Course Guide

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

Course code UCD62 ERC 1.0

IBM Training

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.

Contents

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:

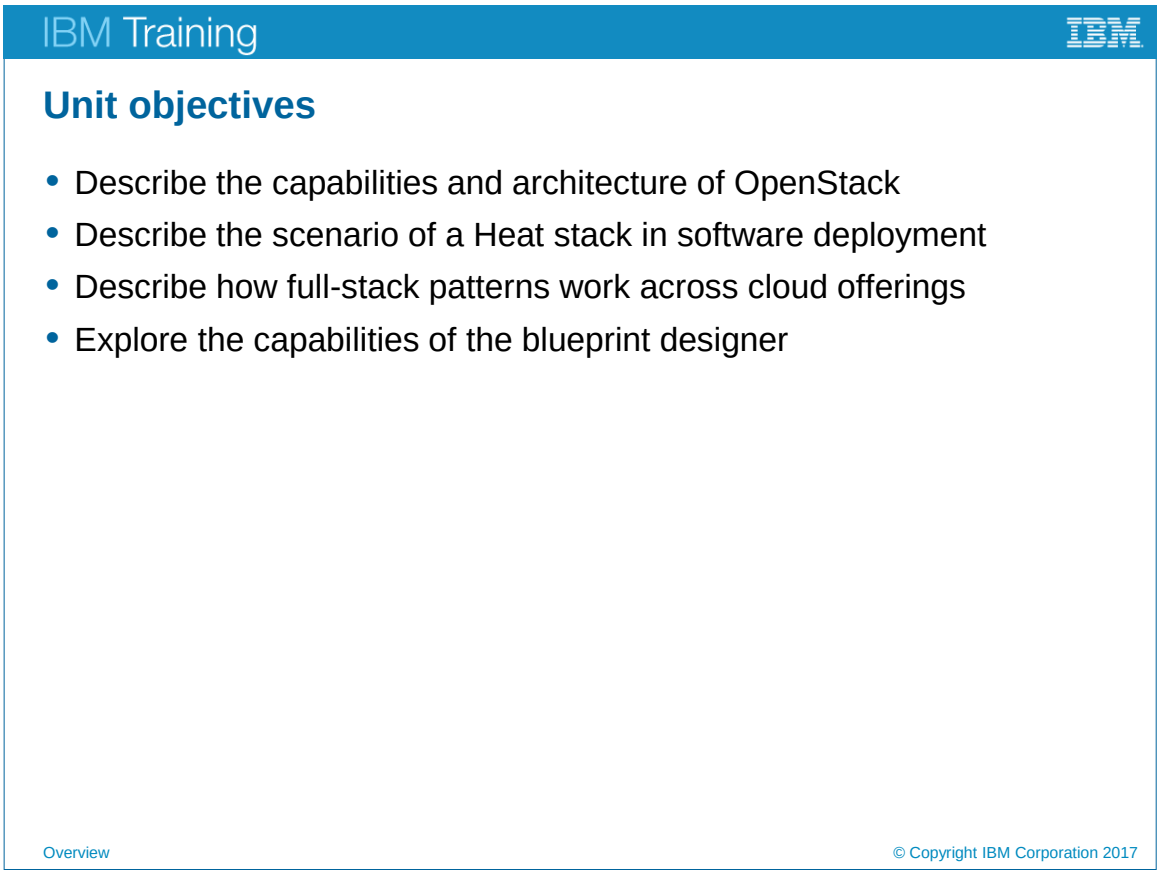
- Instructor-led training in a classroom or online
- Self-paced training that fits your needs and schedule
- Comprehensive curricula, learning journeys, and training paths that help you identify the courses that are right for you

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:	
	• IBM Cloud Education Resource Guide	• www.ibm.biz/CloudEduResources
	• Online support	• https://www.ibm.com/support/home/
	• IBM Website	• http://www.ibm.com

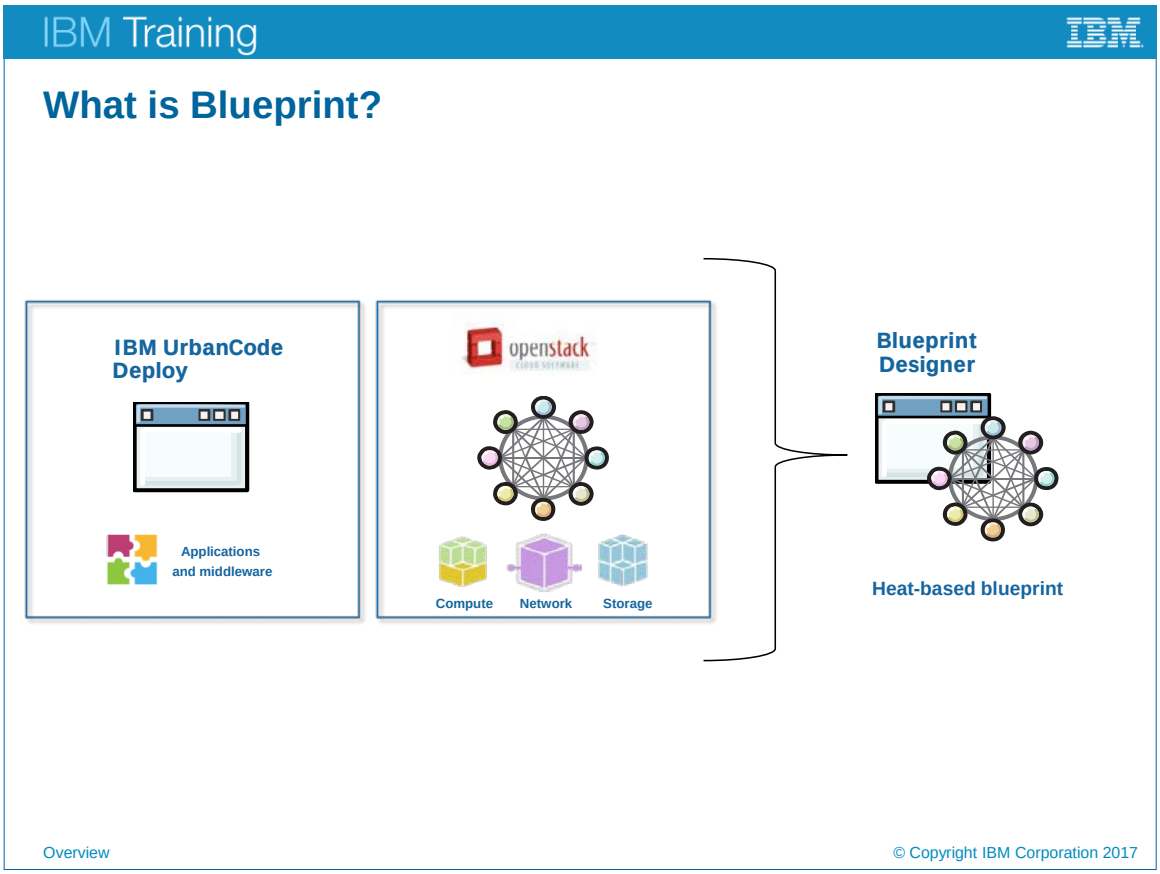
IBM Training

Overview



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.

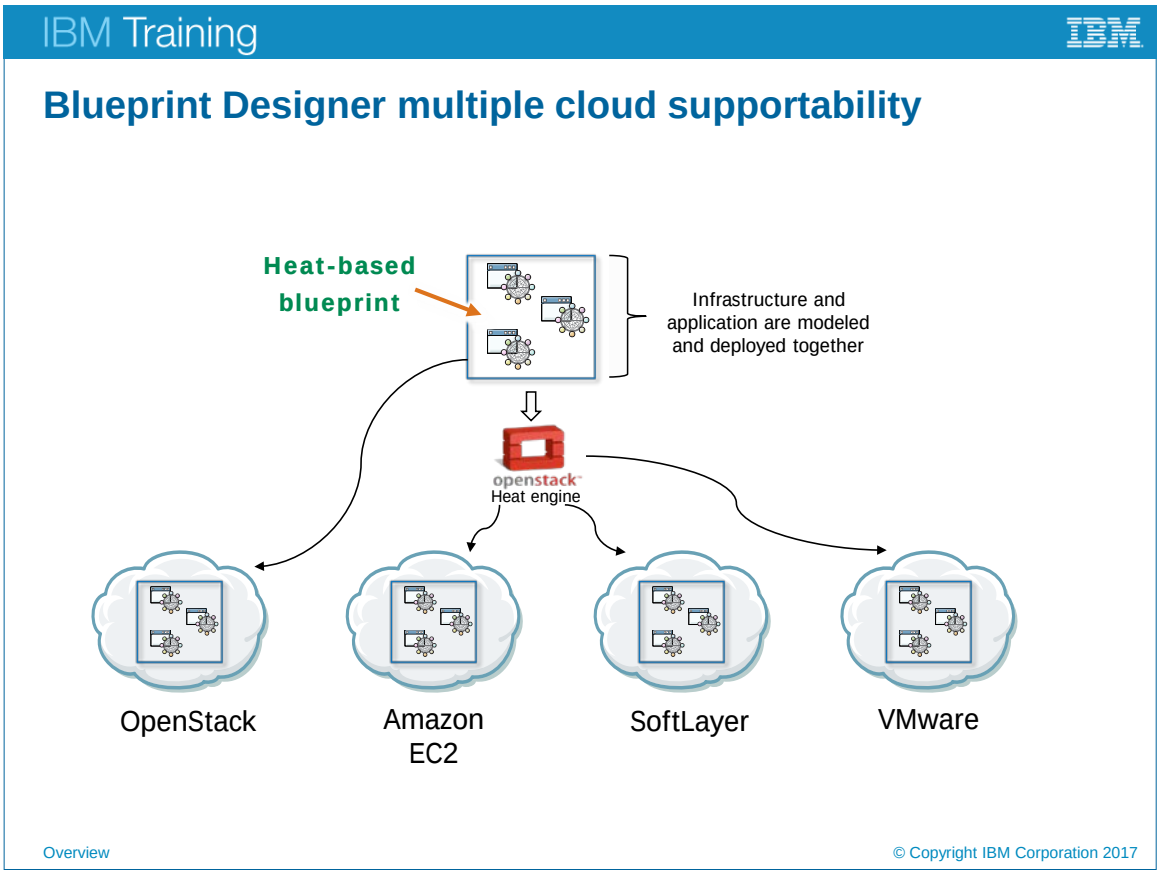


What is Blueprint?

