table has columns for Deployment ID, Deployment Name, Snapshot Name, Status, and Details. One entry is highlighted in red: 'JKE-UAT' with status 'Completed (100%)'. The table also lists other entries like 'JKE-SERVERS' and several 'Snapshot' entries. The bottom of the screen shows a footer with links for Support, Documentation, and Contact.

Task 3. Verify the application.

Ensure that the application is deployed successfully by connecting to the web server IP.

1. Open a new browser and type the URL `http://172.24.4.7/9080`.
2. In the Account Access section, log in to the JKE application by typing the user name of `jbrown` and the password `jbrown`.

3. Observe that the user has balances to confirm that the application is accessing the database.

The screenshot shows a web browser window with the URL 172.24.4.9:9080/#state=accountSummary. The page features a navigation menu on the left with sections for Account Access, Services, and Financial Services. The Account Access section includes links for Logout, Money That Matters, Dividend Contribution, Certificates of Deposit, Open New Account, Close Account, Policy Change Request, and Policy Change Status. The Services section includes Home Equity Lines and Vacation Home Finance. The Financial Services section is currently selected. The main content area displays a promotional banner for "Auto Loans" with the text "Rates as low as 4.9%" over a background image of a road through a green hillside under a blue sky with clouds. Below the banner, the "Accounts Overview" section is shown with a table:

Account Type	Available Balance
Checking Account	\$2,000.00
Savings Account	\$12,500.00
IRA Retirement Account	\$25.00
Money Market Savings Account	\$500.00

Exercise 6

Update a running environment

Exercise 6: Update a running environment

Exercise 6: Update a running environment

When you create a blueprint in the blueprint editor and add components, the blueprint appears on the IBM UrbanCode Deploy server as an application process. You can run this process on an existing environment to update the environment to match the blueprint.

To do this, you will:

- Change the environment.
- Run the modified blueprint.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

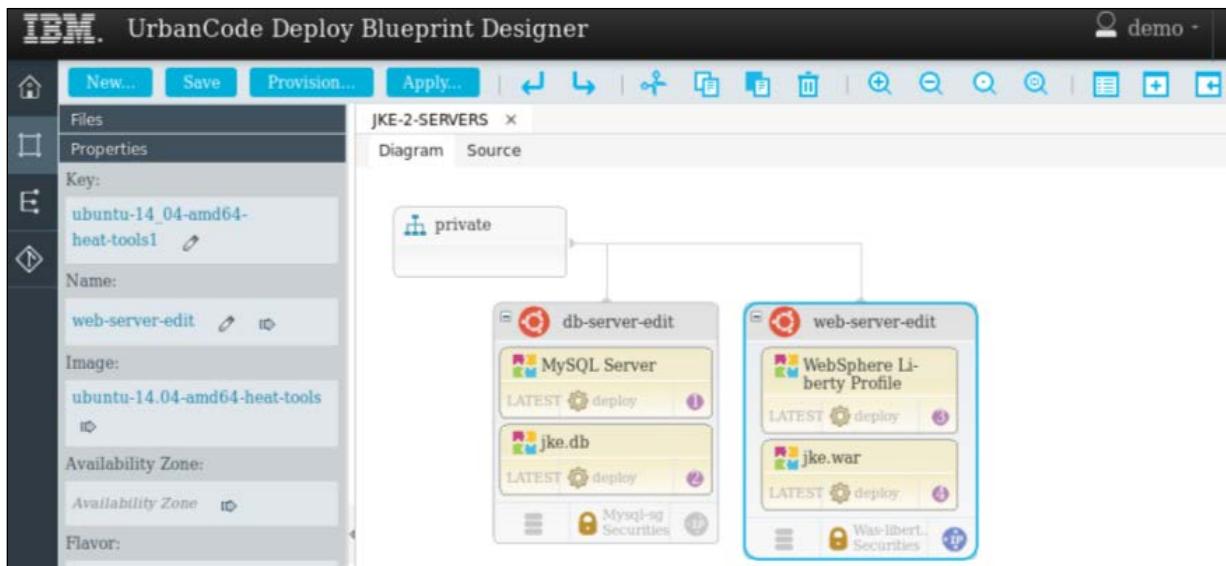
Exercise 6:

Tasks and results

Task 1. Change the blueprint.

In this exercise, you make a simple change by editing the names of the servers in the blueprint.

1. From the browser, switch to the **Blueprint Designer** tab.
2. Open the **JKE-2-SERVERS** blueprint.
3. In the Properties panel for the images, change the names to `db-server-edit` and `web-server-edit`.

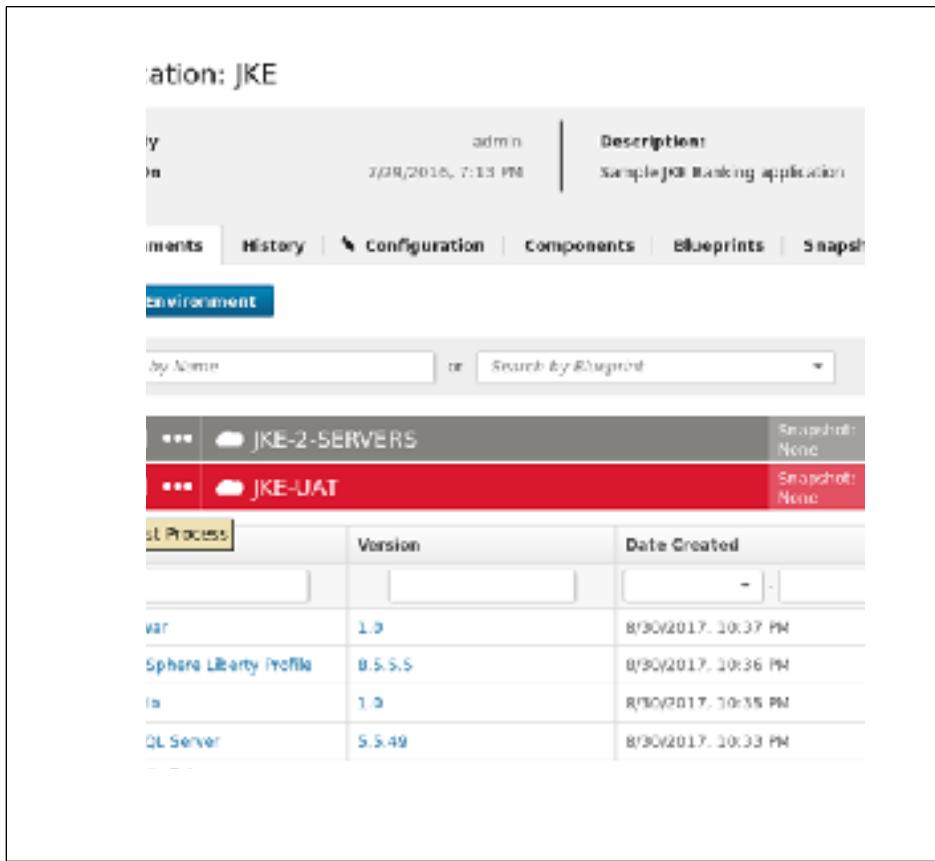


4. Save the blueprint.

Task 2. Run the modified blueprint.

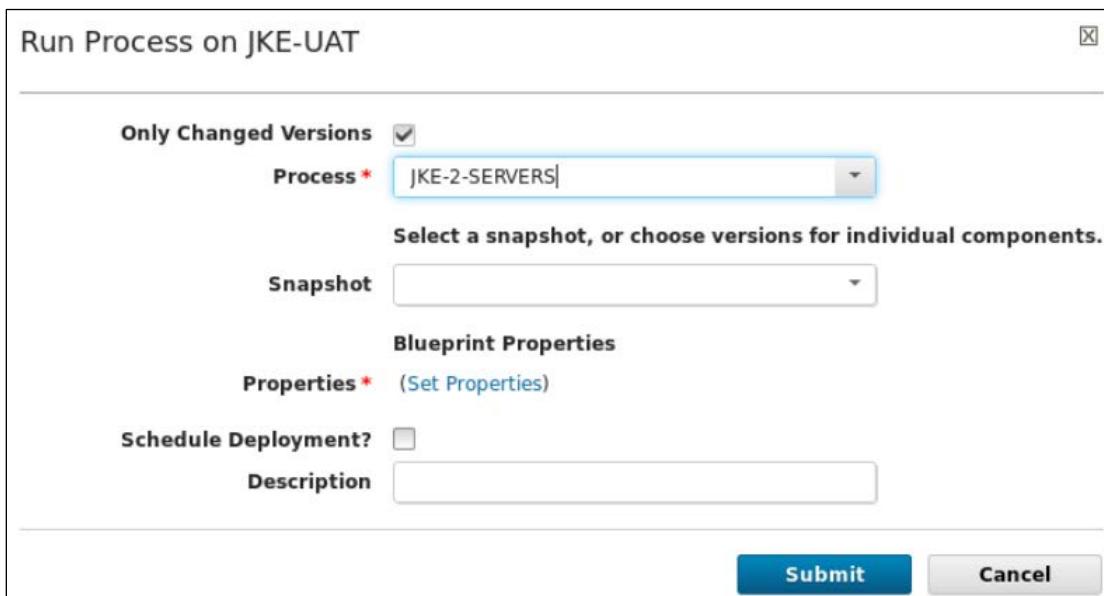
1. From the web browser, switch to the **UrbanCode Deploy** tab.
2. Click **Applications**, and click **JKE**.
3. From the **Environments** tab, expand the **JKE-UAT** environment.

4. To run the same process request for the JKE-UAT environment, click the **Request Process** icon .



Component	Version	Date Created
Tomcat	1.0	8/30/2017, 10:37 PM
Sphere Liberty Profile	8.5.5.5	8/30/2017, 10:36 PM
Java	1.8	8/30/2017, 10:35 PM
MySQL Server	5.5.49	8/30/2017, 10:33 PM

5. In the Run Process window, do the following steps:
- In the **Process** field, select **JKE-2-SERVERS**.
 - Next to **Properties**, click **Set Properties**.



Run Process on JKE-UAT

Only Changed Versions

Process*

Select a snapshot, or choose versions for individual components.

Snapshot

Blueprint Properties

Properties* [\(Set Properties\)](#)

Schedule Deployment?

Description

Submit **Cancel**

6. In the Blueprint Properties window, do the following steps:

- In the **Blueprint Version** field, select **1.0**.
- In the **Configuration** field, select **JKE-OPENSTACK-CONFIG**.
- In the **Configuration Version** field, select **1.0**.

Blueprint Properties

Cloud Provisioning

Blueprint	JKE-2-SERVERS
Cloud Project *	demo@OpenStack
Region *	RegionOne
Blueprint Version *	1.0
Configuration	JKE-OPENSTACK-CONFIG
Configuration Version *	1.0

Set property values for nodes to be created for this environment

availability_zone	nova
flavor	m1.medium
key_name	demo_key
network-id_for_private	private
public_network_id	public
subnet_id_ccde0dff-635f-49c5-b26b-4bd4bef89dc1	private >> private-subnet
ucd_relay_url	None

OK

7. Click **OK**, and click **Submit**.

IBM UrbanCode Deploy runs the application process, which applies the blueprint to the environment. The process shows you both, everything that occurs at the cloud level and everything that occurs at the application level.

Step	Progress	Start Time	Duration	Status
1. Provisioning Blueprint		4:30:56 PM	0:00:13	Success
2. Execute Stack Resources for JKE-2-SERVERS	15 / 17	4:31:10 PM	0:24:22	Running
Configuration	2 / 2	4:12:38 PM	0:00:07	Success
resource_tree (IBM:UrbanCode:ResourceTree)		4:12:38 PM	0:00:04	Success
ucd_agent_install_linux (OS:Heat:SoftwareConfig)		4:12:43 PM	0:00:02	Success
Servers	12 / 13	4:12:37 PM	0:06:06	Running
ubuntu-14_04-amd64-heat-tools1	6 / 6	4:12:37 PM	0:06:06	Success
ubuntu-14_04-amd64-heat-tools1_to_private_port (OS:Neutron:Port)		4:12:37 PM	0:00:06	Success
WebSphere_Liberty_Profile_sw_config (IBM:UrbanCode:SoftwareConfig:UCD)		4:12:43 PM	0:00:00	Success
ubuntu-14_04-amd64-heat-tools1_to_private_port_floating_ip (OS:Neutron:FloatingIP)		4:12:43 PM	0:00:02	Success
ubuntu-14_04-amd64-heat-tools1_mime (OS:Heat:MultipartMime)		4:12:46 PM	0:00:02	Success
ubuntu-14_04-amd64-heat-tools1 (OS:Nova:Server)		4:12:48 PM	0:00:15	Success
jke_war_sw_config (IBM:UrbanCode:SoftwareConfig:UCD)		4:13:08 PM	0:00:00	Success
WebSphere_Liberty_Profile (IBM:UrbanCode:SoftwareDeploy:UCD)		4:16:24 PM	0:01:05	Success
jke_war (IBM:UrbanCode:SoftwareDeploy:UCD)		4:17:29 PM	0:01:14	Success

You can see the changes to the server names in OpenStack.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
db-server-edit	ubuntu-14.04-amd64-heat-tools	10.0.0.16	m1.medium	demo_key	Active	nova	None	Running	19 minutes	<button>Create Snapshot</button>
web-server-edit	ubuntu-14.04-amd64-heat-tools	10.0.0.12 Floating IP: 172.24.4.11	m1.medium	demo_key	Active	nova	None	Running	19 minutes	<button>Create Snapshot</button>

Although changing the name of a server is a minor request, the ability to update a running environment can have more complex changes. For example, you can change the component versions that are deployed to the environment and then run the application process to deploy the new versions. You can also add virtual images to the environment.