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.

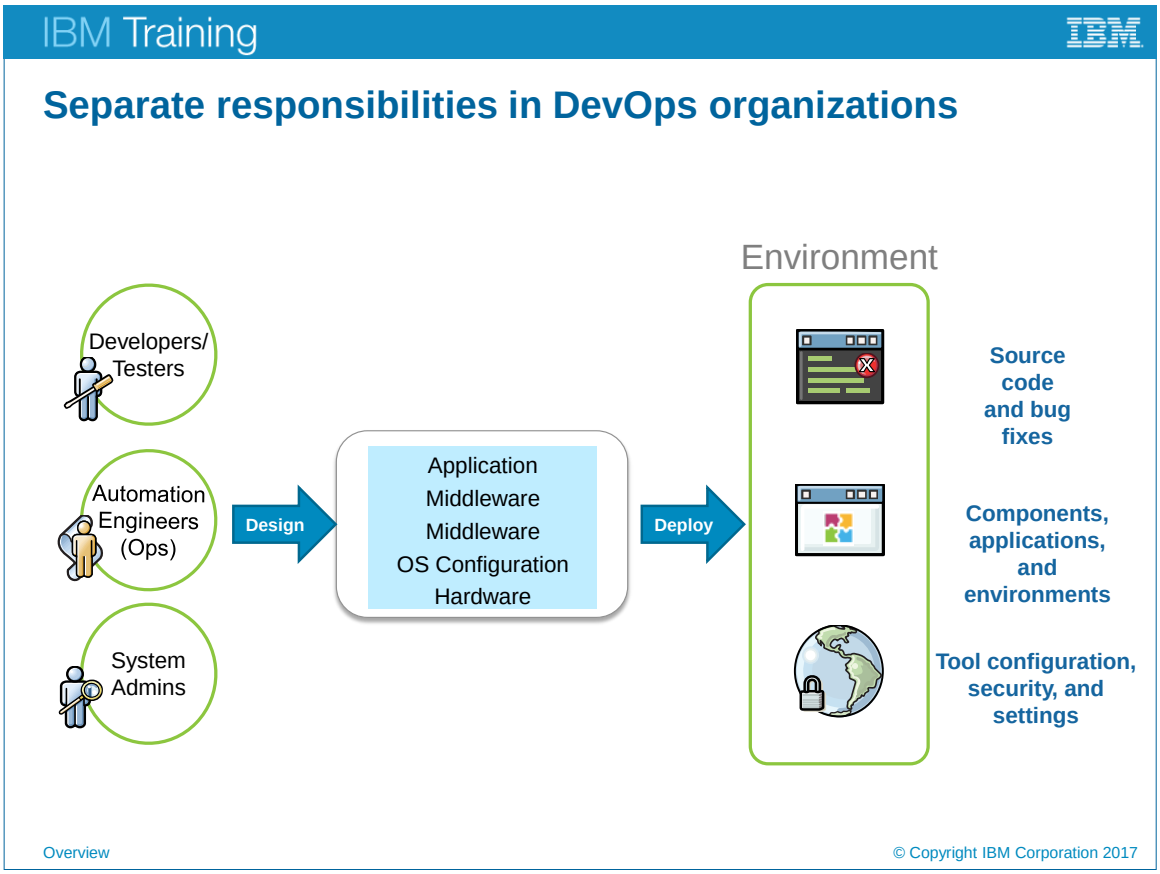
You can create or update a cloud environment based on a blueprint. The blueprint design server runs this provisioning or updating by using a Heat engine. If the blueprint includes software components, the provisioning or updating process also deploys those components through the automation processes 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

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.



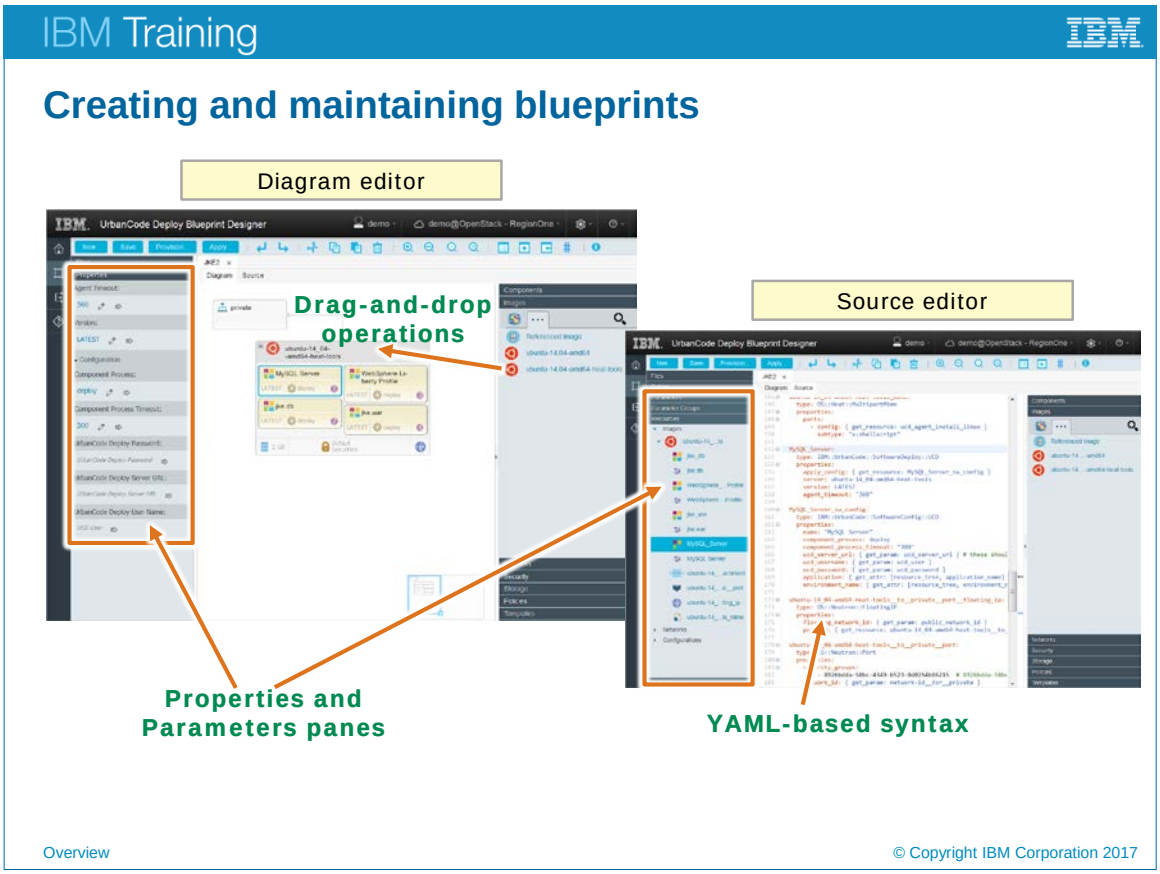
Separate responsibilities in DevOps organizations

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.



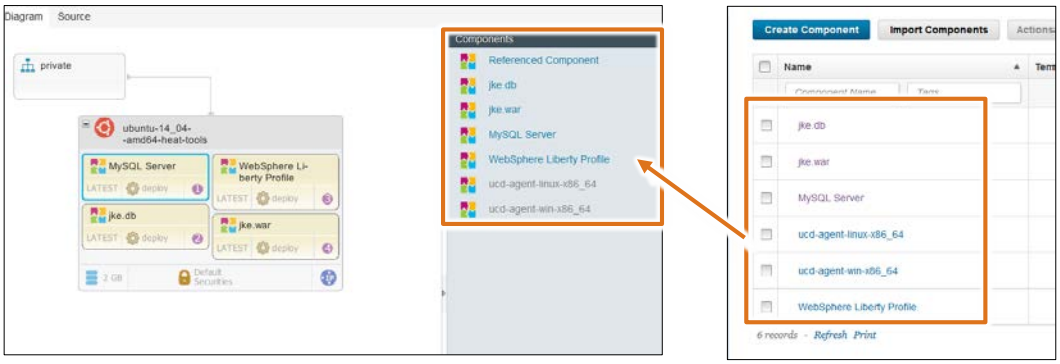
Creating and maintaining blueprints

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.