In this unit, you created a blueprint that contains two servers, a database server and a web server, which represented separate functional areas of an application. Then, you assigned the appropriate security controls to each. You also created a configuration file that contains the initial values for the blueprint properties. You provisioned the new blueprint from IBM UrbanCode Deploy with the configuration file and observed the progress of provisioning. Finally, you made a change to the blueprint and ran the process again in IBM UrbanCode Deploy to update the running environment.

Unit 6 Repositories

IBM Training

Repositories

IBM UrbanCode Deploy Blueprint
Designer - v6.2.1.1

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Review the three types of Git repositories
- Copy a project to your local workspace
- Push and fetch changes with local and remote repositories

Repositories

© Copyright IBM Corporation 2017

Unit objectives

When you work with blueprints, you can organize and share them by using the Git distributed version control system.

The diagram illustrates the storage of files in repositories. It shows three types of repositories: Default, Team, and External. The Default repository is a single computer. The Team repository is a network of computers. The External repository is a computer connected to a cloud icon.

Repositories

© Copyright IBM Corporation 2017

Storage of files in repositories

Git is a distributed version control system. Blueprints and configuration files are stored in Git repositories, where they can be organized and shared with other team members.

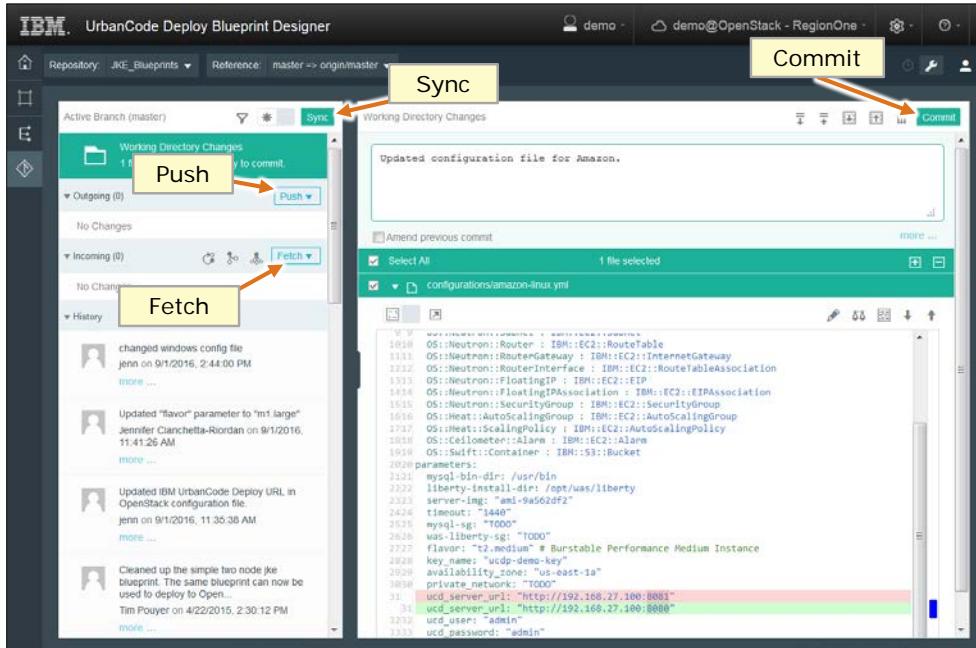
Default repository: By default, blueprint and configuration files are stored in a repository that is named **default**, and only the user that creates the files can access them. When you open a blueprint or configuration file in the Blueprint Designer from this repository, you always see the most recent version of that file. Saving a blueprint or configuration file updates the version in the Git repository automatically. No steps are necessary to store changes in the repository or retrieve updates from the repository.

Team repository: The blueprint design server creates a repository on the local Git server for each team, where all members of the team can access those files. For example, throughout the lab exercises in this course, you saved blueprints and configuration files to the demo project and any members of the demo team have access to the files. To share blueprints and configuration files with other team members, you associate the files with the repository for that team when you create them. Then you push completed changes from your individual clone to the shared repository. You must manipulate the files by using Git functions, such as commit, fetch, and push, to share your changes and view the changes others make.

External repository: You can also create other repositories, so you can treat blueprints with the same level of control as with application source code.

You can clone your repository and access it from within the Blueprint Designer. You can work with Git repositories that a system administrator sets up or with repositories on the web, such as IBM Bluemix DevOps Services.

Git functions on the Repositories page



Repositories

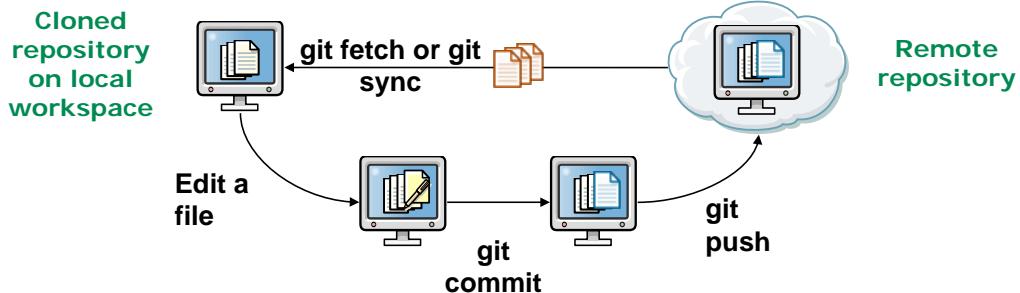
© Copyright IBM Corporation 2017

Git functions on the Repositories page

All Git functions that are available in the blueprint design server are displayed on the Repositories page. For information about Git commands, see the documentation at <https://git-scm.com/doc>.

Minimize conflicts

- Minimize conflicts by adopting a process for sharing repository files



Repositories

© Copyright IBM Corporation 2017

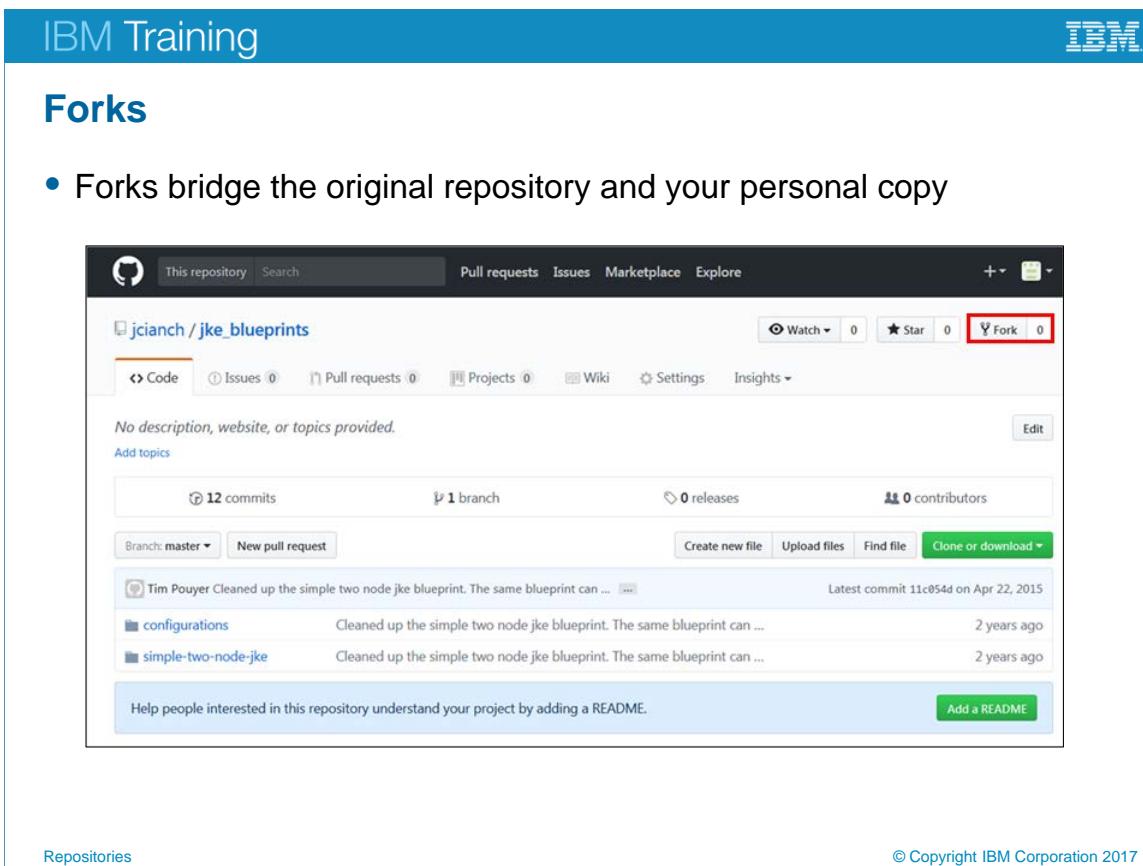
Minimize conflicts

Minimize conflicts between the commits that you make and the commits that you download from the repository by syncing your workspace with the repository before you change files. When you share files with a team or from a remote repository, the following steps show a good process:

Synchronize workspace with the repository: The Sync command downloads commits by using the `git fetch` command, reorders the commits by using the `git rebase` command, and moves your commits to the repository by using the `git push` command.

Make changes to blueprints in the Blueprint Designer.

Push changes from working directory to the Git repository: When you click **Commit**, all files in the working directory are committed to the Git repository. The commit is shown in the Outgoing pane. Then you can push the files from the Outgoing pane to the remote repository.



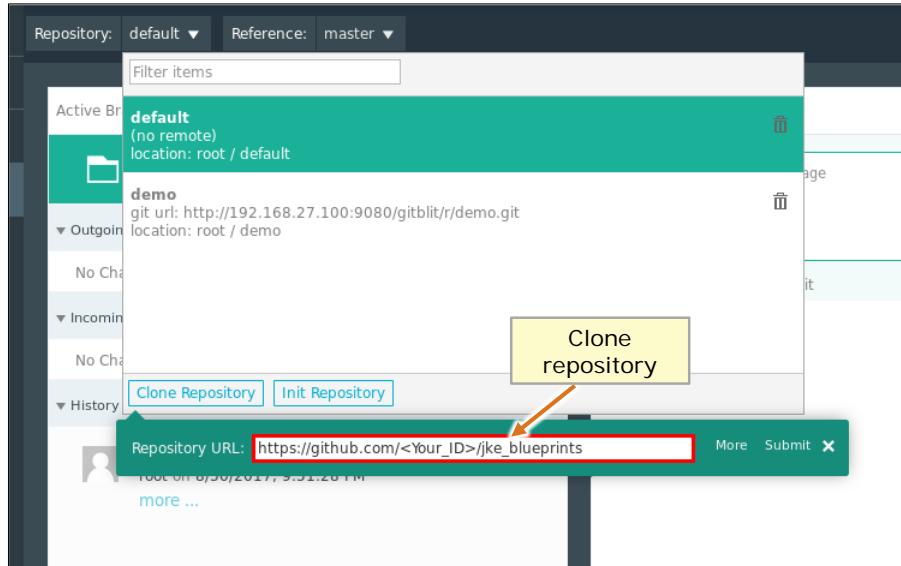
The screenshot shows a GitHub repository page for 'jcianch / jke_blueprints'. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation is a header with the repository name 'jcianch / jke_blueprints' and a 'Fork' button, which is highlighted with a red box. The main content area displays basic repository statistics: 12 commits, 1 branch, 0 releases, and 0 contributors. It also shows a list of recent commits by Tim Poyer. At the bottom, there's a call-to-action button 'Add a README'.

Forks

To work with a project, you first create a *fork*, which produces a personal copy of another project in your remote repository.

Cloned repository

- A cloned repository is a local copy on your computer



Repositories

© Copyright IBM Corporation 2017

Cloned repository

After you create the project, you can *clone* the repository and access it from within the Blueprint Designer, which clones projects by using the URL.

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, the 'File history' panel displays a timeline of changes. A yellow box labeled 'Change set' points to the commit history area. An orange bracket labeled 'Files in change set' points to the list of files in the current commit. The commit details on the right show a list of 9 files changed, including configuration files for autoscaling, configurations, and scaled app services.

File history

Repositories © Copyright IBM Corporation 2017

File history

IBM Training IBM

UrbanCode Deploy Blueprint Designer

Active Branch (master)

Working Directory Changes
Nothing to commit.

Outgoing (0) Push

Incoming (0) Fetch

History

Cleared up the simple two node jke blueprint. The same blueprint can now be used to deploy to Open...
Tim Poyer on 4/22/2015, 2:30:12 PM more ...

Added example two-node JKE blueprint that will deploy to Win2K8R2 images on AWS.
Tim Poyer on 4/22/2015, 12:29:54 AM more ...

Added example autoscaling two node JKE blueprint and additional configuration parameters to the open...
toucuer on 3/24/2015, 11:20:20 AM more ...