IBM Training

Application components accessibility

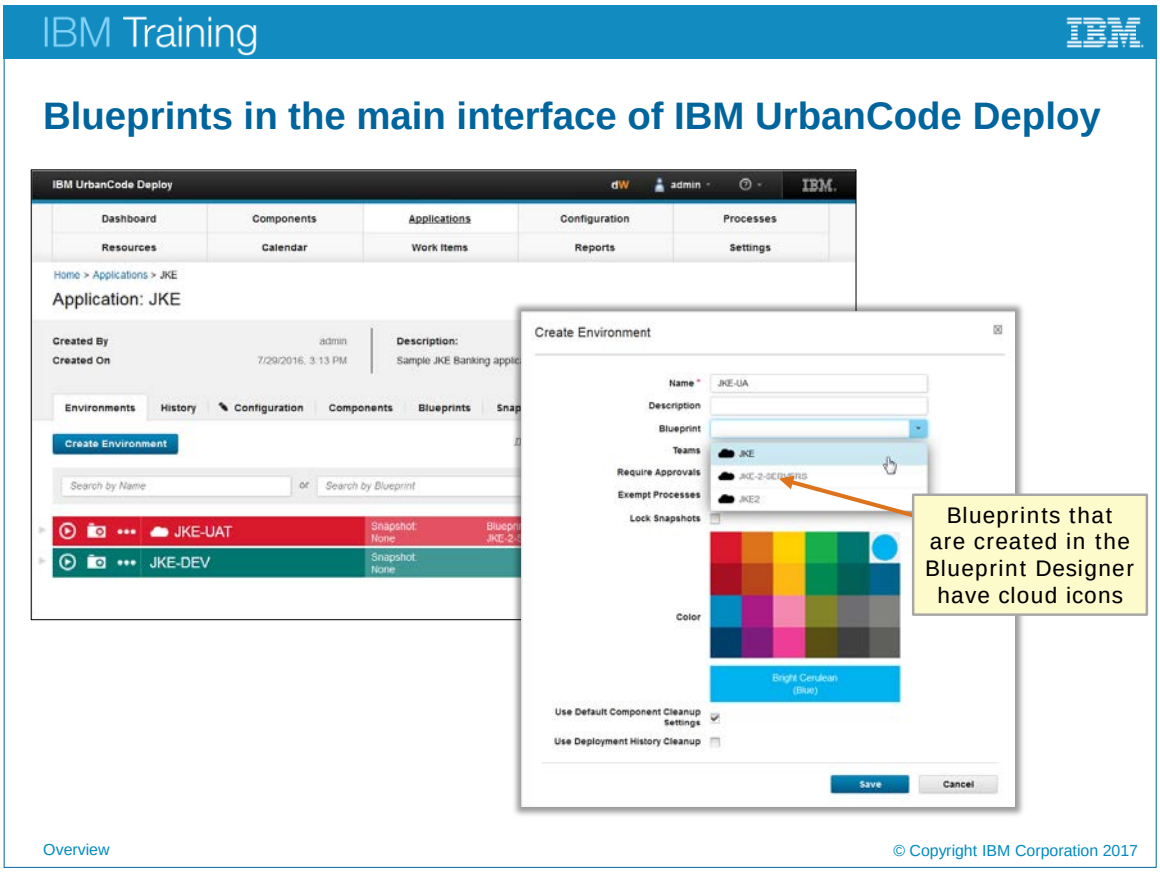


Components in IBM UrbanCode Deploy

Overview

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.

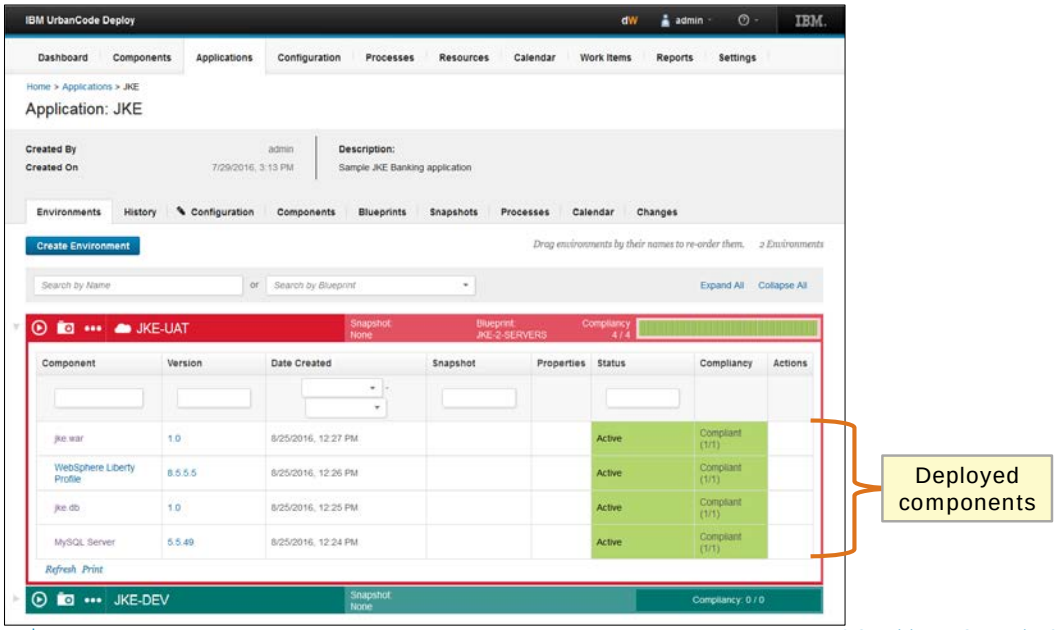


Blueprints in the main interface of IBM UrbanCode Deploy

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.

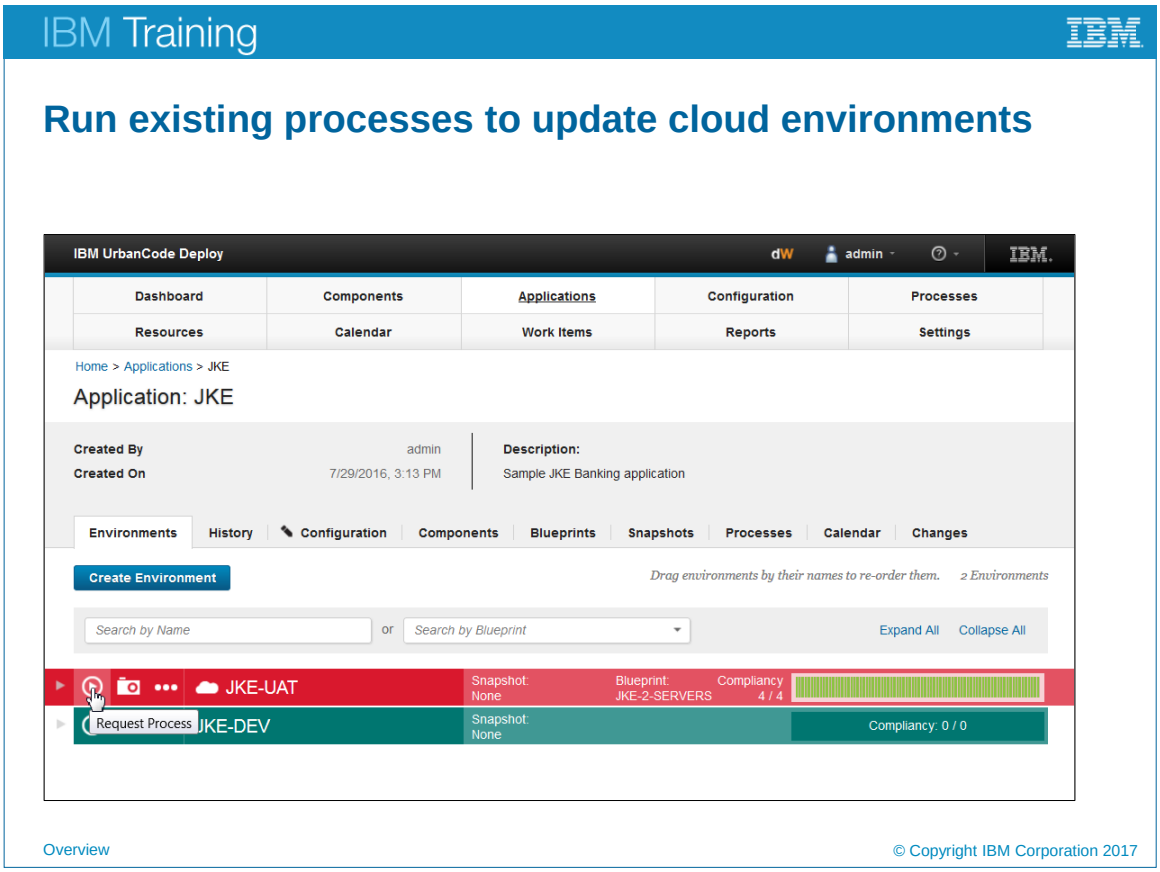
IBM Training

Components deployed during environment creation



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.



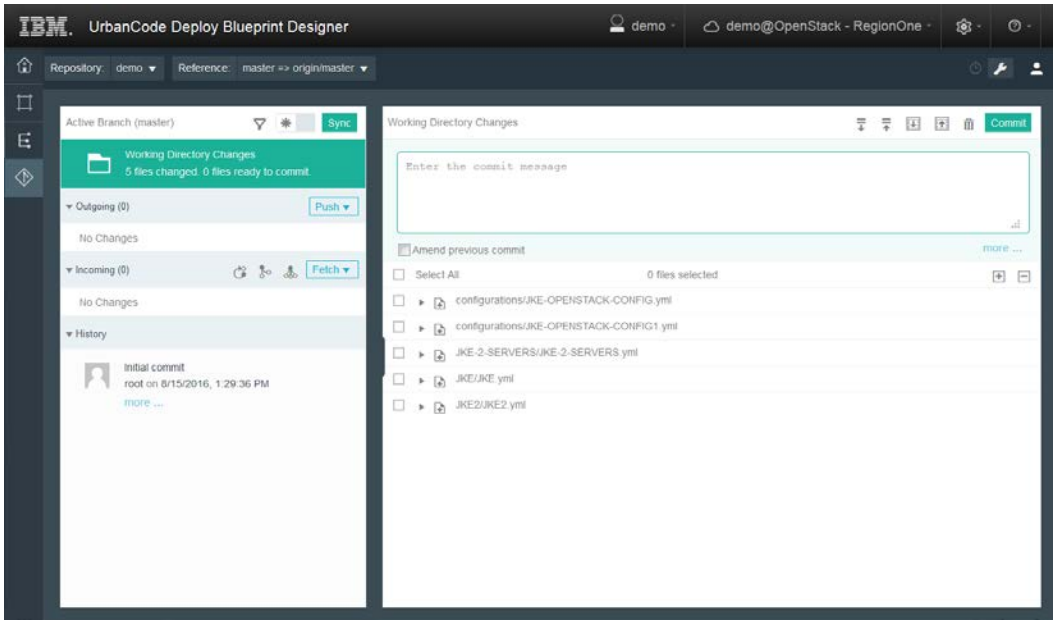
Run existing processes to update cloud enivronments

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.

IBM Training

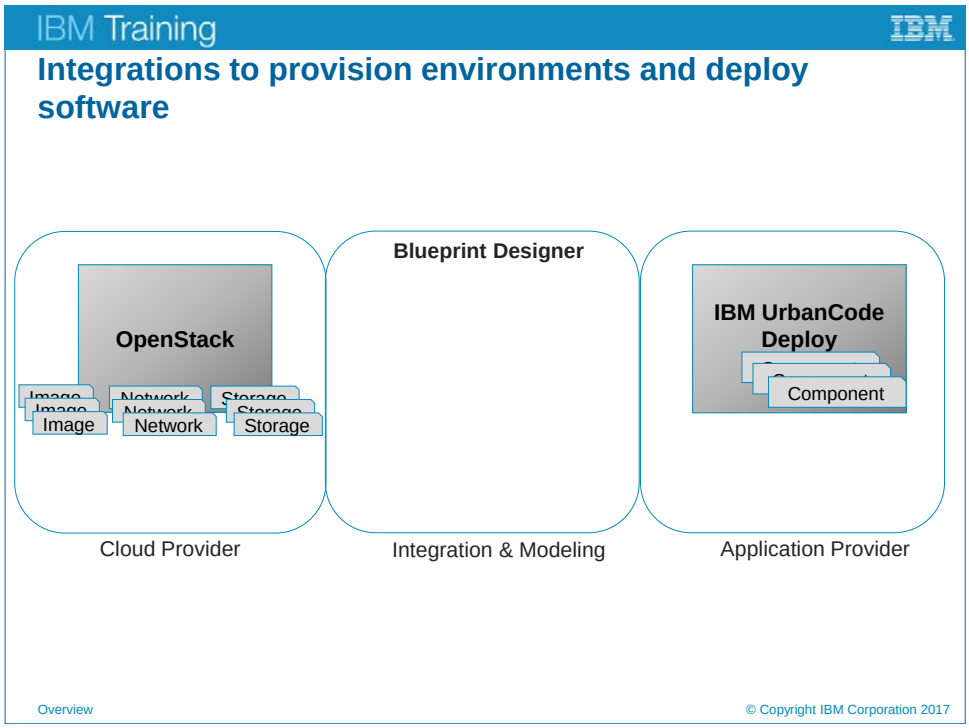
Blueprints in Git repositories



Overview

Blueprints in Git repositories

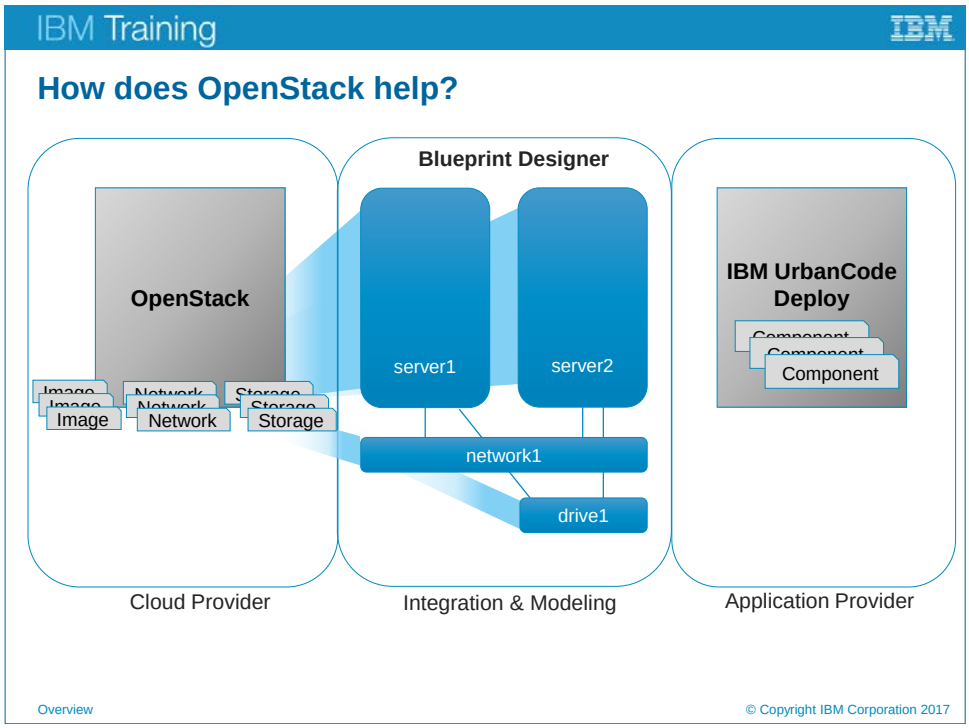
The Blueprint Designer includes a Repositories view where you can treat blueprints like source code. You can create and clone Git repositories, commit changes, and check out previous versions. From a default repository that is automatically created with the product installation, you can create and provision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



Integrations to provision environments and deploy software

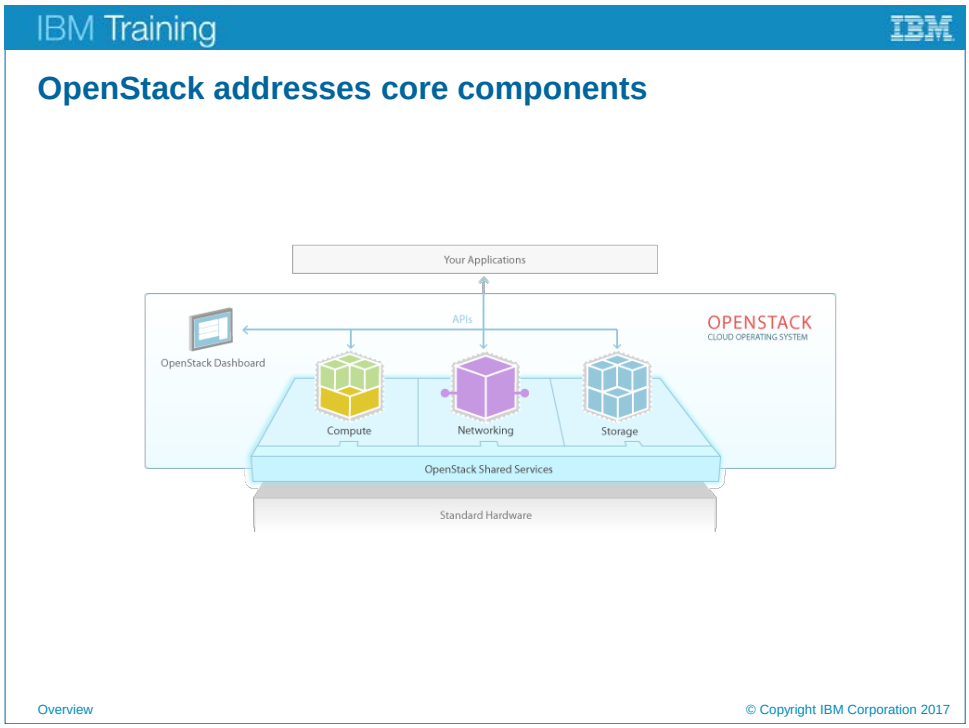
Integrations provide the Blueprint Designer with the resources to provision environments and deploy software. A Heat stack is the collection of resources that Heat creates. This collection might include instances (virtual machines), networks, subnets, routers, ports, router interfaces, security groups,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



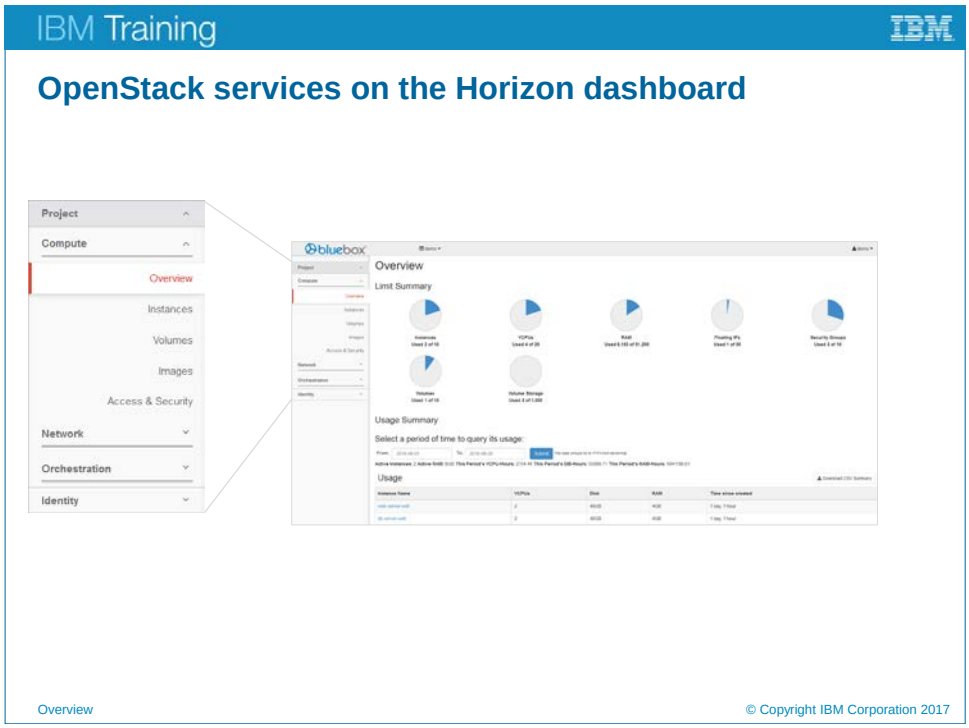
How does OpenStack help?

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.