Commit (11c054)

9 files changed

- autoscale-two-node-jke/autoscale-two-node-jke.yml
- configurations/amazon-linux.yml
- configurations/amazon-windows.yml
- configurations/amazon.yml
- configurations/openstack-linux.yml
- configurations/openstack.yml
- scaled-app-srv/scaling-app-srv.yml
- simple-two-node-jke/simple-two-node-jke-windows-aws.yml
- simple-two-node-jke/simple-two-node-jke.yml

IBM Training

Transferring changes from local repository to remote repository

```

# Configuration file template
parameters:
  mysql-bin-dir: /usr/bin
  liberty-install-dir: /opt/was/liberty
  server-ing: "T000"
  timeout: "1440"
  mysql-sg: "T000"
  was-liberty-sg: "T000"
  flavor: "m1.medium"
  key_name: "demo_key"
  availability_zone: "nova"

ucd_server_url: "http://192.168.27.100:8081"
ucd_password: "admin"
ucd_relay_url: "None"
public_network: "T000" # public

```

```

# Configuration file template
parameters:
  mysql-bin-dir: /usr/bin
  liberty-install-dir: /opt/was/liberty
  server-ing: "T000"
  timeout: "1440"
  mysql-sg: "T000"
  was-liberty-sg: "T000"
  flavor: "m1.medium"
  key_name: "demo_key"
  availability_zone: "nova"

ucd_server_url: "http://192.168.27.100:8080"
ucd_password: "admin"
ucd_relay_url: "None"
public_network: "T000" # public

```

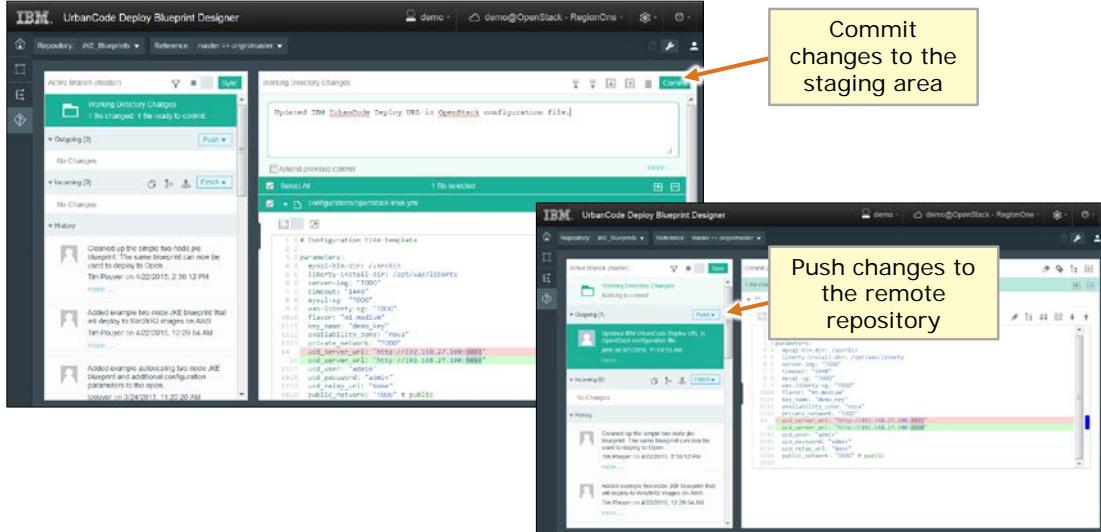
Repositories © Copyright IBM Corporation 2017

Transferring changes from local repository to remote repository

From the blueprint editor, you can see the directory, JKE_Blueprints, which contains all of the blueprints and configuration files from the cloned project. You can provision environments with the files, or you can make changes. In this example, the ucd_server_url parameter is updated. Now that you made changes to the working directory, you can *push* those changes back to your remote repository.

Committing changes

- Changes are stored as change sets in the working directory



Committing changes

Change sets are snapshots of your changes. When you are ready to update the remote repository with your changes, you first commit the change set to a staging area. After committing the change set, it is shown in the Outgoing section. These untracked changes are now tracked. You can then *push* the changes out to the remote repository. The git push command communicates with another repository, calculates what your local database has that the remote one does not, and then pushes the difference into the other repository.

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, there's a sidebar with 'Active Branch (master)' and a 'Working Directory Changes' section indicating 'Nothing to commit'. Below this are sections for 'Outgoing (0)', 'Incoming (0)', and 'History'. The 'History' section lists several commits by Tim Poyer, with one commit highlighted in green: 'Updated IBM UrbanCode Deploy URL in OpenStack configuration file' made on 9/1/2016 at 11:04:55 AM. On the right, the main pane is titled 'Commit (24efc1)' and shows a file named 'configurations/openstack-linux.yml'. The code editor displays the YAML configuration file, with the last few lines highlighted in red:

```

1: # Configuration file template
2:
3: parameters:
4:   mysql-bin-dir: /usr/bin
5:   liberty-install-dir: /opt/was/liberty
6:   server-ing: "T000"
7:   timeout: "1440"
8:   mysql-sg: "T000"
9:   was-liberty-sg: "T000"
10:  flavor: "m1.medium"
11:  key_name: "demo_key"
12:  availability_zone: "nova"
13:  private_network: "T000"
14:  ucd_server_url: "http://192.168.27.100:8081"
15:  ucd_server_url: "http://192.168.27.100:8089"
16:  ucd_user: "admin"
17:  ucd_password: "admin"
18:  ucd_relay_url: "None"
19:  public_network: "T000" # public
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
31:
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:
69:
70:
71:
72:
73:
74:
75:
76:
77:
78:
79:
80:
81:
82:
83:
84:
85:
86:
87:
88:
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:
100:
101:
102:
103:
104:
105:
106:
107:
108:
109:
110:
111:
112:
113:
114:
115:
116:
117:
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
779:
779:
780:
781:
782:
783:
784:
785:
786:
787:
788:
789:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
819:
820:
821:
822:
823:
824:
825:
826:
827:
828:
829:
829:
830:
831:
832:
833:
834:
835:
836:
837:
838:
839:
839:
840:
841:
842:
843:
844:
845:
846:
847:
848:
849:
849:
850:
851:
852:
853:
854:
855:
856:
857:
858:
859:
859:
860:
861:
862:
863:
864:
865:
866:
867:
868:
869:
869:
870:
871:
872:
873:
874:
875:
876:
877:
878:
879:
879:
880:
881:
882:
883:
884:
885:
886:
887:
888:
889:
889:
890:
891:
892:
893:
894:
895:
896:
897:
898:
898:
899:
900:
901:
902:
903:
904:
905:
906:
907:
908:
909:
909:
910:
911:
912:
913:
914:
915:
916:
917:
918:
919:
919:
920:
921:
922:
923:
924:
925:
926:
927:
928:
929:
929:
930:
931:
932:
933:
934:
935:
936:
937:
938:
939:
939:
940:
941:
942:
943:
944:
945:
946:
947:
948:
949:
949:
950:
951:
952:
953:
954:
955:
956:
957:
958:
959:
959:
960:
961:
962:
963:
964:
965:
966:
967:
968:
969:
969:
970:
971:
972:
973:
974:
975:
976:
977:
978:
979:
979:
980:
981:
982:
983:
984:
985:
986:
987:
988:
989:
989:
990:
991:
992:
993:
994:
995:
996:
997:
998:
999:
999:
1000:
1000:
1001:
1002:
1003:
1004:
1005:
1006:
1007:
1008:
1009:
1009:
1010:
1011:
1012:
1013:
1014:
1015:
1016:
1017:
1018:
1019:
1019:
1020:
1021:
1022:
1023:
1024:
1025:
1026:
1027:
1028:
1029:
1029:
1030:
1031:
1032:
1033:
1034:
1035:
1036:
1037:
1038:
1039:
1039:
1040:
1041:
1042:
1043:
1044:
1045:
1046:
1047:
1048:
1049:
1049:
1050:
1051:
1052:
1053:
1054:
1055:
1056:
1057:
1058:
1059:
1059:
1060:
1061:
1062:
1063:
1064:
1065:
1066:
1067:
1068:
1069:
1069:
1070:
1071:
1072:
1073:
1074:
1075:
1076:
1077:
1078:
1078:
1079:
1080:
1081:
1082:
1083:
1084:
1085:
1086:
1087:
1087:
1088:
1089:
1089:
1090:
1091:
1092:
1093:
1094:
1095:
1095:
1096:
1097:
1098:
1099:
1099:
1100:
1101:
1102:
1103:
1104:
1105:
1106:
1107:
1108:
1109:
1109:
1110:
1111:
1112:
1113:
1114:
1115:
1116:
1117:
1118:
1119:
1119:
1120:
1121:
1122:
1123:
1124:
1125:
1126:
1127:
1128:
1129:
1129:
1130:
1131:
1132:
1133:
1134:
1135:
1136:
1137:
1138:
1139:
1139:
1140:
1141:
1142:
1143:
1144:
1145:
1146:
1147:
1148:
1148:
1149:
1150:
1151:
1152:
1153:
1154:
1155:
1156:
1157:
1158:
1159:
1159:
1160:
1161:
1162:
1163:
1164:
1165:
1166:
1167:
1168:
1169:
1169:
1170:
1171:
1172:
1173:
1174:
1175:
1176:
1177:
1178:
1178:
1179:
1180:
1181:
1182:
1183:
1184:
1185:
1186:
1187:
1187:
1188:
1189:
1190:
1191:
1192:
1193:
1194:
1195:
1195:
1196:
1197:
1198:
1199:
1199:
1200:
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:
1209:
1209:
1210:
1211:
1212:
1213:
1214:
1215:
1216:
1217:
1218:
1219:
1219:
1220:
1221:
1222:
1223:
1224:
1225:
1226:
1227:
1228:
1229:
1229:
1230:
1231:
1232:
1233:
1234:
1235:
1236:
1237:
1238:
1239:
1239:
1240:
1241:
1242:
1243:
1244:
1245:
1246:
1247:
1248:
1248:
1249:
1250:
1251:
1252:
1253:
1254:
1255:
1256:
1257:
1258:
1259:
1259:
1260:
1261:
1262:
1263:
1264:
1265:
1266:
1267:
1268:
1269:
1269:
1270:
1271:
1272:
1273:
1274:
1275:
1276:
1277:
1278:
1278:
1279:
1280:
1281:
1282:
1283:
1284:
1285:
1286:
1287:
1287:
1288:
1289:
1290:
1291:
1292:
1293:
1294:
1295:
1295:
1296:
1297:
1298:
1299:
1299:
1300:
1301:
1302:
1303:
1304:
1305:
1306:
1307:
1308:
1309:
1309:
1310:
1311:
1312:
1313:
1314:
1315:
1316:
1317:
1318:
1319:
1319:
1320:
1321:
1322:
1323:
1324:
1325:
1326:
1327:
1328:
1329:
1329:
1330:
1331:
1332:
1333:
1334:
1335:
1336:
1337:
1338:
1339:
1339:
1340:
1341:
1342:
1343:
1344:
1345:
1346:
1347:
1348:
1349:
1349:
1350:
1351:
1352:
1353:
1354:
1355:
1356:
1357:
1358:
1359:
1359:
1360:
1361:
1362:
1363:
1364:
1365:
1366:
1367:
1368:
1369:
1369:
1370:
1371:
1372:
1373:
1374:
1375:
1376:
1377:
1378:
1378:
1379:
1380:
1381:
1382:
1383:
1384:
1385:
1386:
1387:
1387:
1388:
1389:
1390:
1391:
1392:
1393:
1394:
1395:
1395:
1396:
1397:
1398:
1399:
1399:
1400:
1401:
1402:
1403:
1404:
1405:
1406:
1407:
1408:
1408:
1409:
1410:
1411:
1412:
1413:
1414:
1415:
1416:
1417:
1418:
1419:
1419:
1420:
1421:
1422:
1423:
1424:
1425:
1426:
1427:
1428:
1429:
1429:
1430:
1431:
1432:
1433:
1434:
1435:
1436:
1437:
1438:
1439:
1439:
1440:
1441:
1442:
1443:
1444:
1445:
1446:
1447:
1448:
1448:
1449:
1450:
1451:
1452:
1453:
1454:
1455:
1456:
1457:
1458:
1458:
1459:
1460:
1461:
1462:
1463:
1464:
1465:
1466:
1467:
1468:
1468:
1469:
1470:
1471:
1472:
1473:
1474:
1475:
1476:
1477:
1478:
1478:
1479:
1480:
1481:
1482:
1483:
1484:
1485:
1486:
1487:
1487:
1488:
1489:
1490:
1491:
1492:
1493:
1494:
1495:
1495:
1496:
1497:
1498:
1499:
1499:
1500:
1501:
1502:
1503:
1504:
1505:
1506:
1507:
1508:
1508:
1509:
1510:
1511:
1512:
1513:
1514:
1515:
1516:
1517:
1518:
1519:
1519:
1520:
1521:
1522:
1523:
1524:
1525:
1526:
1527:
1528:
1529:
1529:
1530:
1531:
1532:
1533:
1534:
1535:
1536:
1537:
1538:
1538:
1539:
1540:
1541:
1542:
1543:
1544:
1545:
1546:
1547:
1548:
1548:
1549:
1550:
1551:
1552:
1553:
1554:
1555:
1556:
1557:
1558:
1558:
1559:
1560:
1561:
1562:
1563:
1564:
1565:
1566:
1567:
1568:
1568:
1569:
1570:
1571:
1572:
1573:
1574:
1575:
1575:
1576:
1577:
1578:
1579:
1580:
1581:
1582:
1583:
1584:
1585:
1585:
1586:
1587:
1588:
1589:
1589:
1590:
1591:
1592:
1593:
1594:
1595:
1595:
1596:
1597:
1598:
1599:
1599:
1600:
1601:
1602:
1603:
1604:
1605:
1606:
1607:
1608:
1608:
1609:
1610:
1611:
1612:
1613:
1614:
1615:
1616:
1617:
1618:
1619:
1619:
1620:
1621:
1622:
1623:
1624:
1625:
1626:
1627:
1628:
1629:
1629:
1630:
1631:
1632:
1633:
1634:
1635:
1636:
1637:
1638:
1638:
1639:
1640:
1641:
1642:
1643:
1644:
1645:
1646:
1647:
1648:
1648:
1649:
1650:
1651:
1652:
1653:
1654:
1655:
1656:
1657:
1658:
1658:
1659:
1660:
1661:
1662:
1663:
1664:
1665:
1666:
1667:
1668:
1668:
1669:
1670:
1671:
1672:
1673:
1674:
1675:
1675:
1676:
1677:
1678:
1679:
1680:
1681:
1682:
1683:
1684:
1685:
1685:
1686:
1687:
1688:
1689:
1689:
1690:
1691:
1692:
1693:
1694:
1695:
1695:
1696:
1697:
1698:
1699:
1699:
1700:
1701:
1702:
1703:
1704:
1705:
1706:
1707:
1708:
1708:
1709:
1710:
1711:
1712:
1713:
1714:
1715:
1716:
1717:
1718:
1719:
1719:
1720:
1721:
1722:
1723:
1724:
1725:
1726:
1727:
1728:
1729:
1729:
1730:
1731:
1732:
1733:
1734:
1735:
1736:
1737:
1738:
1738:
1739:
1740:
1741:
1742:
1743:
1744:
1745:
1746:
1747:
1748:
1748:
1749:
1750:
1751:
1752:
1753:
1754:
1755:
1756:
1757:
1758:
1758:
1759:
1760:
1761:
1762:
1763:
1764:
1765:
1766:
1767:
1768:
1768:
1769:
1770:
1771:
1772:
1773:
1774:
1775:
1776:
1777:
1778:
1778:
1779:
1780:
1781:
1782:
1783:
1784:
1785:
1785:
1786:
1787:
1788:
1789:
1789:
1790:
1791:
1792:
1793:
1794:
1795:
1795:
1796:
1797:
1798:
1799:
1799:
1800:
1801:
1802:
1803:
1804:
1805:
1806:
1807:
1808:
1808:
1809:
1810:
1811:
1812:
1813:
1814:
1815:
1816:
1817:
1818:
1819:
1819:
1820:
1821:
1822:
1823:
1824:
1825:
1826:
1827:
1828:
1829:
1829:
1830:
1831:
1832:
1833:
1834:
1835:
1836:
1837:
1838:
1838:
1839:
1840:
1841:
1842:
1843:
1844:
1845:
1846:
1847:
1848:
1848:
1849:
1850:
1851:
1852:
1853:
1854:
1855:
1856:
1857:
1858:
1858:
1859:
1860:
1861:
1862:
1863:
1864:
1865:
1866:
1867:
1868:
1868:
1869:
1870:
1871:
1872:
1873:
1874:
1875:
1876:
1877:
1878:
1878:
1879:
1880:
1881:
1882:
1883:
1884:
1885:
1886:
1887:
1888:
1889:
1889:
1890:
1891:
1892:
1893:
1894:
1895:
1896:
1897:
1898:
1898:
1899:
1900:
1901:
1902:
1903:
1904:
1905:
1906:
1907:
1908:
1909:
1909:
1910:
1911:
1912:
1913:
1914:
1915:
1916:
1917:
1918:
1919:
1919:
1920:
1921:
1922:
1923:
1924:
1925:
1926:
1927:
1928:
1929:
1929:
1930:
1931:
1932:
1933:
1934:
1935:
1936:
1937:
1938:
1939:
1939:
1940:
1941:
1942:
1943:
1944:
1945:
1946:
1947:
1948:
1949:
1949:
1950:
1951:
1952:
1953:
1954:
1955:
1956:
1957:
1958:
1959:
1959:
1960:
1961:
1962:
1963:
1964:
1965:
1966:
1967:
1968:
1969:
1969:
1970:
1971:
1972:
1973:
1974:
1975:
1976:
1977:
1978:
1978:
1979:
1980:
1981:
1982:
1983:
1984:
1985:
1986:
1987:
1988:
1989:
1989:
1990:
1991:
1992:
1993:
1994:
1995:
1996:
1997:
1998:
1999:
1999:
2000:
2001:
2002:
2003:
2004:
2005:
2006:
2007:
2008:
2009:
2009:
2010:
2011:
2012:
2013:
2014:
2015:
2016:
2017:
2017:
2018:
2019:
2019:
2020:
2021:
2022:
2023:
2024:
2025:
2026:
2027:
2028:
2029:
2029:
2030:
2031:
2032:
2033:
2034:
2035:
2036:
2037:
2038:
2039:
2039:
2040:
2041:
2042:
2043:
2044:
2045:
2046:
2047:
2048:
2049:
2049:
2050:
2051:
2052:
2053:
2054:
2055:
2056:
2057:
2058:
2059:
2059:
2060:
2061:
2062:
2063:
2064:
2065:
2066:
2067:
2068:
2069:
2069:
2070:
2071:
2072:
2073:
2074:
2075:
2076:
2077:
2078:
2078:
2079:
2080:
2081:
2082:
2083:
2084:
2085:
2086:
2087:
2088:
2089:
2089:
2090:
2091:
2092:
2093:
2094:
2095:
2096:
2097:
2098:
2099:
2099:
2100:
2101:
2102:
2103:
2104:
2105:
2106:
2107:
2108:
2109:
2109:
2110:
2111:
2112:
2113:
2114:
2115:
2116:
2117:
2118:
2119:
2119:
2120:
2121:
2122:
2123:
2124:
2125:
2126:
2127:
2128:
2129:
2129:
2130:
2131:
2132:
2133:
2134:
2135:
2136:
2137:
2138:
2139:
2139:
2140:
2141:
2142:
2143:
2144:
2145:
2146:
2147:
2148:
2149:
2149:
2150:
2151:
2152:
2153:
2154:
2155:
2156:
2157:
2158:
2159:
2159:
2160:
2161:
2162:
2163:
2164:
2165:
2166:
2167:
2168:
2169:
2169:
2170:
2171:
2172:
2173:
2174:
2175:
2176:
2177:
2178:
2178:
2179:
2180:
2181:
2182:
2183:
2184:
2185:
2186:
2187:
2188:
2188:
2189:
2190:
2191:
2192:
2193:
2194:
2195:
2196:
2197:
2198:
2199:
2199:
2200:
2201:
2202:
2203:
2204:
2205:
2206:
2207:
2208:
2209:
2209:
2210:
2211:
2212:
2213:
2214:
2215:
2216:
2217:
2218:
2219:
2219:
2220:
2221:
2222:
2223:
2224:
222
```