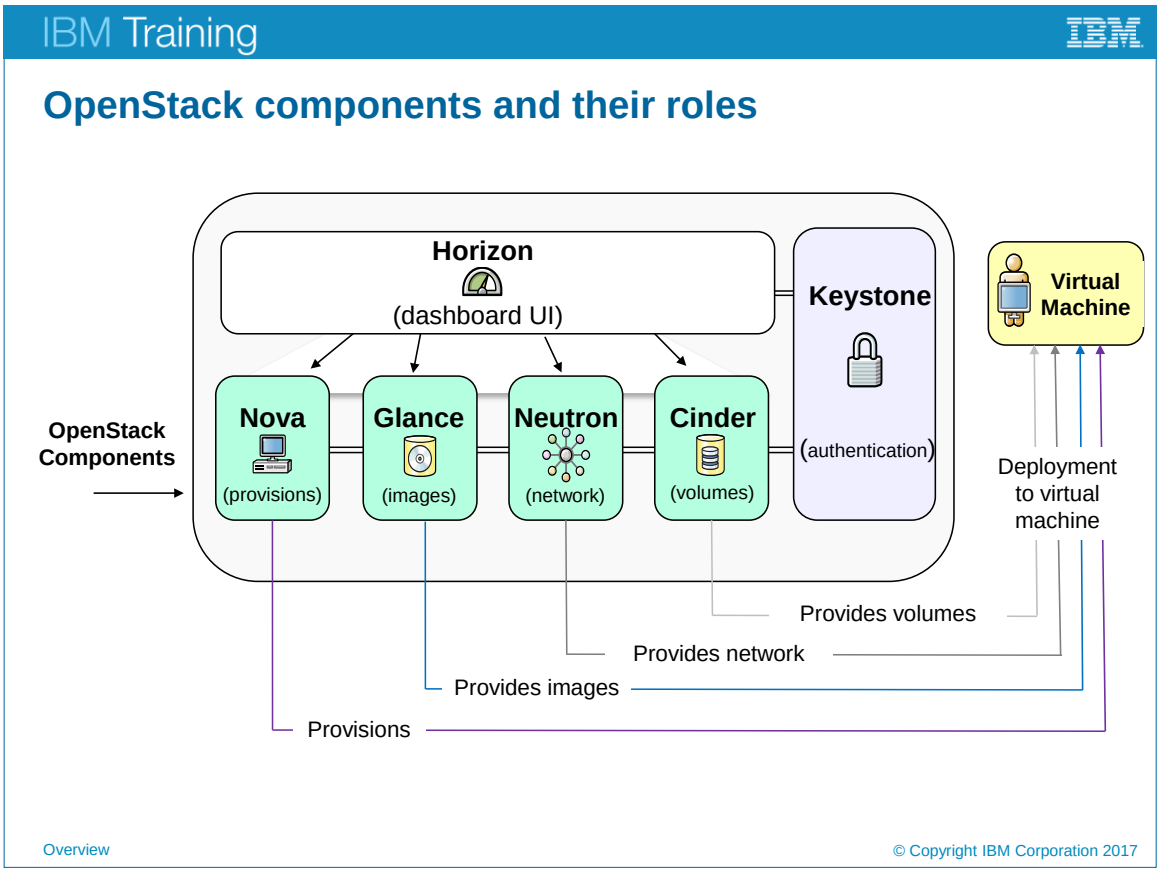
OpenStack addresses core components

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.



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

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:

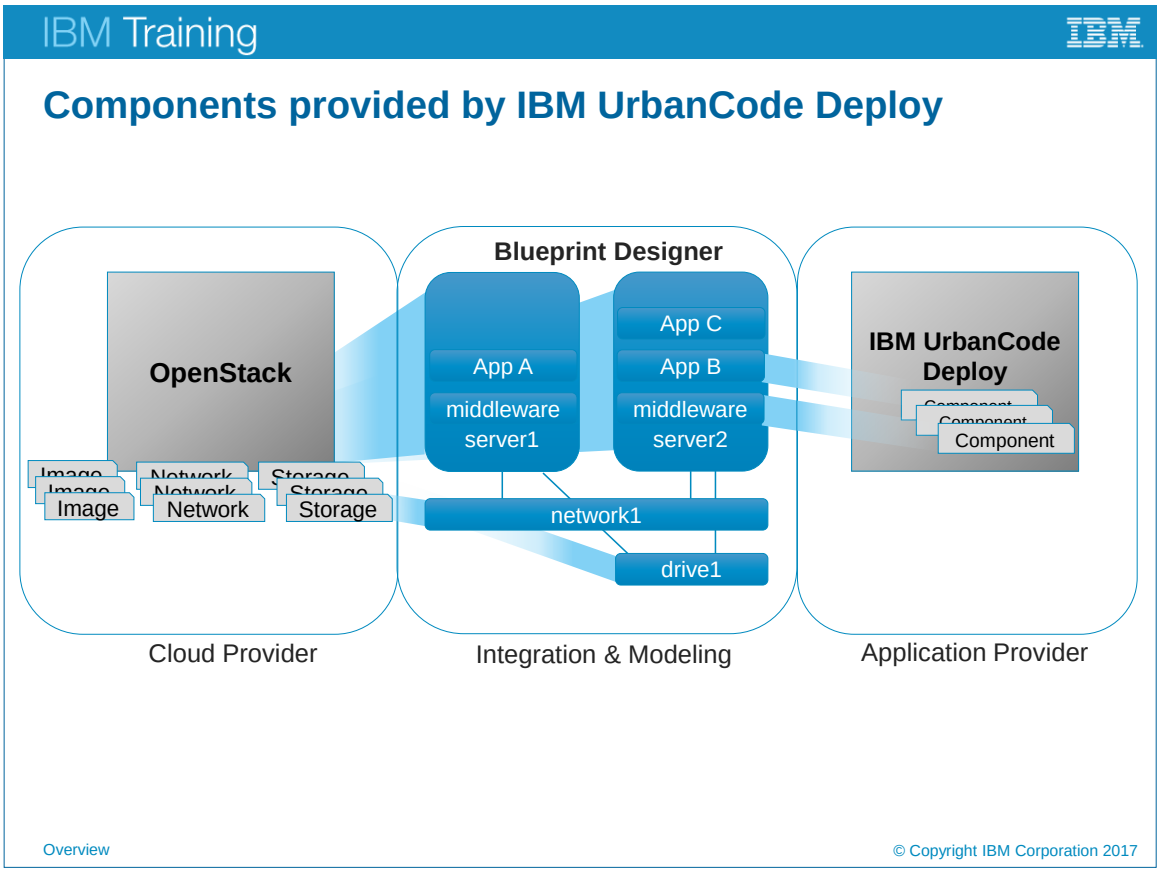
Nova (compute): Creates the virtual machine and manages the lifecycle of compute instances in an OpenStack environment.

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.

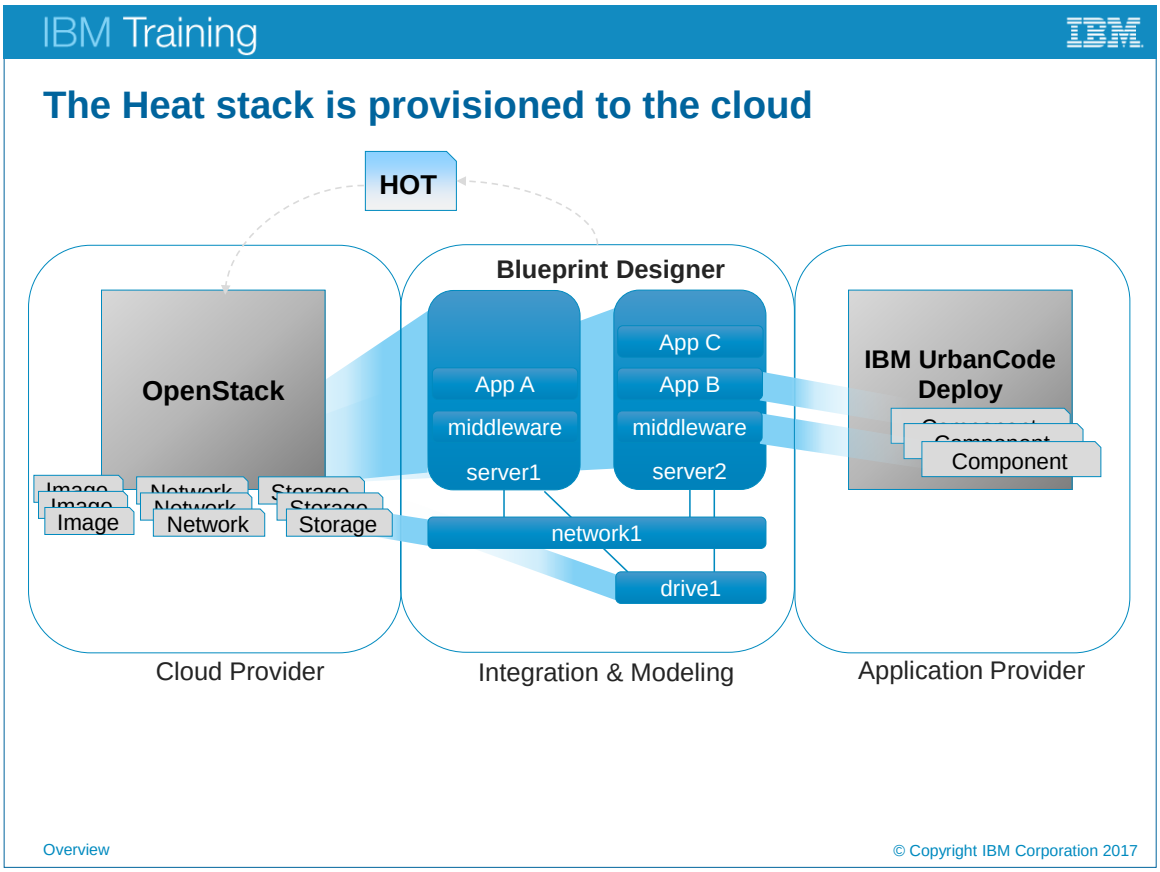
Keystone (identity): Manages users and what they have access to and is typically backed by Active Directory or other LDAP. Provides an authentication and authorization service for other OpenStack services.