The screenshot shows a GitHub repository page for `jke_blueprints`. At the top, there are buttons for Watch, Star, and Fork. Below the header, there are tabs for Code, Issues, Pull requests, Projects, Wiki, Settings, and Insights. A message says "No description, website, or topics provided." There are 13 commits, 1 branch, 0 releases, and 0 contributors. A green button at the top right says "Clone or download". Below the commit list, there are two commits highlighted with orange boxes:

- jclanch Updated OpenStack configuration file** - Latest commit 54e18fe 3 minutes ago
- configurations Updated OpenStack configuration file** - 3 minutes ago

Below the commits, a code editor shows a configuration file named `simple-two-node-jke` with the following content:

```

Configuration
1 # Configuration file template
2
3# parameters:
4 mysql-bin-dir: /usr/bin
5 liberty-install-dir: /opt/was/liberty
6 server-img: "e4b0ead7-68ba-4c87-ba4f-f1649b1be11c" # ubuntu-14.04-amd64-heat-tools
7 timeout: "1440"
8 mysql-sg: "7b5bb8ce-22c2-4511-a9f2-6c77fb70904a02" # mysql-sg
9 was-liberty-sg: "2fc88ade-17b0-4602-acf5-c6e03903e342" # was-liberty-sg
10 flavor: "medium"
11 key_name: "medium"
12 availability: "key"
13 availability: "nova"
14 ucd_server_url: "http://192.168.27.100:8880" # private
15 ucd_password: "admin"
16 ucd_relay_url: "None"
17 public_network: "4a4ff7b4a-596b-4396-a7ad-edfcf26aa367" # public
18
19

```

At the bottom of the code editor, there is a copyright notice: "© Copyright IBM Corporation 2017".

Recent activity

In the remote project, you can see the recent project activity. Open the file to confirm the changes.

Editing the source code

- You can edit the code of blueprints and configuration files

```

1 # Configuration file template
2 # Updated configurations for OpenStack
3
4 parameters:
5   mysql-bin-dir: /usr/bin
6   liberty-install-dir: /opt/was/liberty
7   server-img: "e4b9ea67-688a-4c07-ba4f-f1649b1bel1c" # ubuntu-14.04
8   timeout: "1440"
9   mysql-sq: "7b5b88ce-22c2-4511-a9f2-6c7f87090402" # mysql-sq
10  was-liberty-sq: "2fc88ade-17b0-4602-acf5-c6e03903e342" # was-liberty-sq
11  flavor: "m1.medium" flavor: "m1.large"
12  key_name: "demo key"
13  availability_zone: "nova"
14  private_network: "dec76954-f367-4de8-bba7-5b8a7992612d" # private
15  ucd_server_url: "http://192.168.27.100:8080"
16  ucd_user: "admin"
17  ucd_password: "admin"
18  ucd_relay_url: "None"
19  public_network: "4a4f7b4a-596b-4396-a7ad-efdcf26aa367" # public

```

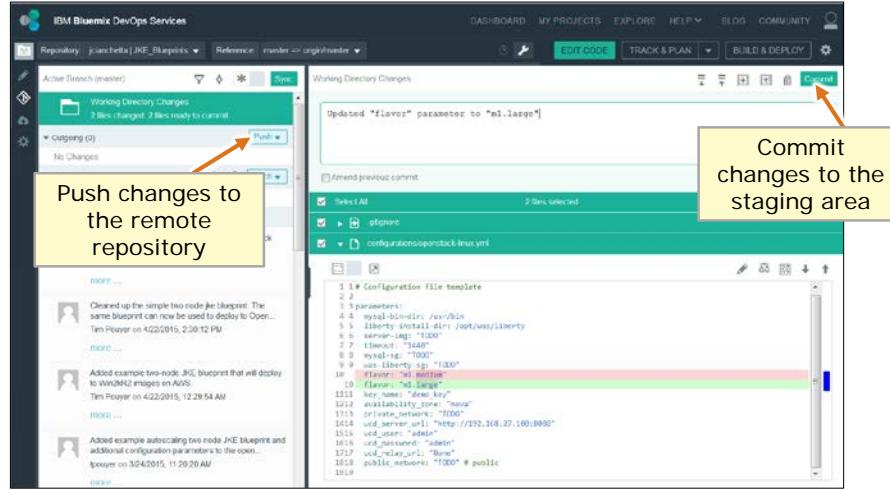
© Copyright IBM Corporation 2017

Editing the source code

You can edit the HOT source code of blueprints and configuration files directly.

Making changes available to local repositories

- Remote repositories make changes available to local repositories



Repositories

© Copyright IBM Corporation 2017

Making changes available to local repositories

When you want to share work with others from the remote repository, you can push changes to make them available to local repositories.

IBM Training

Fetching

- Fetching checks for changes on the remote repository

The screenshot shows the UrbanCode Deploy Blueprint Designer interface. On the left, there's a sidebar with 'Active Branch (master)', 'Working Directory Changes' (empty), 'Outgoing (0)' (empty), 'Incoming (1)' (with a message from Jennifer Chanchella-Houtan about updating a flavor parameter), and 'History' (with a message from Tim Poyer about cleaning up a blueprint). In the center, there's a 'Commit (5a663d)' panel showing a file named 'configurations/openstack/linux.yml'. The code editor displays configuration parameters like 'flavor: "m1.large"', 'key_name: "demo_key"', and 'ucd_server_url: "http://192.168.27.106:8080"'. The right side of the screen has the 'IBM' logo.

Repositories

© Copyright IBM Corporation 2017

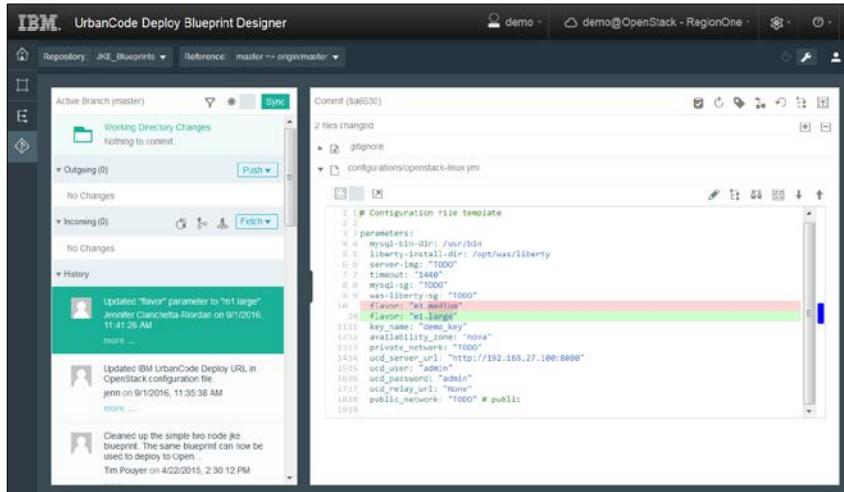
Fetching

From the repository tab of the Blueprint Designer, you can check for change by clicking **Fetch**. The `git fetch` command communicates with a remote repository and fetches all of the information that is in that repository that is not in your current one. Then it stores it in your local database.

IBM Training IBM

Syncing

- Syncing updates your local copy of the repository



The screenshot shows the UrbanCode Deploy Blueprint Designer application. On the left, there's a sidebar with 'Active Branch (master)', 'Working Directory Changes' (nothing to commit), 'Outgoing (0)' (no changes), 'Incoming (0)' (no changes), and a 'History' section. The 'History' section contains three entries:

- Updated "flavor" parameter to "m1.large" under Chancetta-Riduan on 9/1/2016, 11:41:26 AM
- Updated IBM UrbanCode Deploy URL in OpenStack configuration file. jenn on 9/1/2016, 11:35:38 AM
- Cleaned up the simple two node jke blueprint. The same blueprint can now be used to deploy to Open... Tim Poyer on 4/22/2015, 2:30:12 PM

On the right, the main pane shows a 'Commit (ba663c)' with '2 files changed'. It displays a configuration file with the following content:

```

1: # Configuration file template
2: 
3: parameters:
4:   mylibt-bin-dir: /usr/lib
5:   liberty-install-dir: /opt/was/liberty
6:   server-img: "7000"
7:   timeout: "10000"
8:   was-lgi: "7000"
9:   was-liberty-rq: "1000"
10:  flavor: "m1.medium"
11:  key_name: "openstack"
12:  availability_zone: "nova"
13:  private_network: "7000"
14:  ucd_ip: "http://192.168.37.100:8080"
15:  ucd_user: "admin"
16:  ucd_password: "admin"
17:  ucd_relay_url: "None"
18:  public_network: "7000" # public
19:

```

Repositories

© Copyright IBM Corporation 2017

Syncing

After you sync, the change is accepted and becomes part of the history.

Unit summary

- Review the three types of Git repositories
- Copy a project to your local workspace
- Push and fetch changes with local and remote repositories

Repositories

© Copyright IBM Corporation 2017

Unit summary

You can use three types of Git repositories: the default Git repository, a team-based repository on the local Git server, and an external Git repository. Saving a blueprint or configuration file updates the version in the default Git repository automatically.

If items are stored in a team-based or external repository, you must modify the items in an individual clone of that repository and push completed changes to the shared repository. You can minimize conflicts in the repository by synchronizing your workspace with the repository before you change files.

Exercises: Using repositories to store and manage blueprints

- Copy a remote repository to your local workspace
- Update a configuration file and provision a blueprint to the OpenStack cloud
- Push changes to a remote repository
- Fetch changes from a remote repository

Repositories

© Copyright IBM Corporation 2017

Exercises: Using repositories to store and manage blueprints

You store blueprints and configuration files that you create with the blueprint designer in Git repositories, so that you can version and back up the blueprints. By storing blueprints and configurations in the Git repository for a team, you can also share the files with members of that team.

In these exercises, you explore the integration of the blueprint design server with Git to make versions of blueprints.

Before beginning the following exercises, sign up for a Bluemix account. Go to <https://github.com/>, sign up for an account, and confirm it through your email.

Exercise 1

Copy a project to your local workspace

Repositories

© Copyright IBM Corporation 2017

Exercise 1: Copy a project to your local workspace

Exercise 1: Copy a project to your local workspace

In this exercise, you copy a project to your local workspace.

To do this, you will:

- Fork a repository.
- Clone a Git repository in the Blueprint Designer.

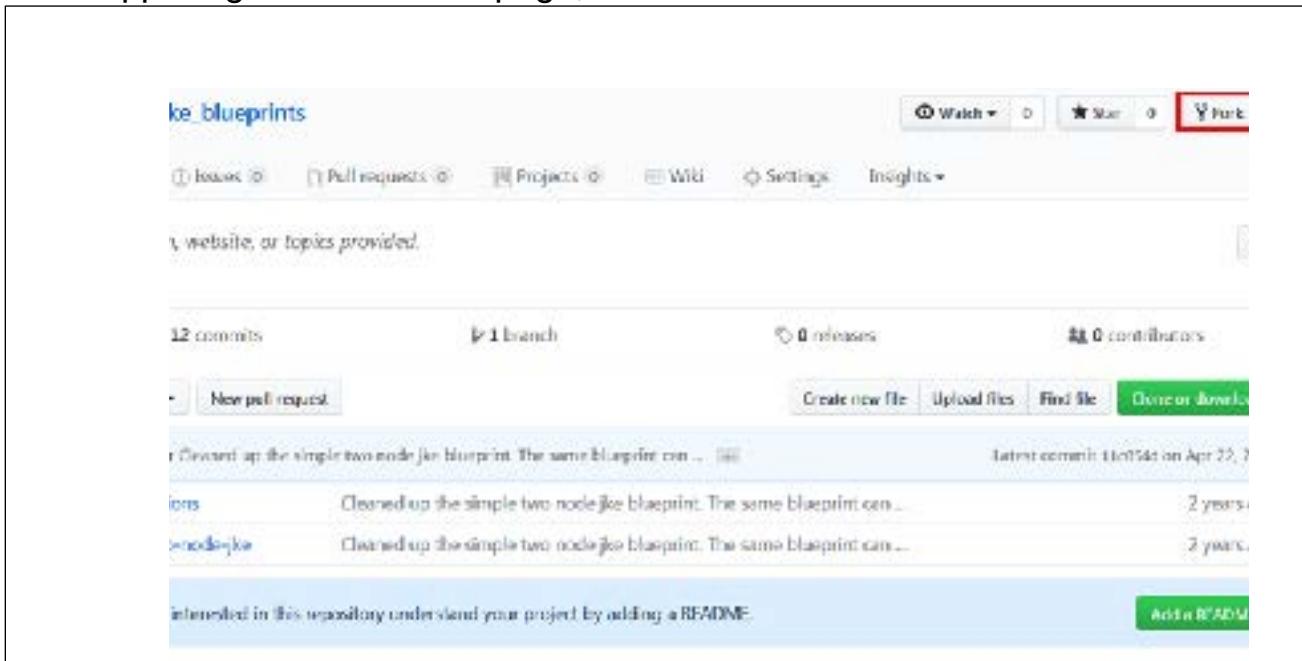
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 1: Tasks and results

Task 1. Fork a repository.

In this exercise, you create a personal copy of a remote repository through a Git process called *forking*.

1. Locate the *jke blueprints* repository by opening a new tab, and typing the GitHub URL at https://github.com/jcianch/jke_blueprints.
2. In the upper-right corner of the page, click **Fork**.



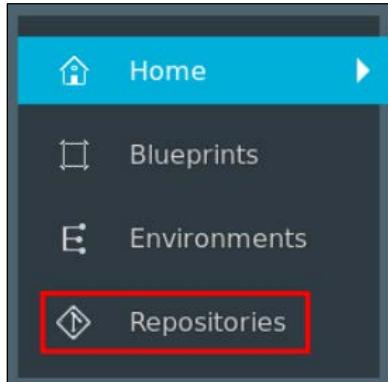
The screenshot shows the GitHub repository page for 'jke_blueprints'. At the top right, there is a red box highlighting the 'Fork' button. Below the header, there are tabs for Issues, Pull requests, Projects, Wiki, Settings, and Insights. The main area displays 12 commits, 1 branch, 0 releases, and 0 contributors. A 'New pull request' button is visible. The commit history shows two entries from 'jke' and 'jke_blueprints'. A note at the bottom encourages adding a README, with a 'Add a README' button.

3. You now have a personal copy of the *jke_blueprints* project.
4. Note the URL for your repository. It is similar to https://github.com/<Your_ID/jke_blueprints.

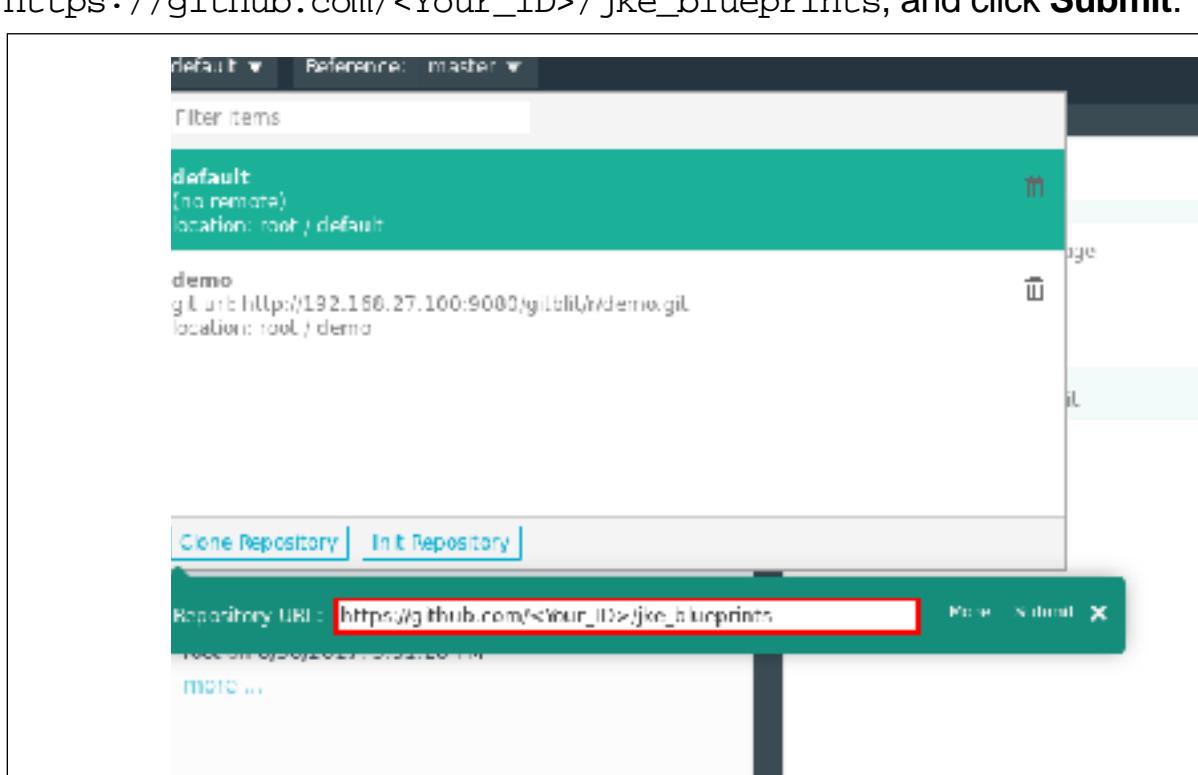
Task 2. Clone a Git repository in the Blueprint Designer.

To work with your newly created project, *jke_blueprints*, you can *clone* the repository and access it from within the Blueprint Designer. When you clone a repository, you create a local copy on your computer and sync between the two locations.

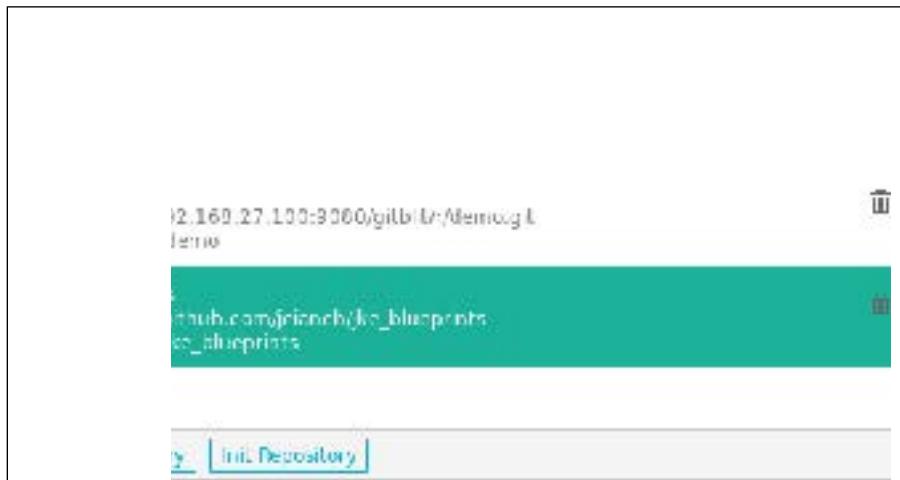
1. From the Blueprint Designer home page, click **Repositories**.



2. Next to the **Repository** field, click the drop-down arrow, and click **Clone Repository**. Currently, the default repository is open.
3. In the **Repository URL** field, type https://github.com/<Your_ID>/jke_blueprints, and click **Submit**.



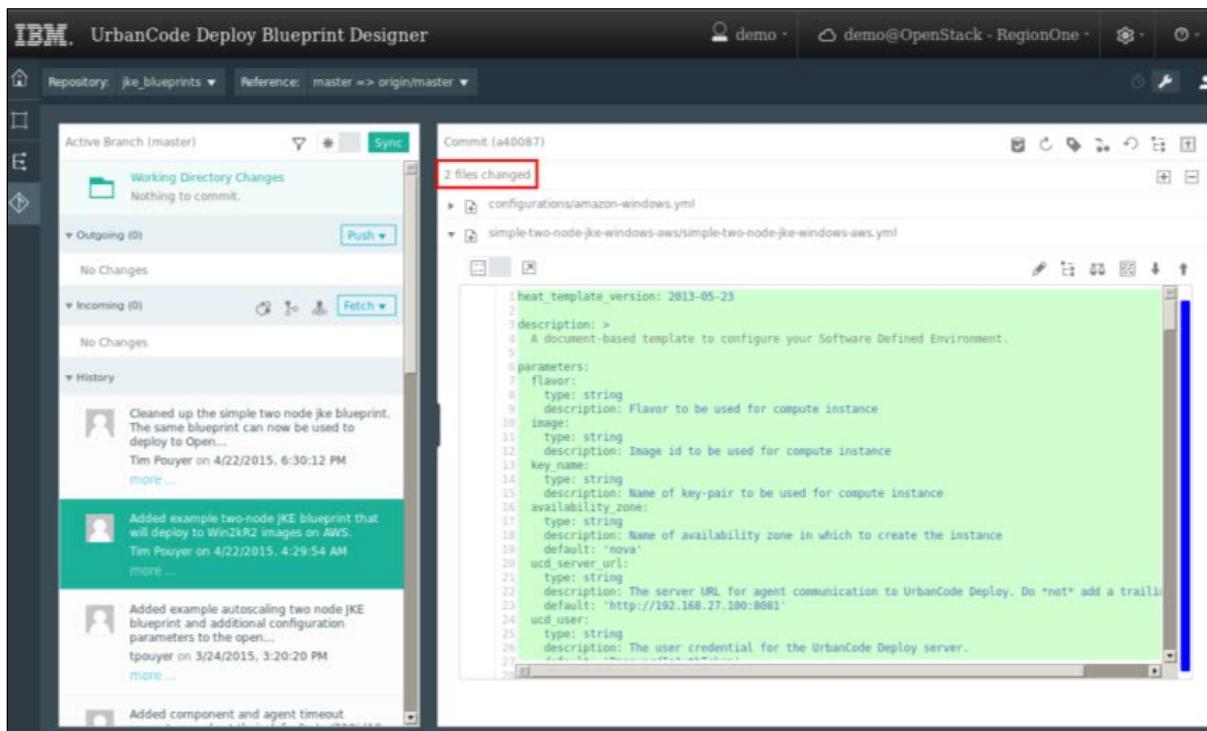
4. Click **jke_blueprints**, the cloned repository, to select and open it.