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



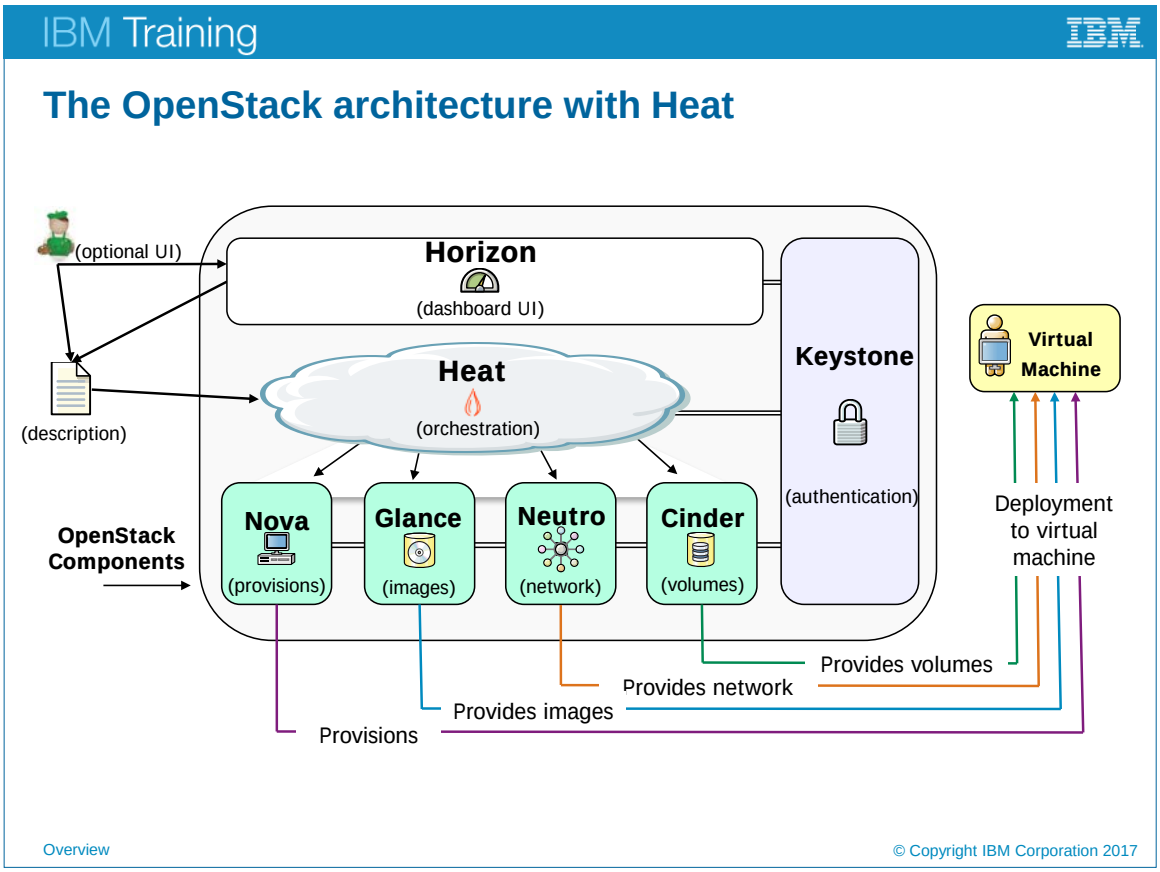
Components provided by IBM UrbanCode Deploy

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



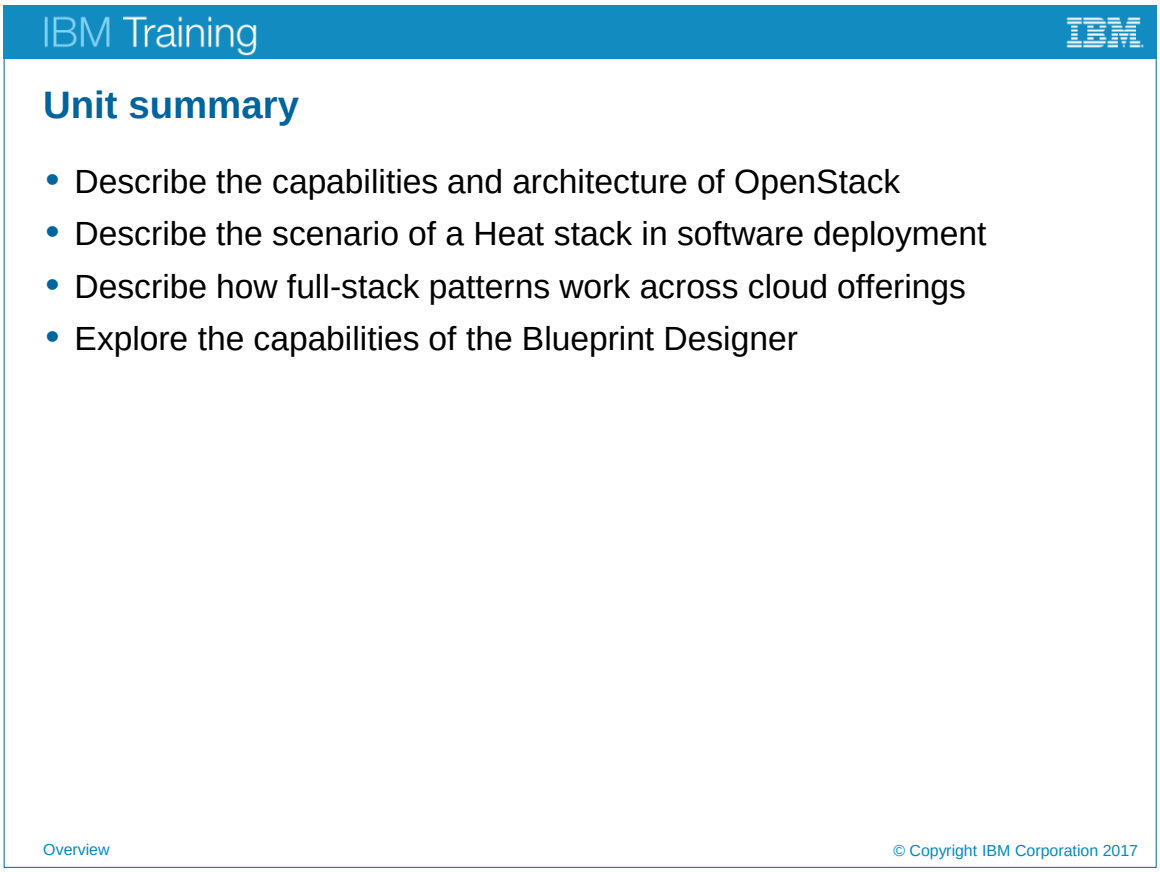
The Heat stack is provisioned to the cloud

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

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



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.

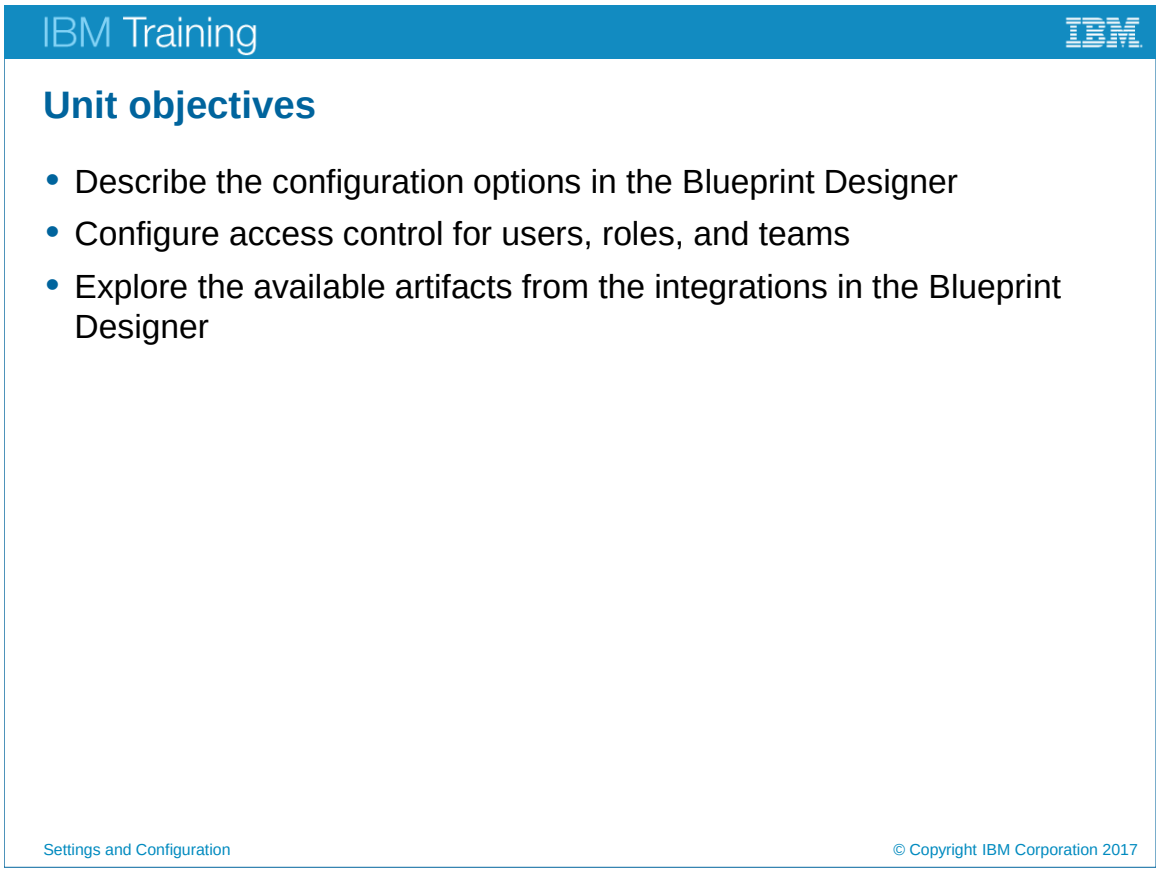
Blueprints provision an environment in the cloud and deploy applications in to that cloud environment in a single step. The Blueprint Designer uses an OpenStack Heat engine to translate blueprints (by using HOT) and make them appropriate for your target cloud.