5. Observe the history that is associated with this project. The cloned repository contains the history of the original, or *upstream*, repository.

Commit Log	Date	Message
Cleaned up the simple two node jke blueprint. The same blueprint can now be used to deploy to Open...	Tim Pouyer on 4/22/2015, 6:30:12 PM	more ...
Added example two-node JKE blueprint that will deploy to Win2K2 images on AWS.	Tim Pouyer on 4/22/2015, 4:29:54 AM	more ...
Added example autoscaling two node JKE blueprint and additional configuration parameters to the open...	tpouyer on 3/24/2015, 3:20:20 PM	more ...
Added component and agent timeout		

6. Click the different versions in the History panel and note the attached file changes for each.



Exercise 2

Modify the configuration file and provisioning the blueprint

Exercise 2: Modify the configuration file and provisioning the blueprint

Exercise 2: Modify the configuration file and provisioning the blueprint

Working with your own copy of a remote project, you can modify the files to use in your cloud deployments

To do this, you will:

- Modify the configuration file.
- Provision the blueprint to the OpenStack cloud.

For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 2:

Tasks and results

Task 1. Modify the configuration file.

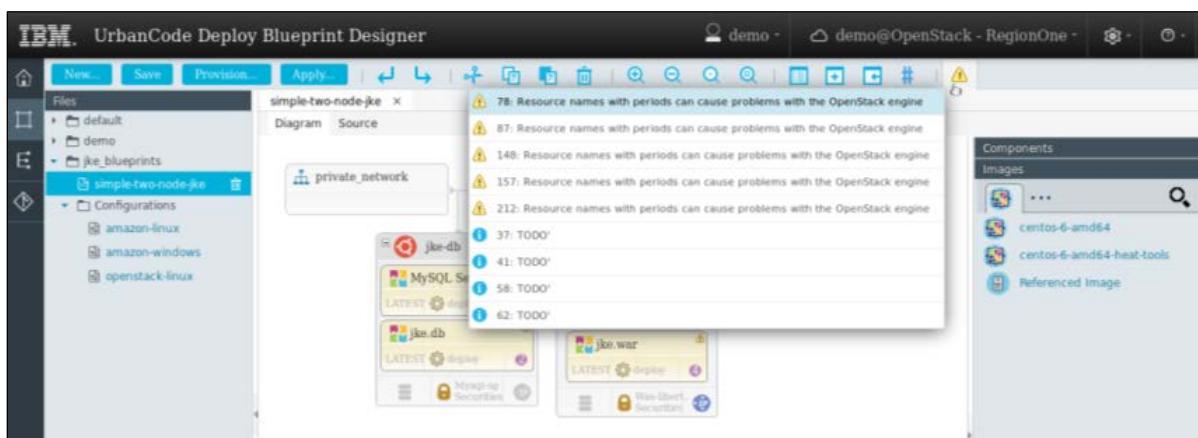
Working with your own copy of a remote project, you can modify the files to use in your cloud deployments.

1. From the left navigation, click the **Blueprints** tab.
2. Observe that you now have a new folder, called **jke_blueprints**, under the Files panel. It contains one blueprint, **simple-two-node-jke**, and three configuration files.



3. Click the **simple-two-node-jke** blueprint to open it.

Notice the status messages. The last four messages identify areas where parameters must be defined.



4. Click the **Source** tab to see the specific lines of code that need parameters.

```

simple-two-node-jke x
Diagram Source
33
34 private_network:
35   type: string
36   description: Private network
37   default: 'TODO'
38 public_network:
39   type: string
40   description: Generated for floating IP
41   default: 'TODO'
42 server-img:
43   type: string
44   description: ubuntu-trusty-cloudimg-amd64
45 mysql-sg:
46   type: string
47   description: The security group for MySQL (allows for communication over port 3306)
48 was-liberty-sg:
49   type: string
50   description: The security group for WAS Liberty (allows for communication over port 1440)
51 timeout:
52   type: string
53   description: Generated
54   default: '1440'
55 mysql-bin-dir:
56   type: string
57   description: Generated
58   default: 'TODO'
59 liberty-install-dir:
60   type: string
61   description: Generated
62   default: 'TODO'
63 resources:
64
65 jke-db:
66   type: OS::Nova::Server
67   properties:
68     user_data format: RAW

```

Task 2. Provision the blueprint to the OpenStack cloud.

Use the modified configuration file to provision the **simple-two-node-jke** blueprint to your OpenStack cloud.

1. In the **jke_blueprints** folder, expand the **Configurations** folder, and click the **openstack-linux** configuration file to open it.

```

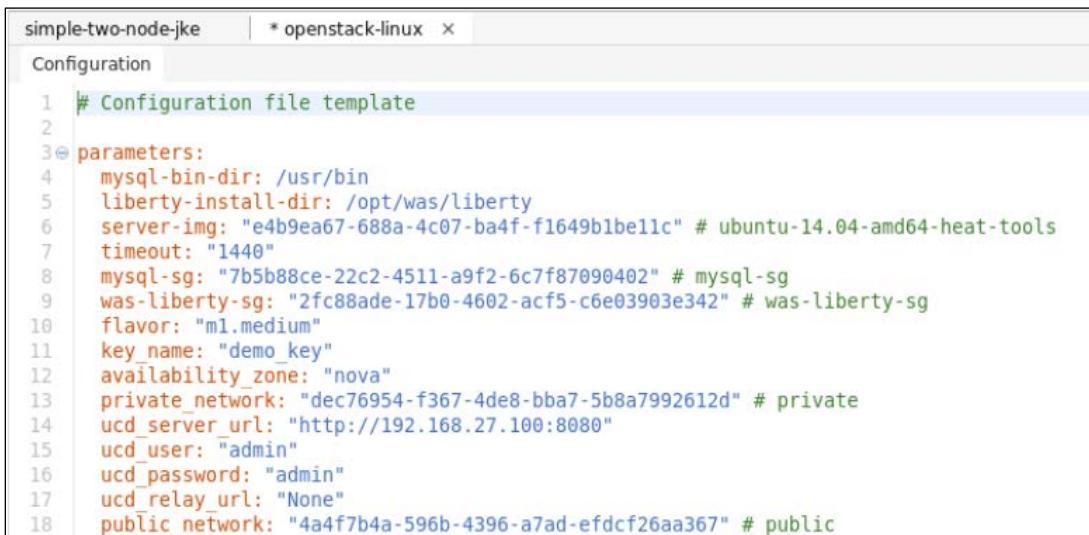
simple-two-node-jke | openstack-linux x
Configuration
1 # Configuration file template
2
3@ parameters:
4   mysql-bin-dir: /usr/bin
5   liberty-install-dir: /opt/was/liberty
6   server-img: "TODO"
7   timeout: "1440"
8   mysql-sg: "TODO"
9   was-liberty-sg: "TODO"
10  flavor: "m1.medium"
11  key_name: "demo_key"
12  availability_zone: "nova"
13  private_network: "TODO"
14  ucd_server_url: "http://192.168.27.100:8081"
15  ucd_user: "admin"
16  ucd_password: "admin"
17  ucd_relay_url: "None"
18  public_network: "TODO" # public

```

2. To import properties from the blueprint, click the **Add** icon , select the **simple-two-node-jke** blueprint, and click **OK**.
3. In the configuration file, specify the properties for the deployment to the OpenStack cloud. To assign a value to a missing parameter, click the word

TODO to the right of a parameter, press Ctrl+Space bar at the same time, and select the following options:

- server-img: Ubuntu-14.04-amd64-heat-tools
 - mysql-sg: mysql-sq
 - was-liberty-sg: was-liberty-sq
 - private_network: private
 - public_network: public
4. The IBM UrbanCode Server port number has changed since the creation of this file. In the **ucd_server_url** field, change the port number of the URL to 8080.



```

simple-two-node-jke | * openstack-linux ×
Configuration
1 # Configuration file template
2
3 @ parameters:
4   mysql-bin-dir: /usr/bin
5   liberty-install-dir: /opt/was/liberty
6   server-img: "e4b9ea67-688a-4c07-ba4f-f1649b1be11c" # ubuntu-14.04-amd64-heat-tools
7   timeout: "1440"
8   mysql-sg: "7b5b88ce-22c2-4511-a9f2-6c7f87090402" # mysql-sg
9   was-liberty-sg: "2fc88ade-17b0-4602-acf5-c6e03903e342" # was-liberty-sg
10  flavor: "m1.medium"
11  key_name: "demo_key"
12  availability_zone: "nova"
13  private_network: "dec76954-f367-4de8-bba7-5b8a7992612d" # private
14  ucd_server_url: "http://192.168.27.100:8080"
15  ucd_user: "admin"
16  ucd_password: "admin"
17  ucd_relay_url: "None"
18  public_network: "4a4f7b4a-596b-4396-a7ad-efdcf26aa367" # public

```

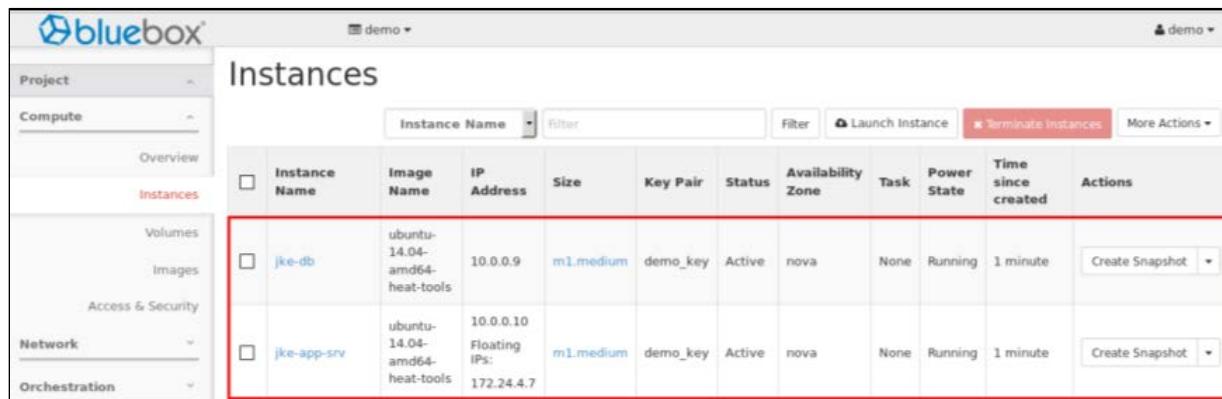
5. Save the configuration file.
6. From the **simple-two-node-jke** blueprint, click **Provision**.
7. In the “Provision Blueprint to new Environment” window, do the following steps:
 - In the **Environment Name** field, type **JKE-DEV2**.
 - In the **Configuration** field, select **openstack-linux**.

Provision Blueprint to new Environment

Environment Name *	JKE-DEV2
Cloud Project	demo@OpenStack
Configuration	openstack-linux
Additional Configuration	Select Configuration...
Set the parameter values for this environment	
Liberty Install Dir	/opt/was/liberty
Mysql Bin Dir	/usr/bin
Timeout	1440
UrbanCode Deploy Password	****
UCD Relay URL	None
UrbanCode Deploy Server URL	http://192.168.27.100:8080
UCD User	admin
Availability Zone	nova
Key Name	demo_key
Default Network	internal
Provision  Cancel	

8. Click Provision.

- Check the provisioning in OpenStack, the Blueprint Designer, and IBM UrbanCode Deploy in the way that you have verified changes throughout this lab. For example, here are the provisioned instances in OpenStack:



<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	jke-db	ubuntu-14.04-amd64-heat-tools	10.0.0.9	m1.medium	demo_key	Active	nova	None	Running	1 minute	<input type="button" value="Create Snapshot"/>
<input type="checkbox"/>	jke-app-srv	ubuntu-14.04-amd64-heat-tools	10.0.0.10 Floating IPs: 172.24.4.7	m1.medium	demo_key	Active	nova	None	Running	1 minute	<input type="button" value="Create Snapshot"/>

Exercise 3

Make file changes and updating the remote repository

Exercise 3: Make file changes and updating the remote repository

Exercise 3: Make file changes and updating the remote repository

Now that you made changes to the working directory, you can push those changes back to your remote repository.

To do this, you will:

- Push changes to the remote repository.
- Verify changes in the remote repository.
- Push changes to local repositories.
- Verify changes in the local repository.

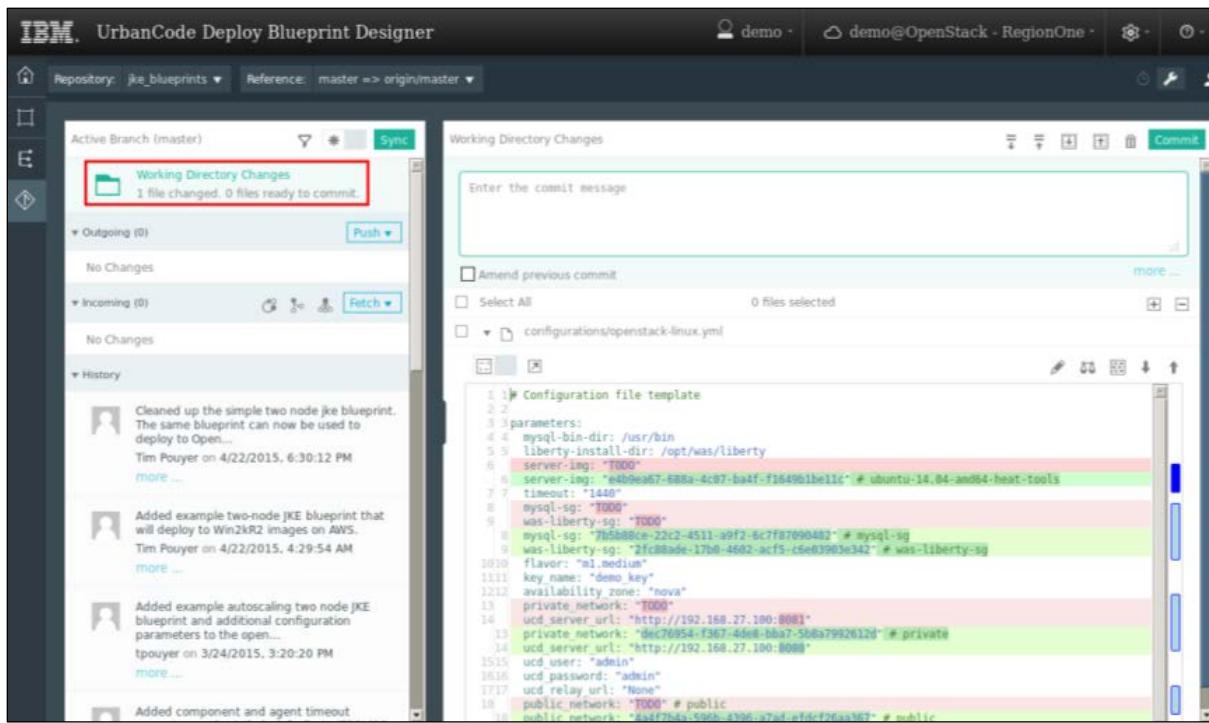
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 3:

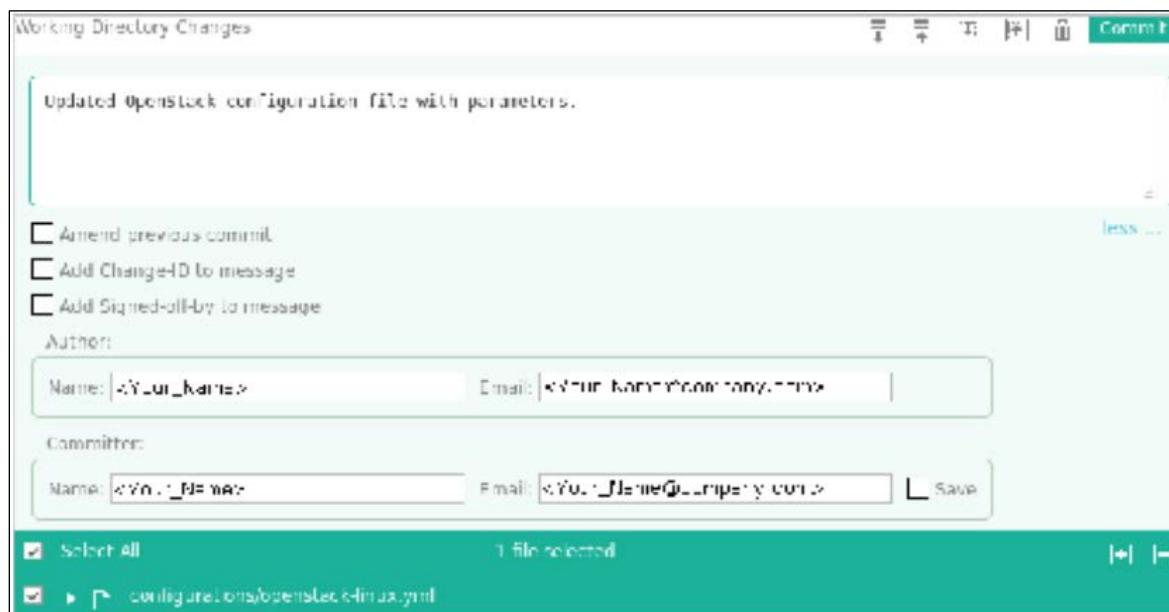
Tasks and results

Task 1. Push changes to the remote repository.

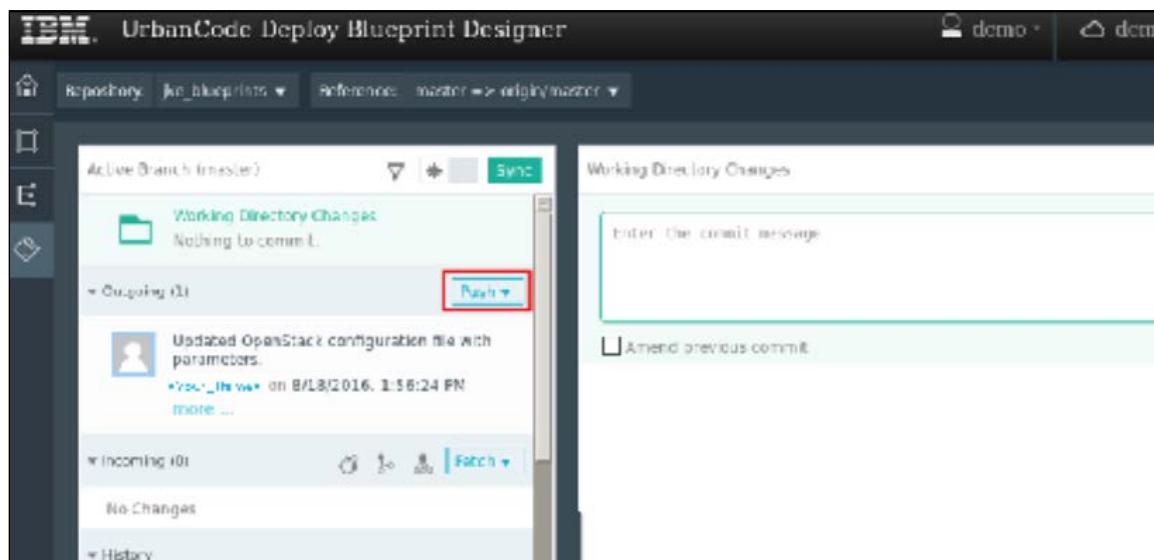
- From the left navigation, click **Repository**. Notice the file change in your working directory.



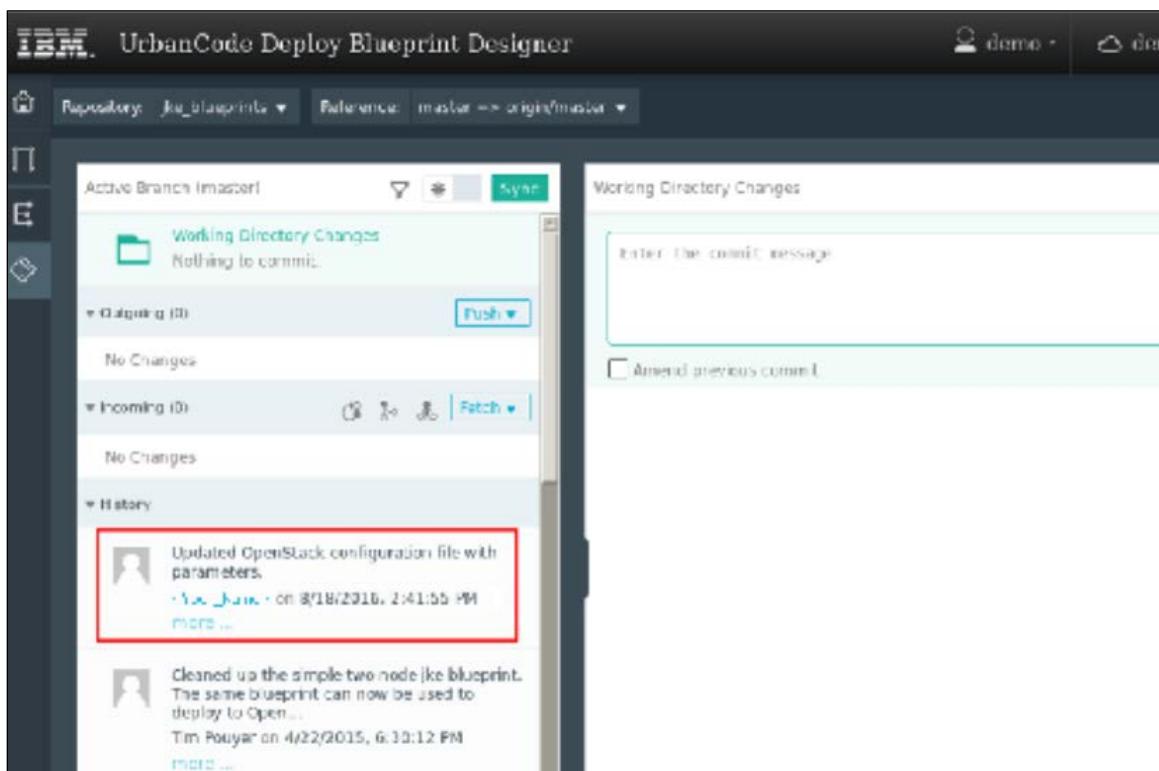
- In the **Enter a commit message** field, type Updated OpenStack configuration file with parameters.
- Under the message field, click **more**, and do the following steps:
 - In the Author section, type your name and email address.
 - In the Committer section, type your name and email address for the email.
 - Select the check box next to the changed file.



4. Click **Commit**.
5. To send the changes to the remote repository, click **Push**.



6. Type your user name and password, and click **Submit**.
Now your change is pushed to the remote repository, and you can see the version in the History section.



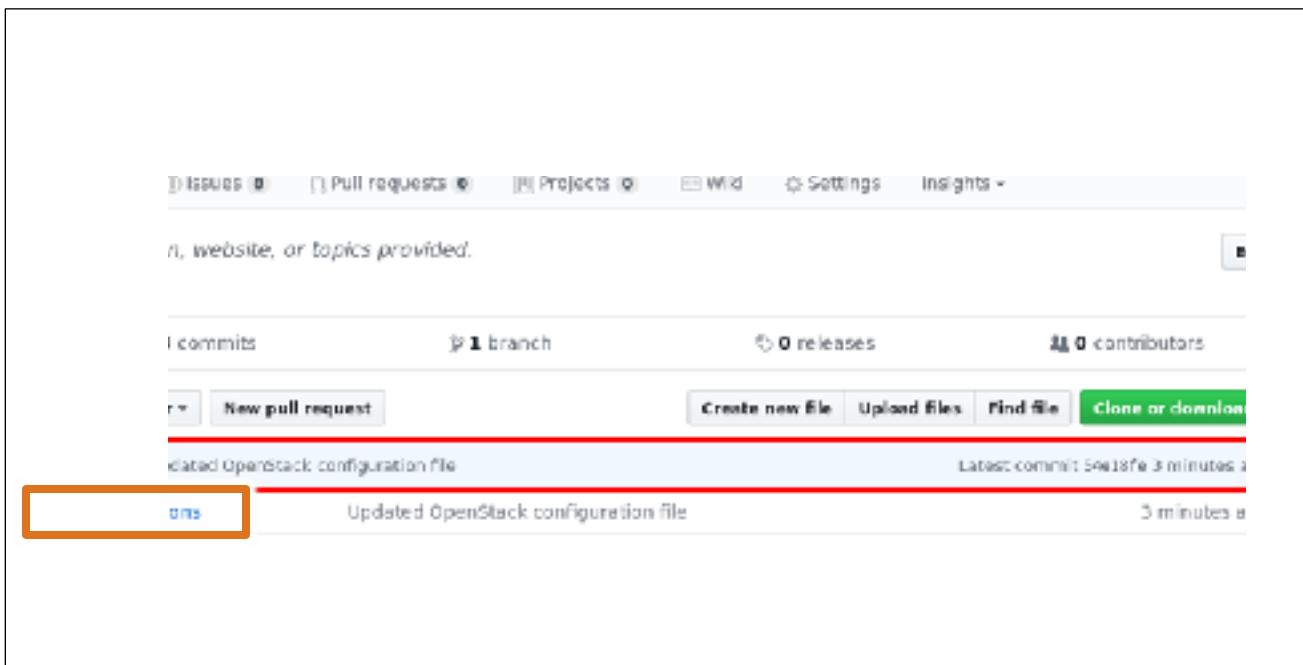
Task 2. Verify changes in the remote repository.

You can see recent project activity from your remote project.

1. Switch back to the tab with your GitHub project and refresh the browser.
From the **Code** tab, you see the latest changes to the project.

The screenshot shows the GitHub repository 'jke_blueprints'. At the top, there are tabs for Issues, Pull requests, Projects, Wiki, Settings, and Insights. Below that, there are buttons for Watch, Star, and Fork. The main area shows a commit history with three entries. The first entry is a pull request from 'jke' titled 'Updated OpenStack configuration file' with a timestamp of '3 minutes ago'. The second entry is a commit from 'jke' titled 'Updated OpenStack configuration file' with a timestamp of '3 minutes ago'. The third entry is a commit from 'node-jke' titled 'Cleaned up the simple two node jke blueprint. The same blueprint can ...' with a timestamp of '2 years ago'. At the bottom, there are buttons for Create new file, Upload files, Find file, and Clone or download.