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

IBM Training

Settings and Configuration

IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1



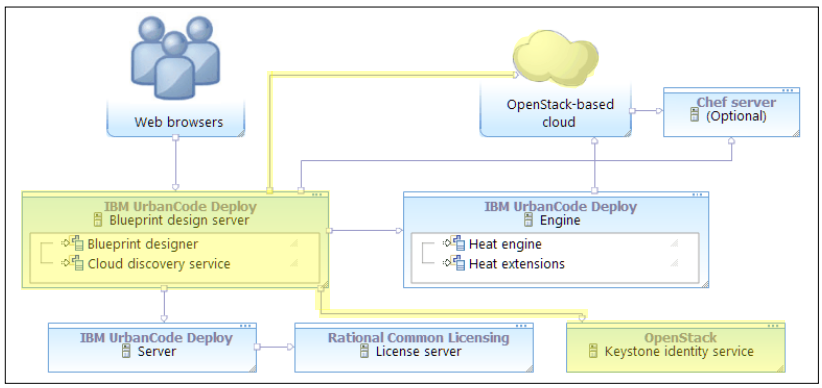
Unit objectives

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.

IBM Training

Blueprint design server

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



Settings and Configuration

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.

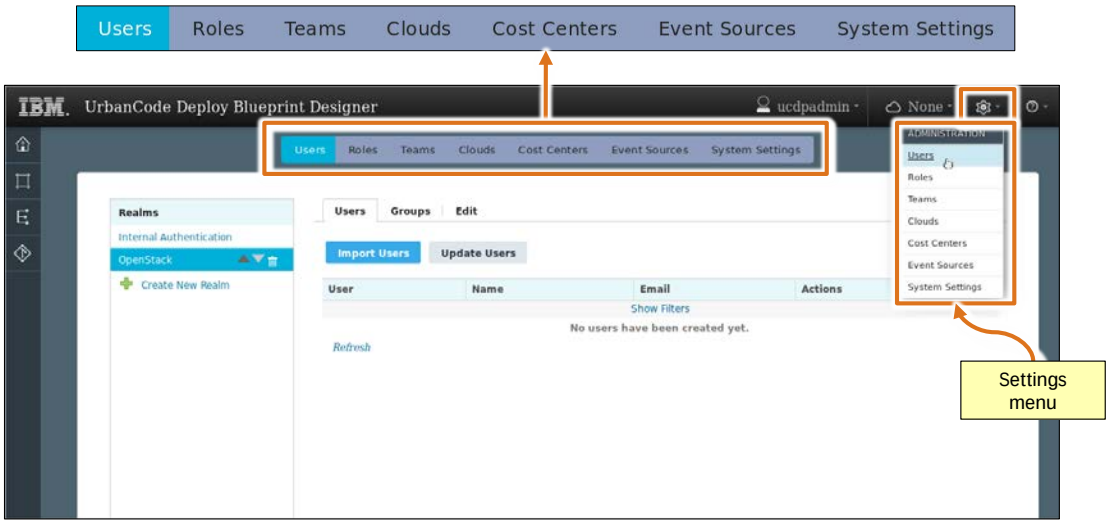
This diagram shows the scenario of when the blueprint design server retrieves account and authentication information from the Keystone identity service. If you are using an OpenStack cloud, the easiest way to set up authentication is to use that cloud's Keystone identity service. In this case, you log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

Blueprint design server configuration

- Configure the blueprint design server from the Settings menu



Settings and Configuration

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.

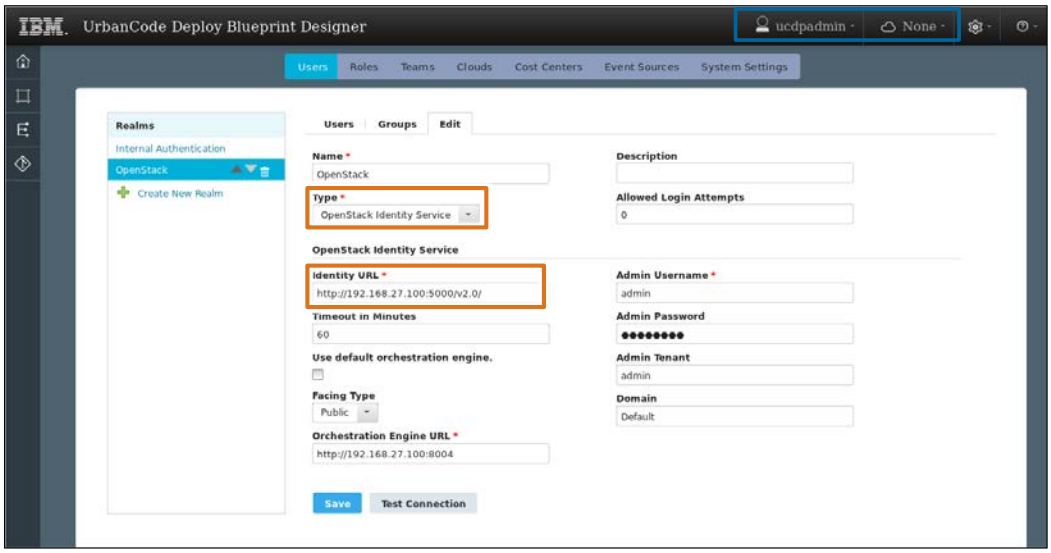
In this course, you explore **Users**, **Roles**, **Teams**, **Clouds**, and **System Settings**:

- **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.

IBM Training

Authentication realm

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



Settings and Configuration

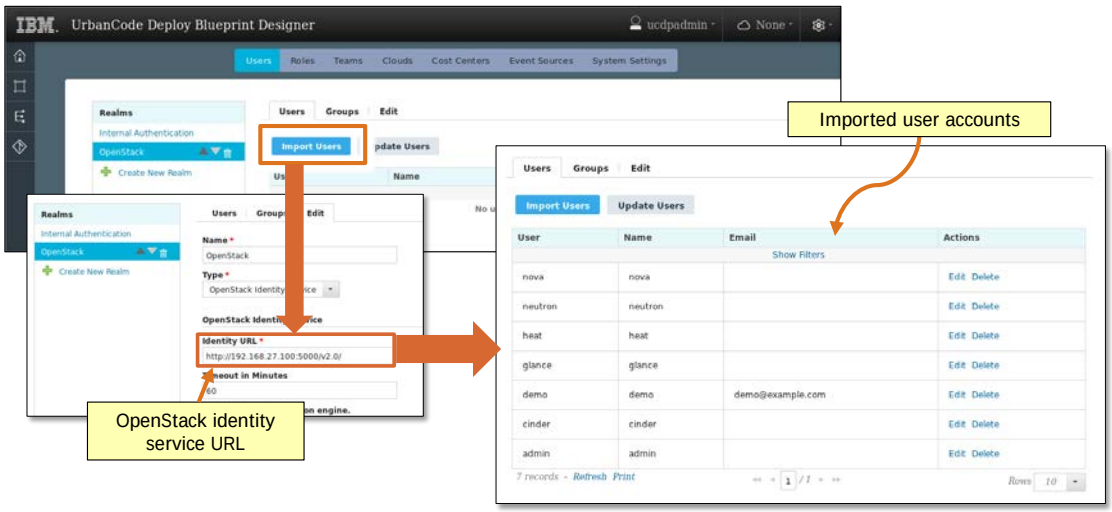
Authentication realm

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.