2. To see the files, click **configurations**.



3. In the configurations folder, click the **openstack-linux.yml** file.



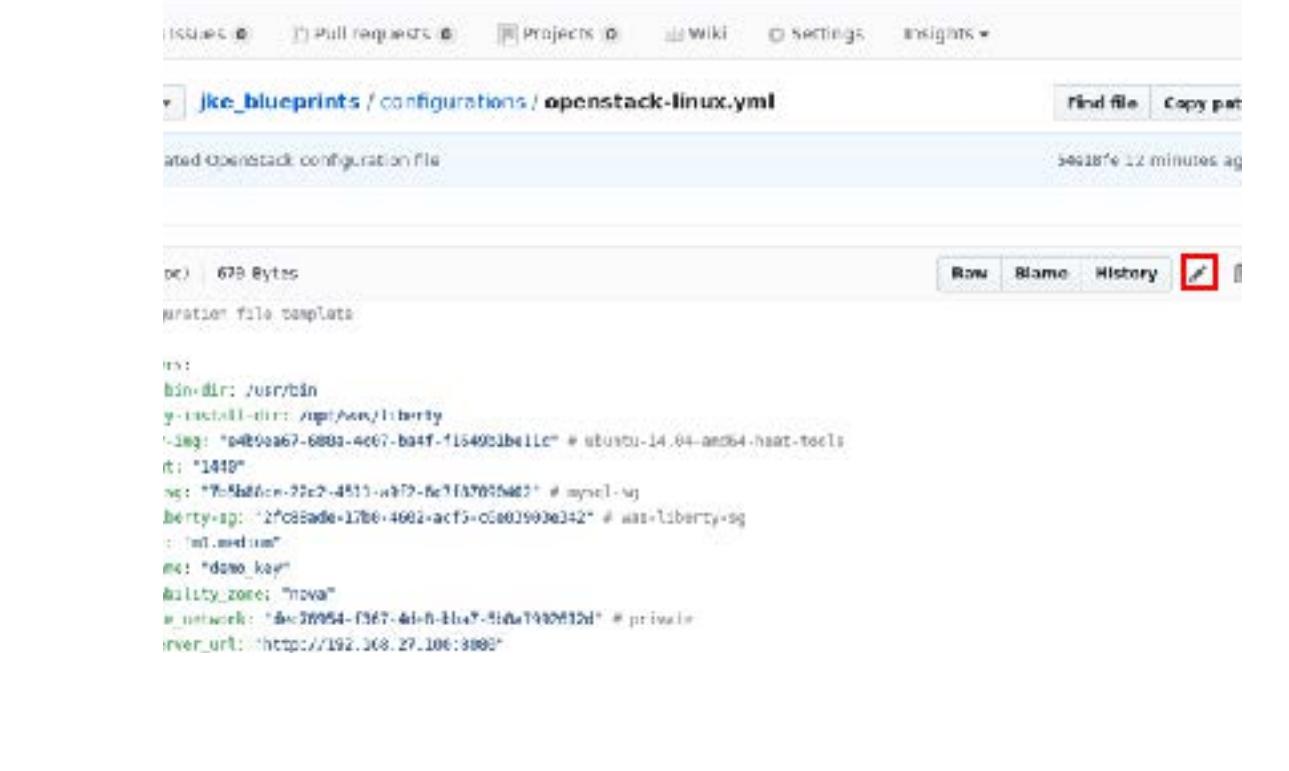
4. You can see the updated file with the substituted parameters:

```
sql_bin_dir: /usr/bin
 liberty_install_dir: /opt/was/liberty
 rver_img: "eab3ac0b-688d-4c6b-ba4f-1048561c1c" # ubuntu 14.04 amd64 final tools
 recent: "1440"
 sql_sg: "7a888c22c2-4511-9af2-ecf8/093482" # mysql_sg
 s Liberty_sg: "21c09ade-17b0-4502-ac15-c5e03903e342" # was-liberty_sg
 avor: "m1.medium"
 y_name: "demo_key"
 aility_zone: "nova"
 ivate_network: "dec75954-1367-4de6-bba7-5b0e7992612d" # private
 d_server_url: "http://192.168.27.100:8080"
```

Task 3. Push changes to local repositories.

Make a change in the remote repository and push it to your local copy in the Blueprint Designer.

1. To edit a file, click **Edit Code**.



```
name: jike_blueprints / configurations / openstack-linux.yml
Last updated OpenStack configuration file
54618f6 12 minutes ago

pc | 679 Bytes
Creation file samples

private:
  bin-dir: /usr/bin
  y-install-dir: /opt/aww/liberty
  v-img: "0420ea67-688d-4c67-ba4f-f1549b1be11c" # ubuntu-14.04-amd64-haas-tools
  vlt: "1410"
  vsg: "9e5d87cc-22e2-4511-a972-8e91a70754d2" # mysql-vsg
  liberty-vsg: "2fc85ade-17b6-4602-acf5-c5603993e342" # waz-liberty-vsg
  vif-medium
  key: "demo_key"
  ability_zones: "nova"
  v-network: "5c38954-0367-4d5b-bbaf-51da1980812d" # private
  vcenter_url: "http://192.168.27.106:9000"
```

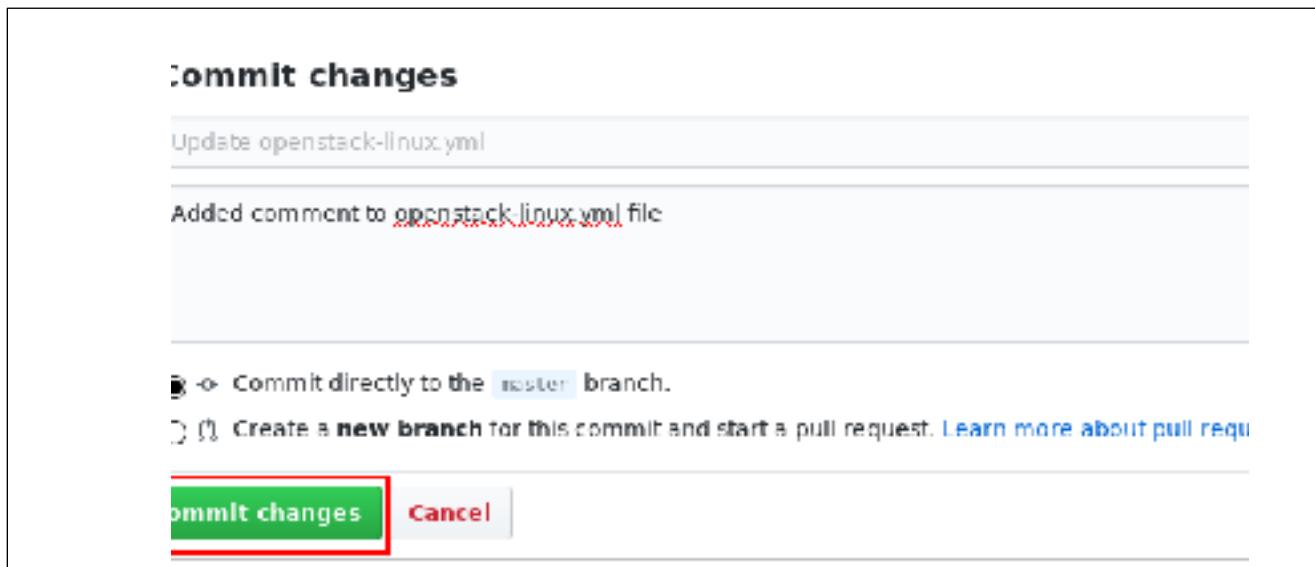
2. In the second line of the file, type `# Updated configurations for OpenStack 08/2016.`

```

|inters:
|ql-bin-dir: /usr/bin
|erty-install-dir: /opt/was/liberty
|ver-img: "eab6ea67-588a-4c87-ba1f-f16d481b11c" # ubuntu-14.04-andal-libre-tools
|outf: '1448'
|ql_sg: "7bb88cc2-22c2-4811-8f72-bc7fe7090402" # mysql_sg
|Liberty_sg: "21c88ade-17b0-4592-ac15-c6e63933c342" # was Liberty_sg
|wrm: "ml.medium"
|name: 'demo_key'
|ability zone: 'nova'
|vate_network: "dec78854-f367-4de8-bb07-3b807992612d" # private
|_server_url: "http://192.168.27.100:8080"
|_user: 'edrin'
|_password: 'admin'
|_relay_url: 'None'
|lic_network: "4n4f/b4a-595b-4396-a7dc-efdcf2bae367" # public

```

3. Scroll to the bottom of the page.
4. In the Commit Changes section, type Added comment to openstack-linux.yml file in the comment box.

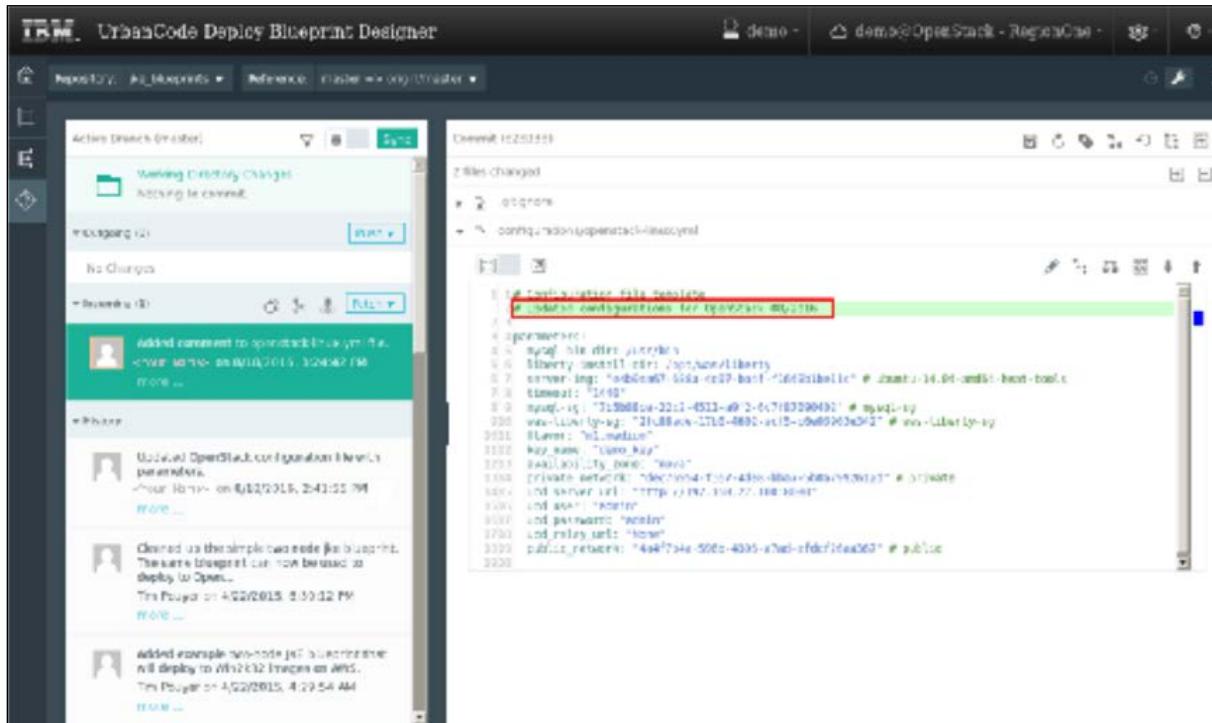


5. Click **Commit changes**.

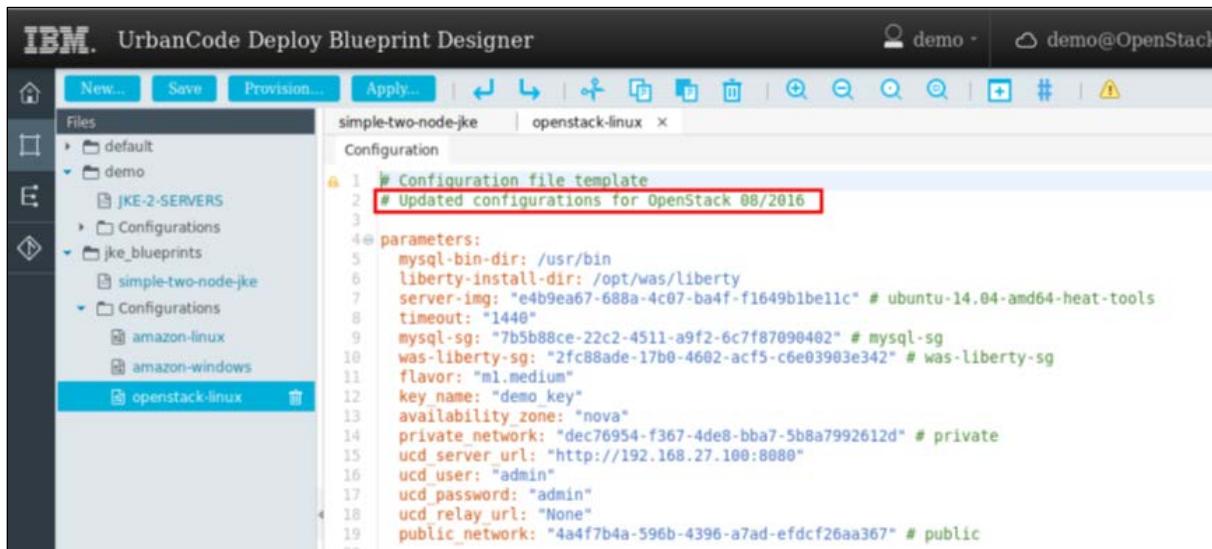
Task 4. Verify changes in the local repository.

Accept and verify the changes in your local repository.

1. Switch back to the **Blueprint Designer** tab.
2. In the Incoming section, click **Fetch** to get the incoming change.
3. Click the incoming change and verify that the note was added.



4. Click **Sync** to update the local file.
5. From the left navigation, click **Blueprints**.
6. Open the **openstack-linux** file, and verify that the change is in your local repository.



In this unit, you explored the integration of the Blueprint Designer and the Git version control system. You learned to fork a project and clone a repository into a local workspace. You also worked with files change on both the local and remotes sides of a repository.



IBM Training



© Copyright IBM Corporation 2017. All Rights Reserved.