IBM Training

Importing user accounts

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

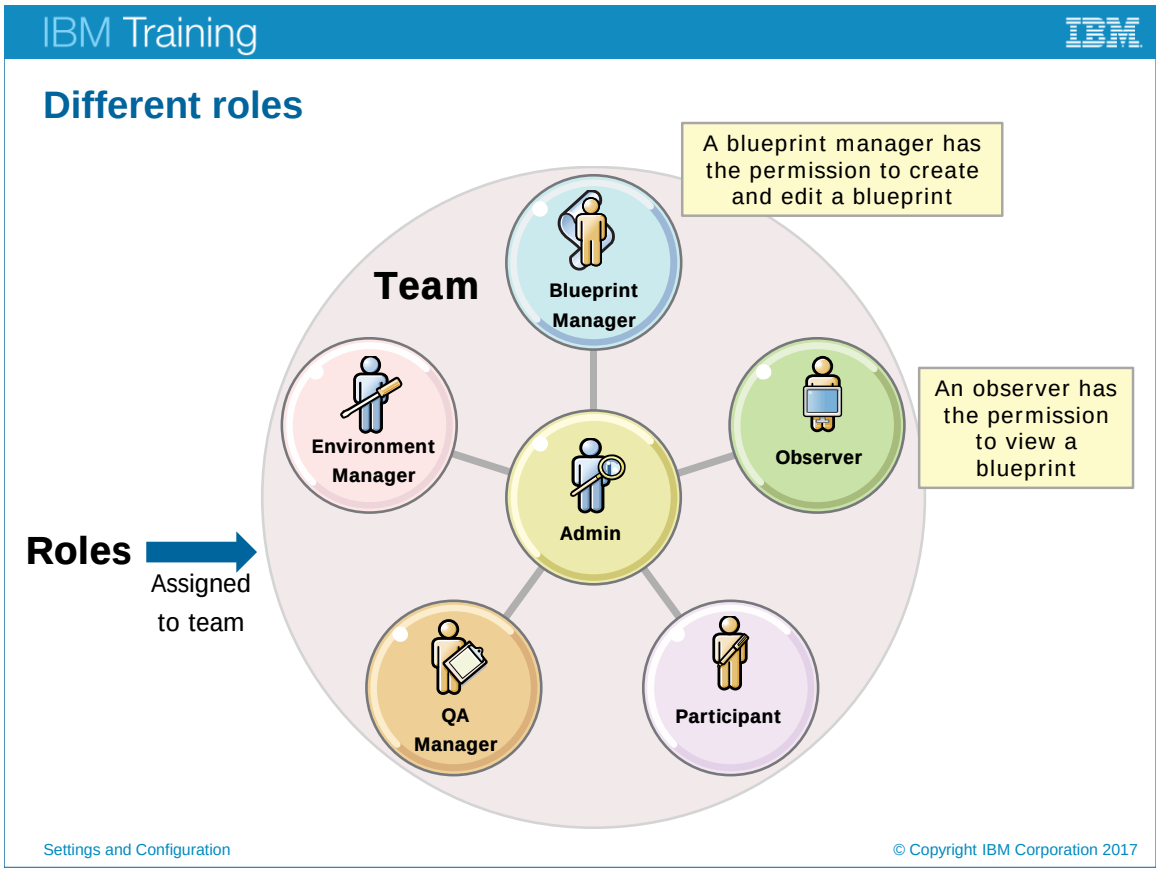


Settings and Configuration

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

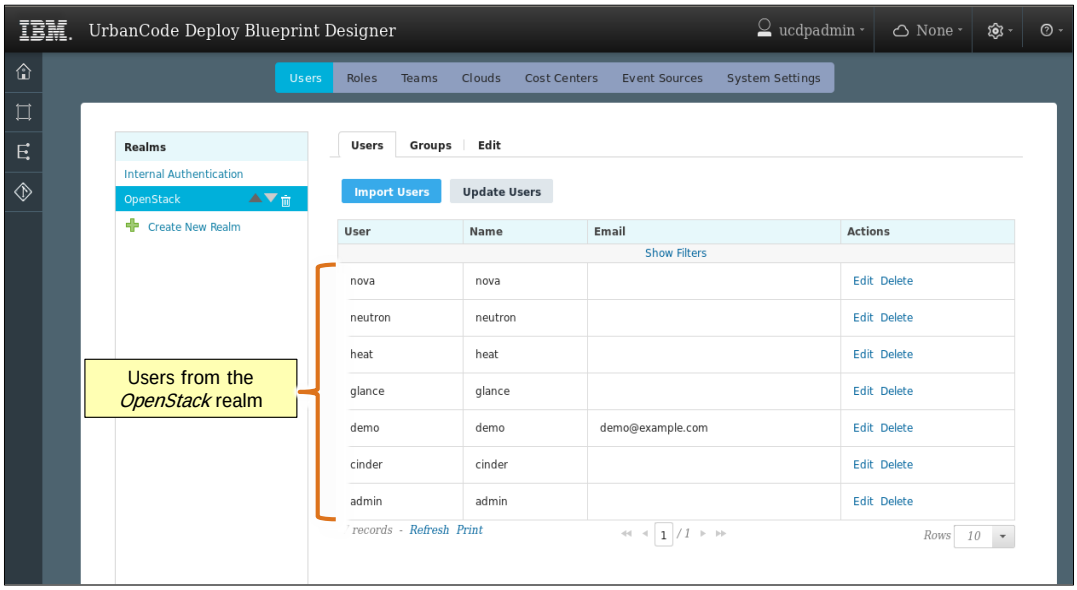
Roles define access to the various capabilities and resources of the blueprint design server. A role is defined by what a user can do to what resources, such as create a blueprint.

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.

IBM Training

Authentication realm

- An authentication realm is a source of information about users



Settings and Configuration

Authentication realm

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

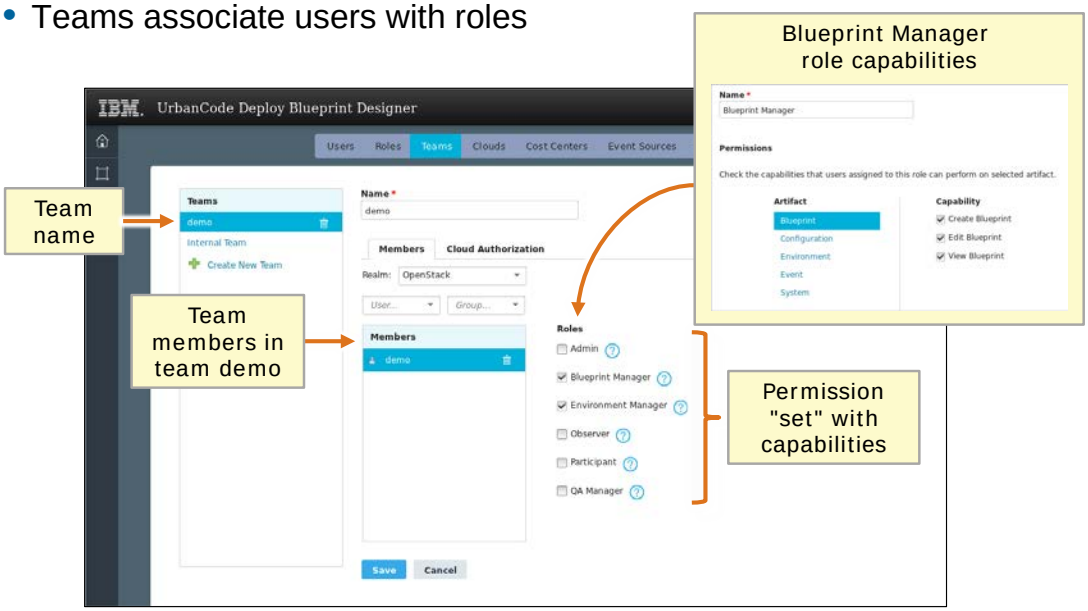
Authentication realms use authorization realms to associate users with groups and to determine user access. IBM UrbanCode Deploy includes an internal database for storing security information and integration with the Lightweight Directory Access Protocol (LDAP). LDAP is a widely used protocol for accessing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

Each user represents an account on the Blueprint Designer.

IBM Training

Users and roles

- Teams associate users with roles



Settings and Configuration

Users and roles

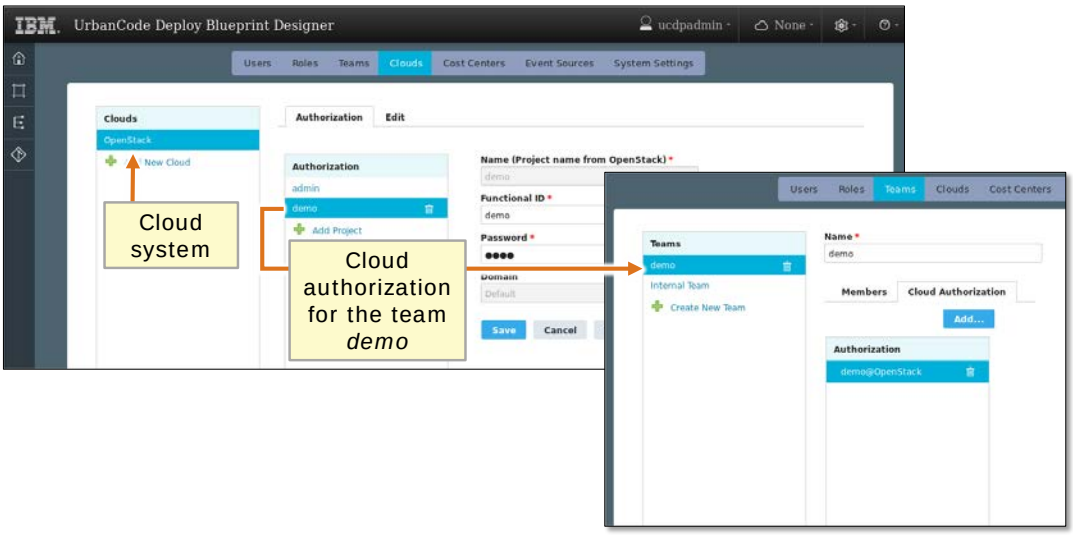
You assign users to teams on a specific role basis. For example, after a user is imported into the blueprint design server, you can assign the user to a team and then assign a role with access to whatever capabilities are defined in that role.

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

User access

- Give users access to cloud systems through cloud projects



Settings and Configuration

User access

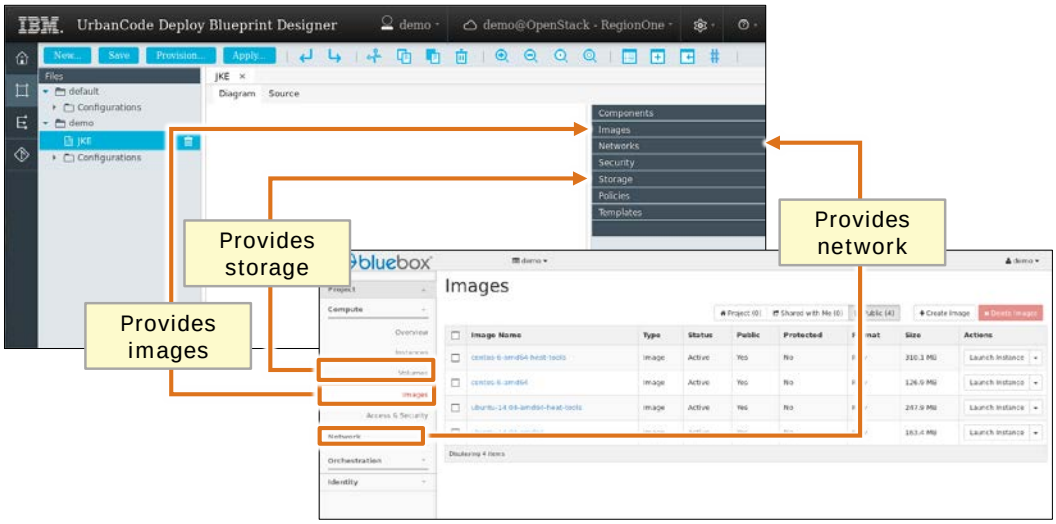
When you import users from the authentication realm, the blueprint design server creates a cloud connection that is based on this authentication realm and cloud projects that are based on the projects on the cloud. A cloud project includes a functional ID with access to a cloud system. When you assign 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.

IBM Training

Demo user access

- The demo user has access to the OpenStack cloud resources

Demo user is logged in



Settings and Configuration

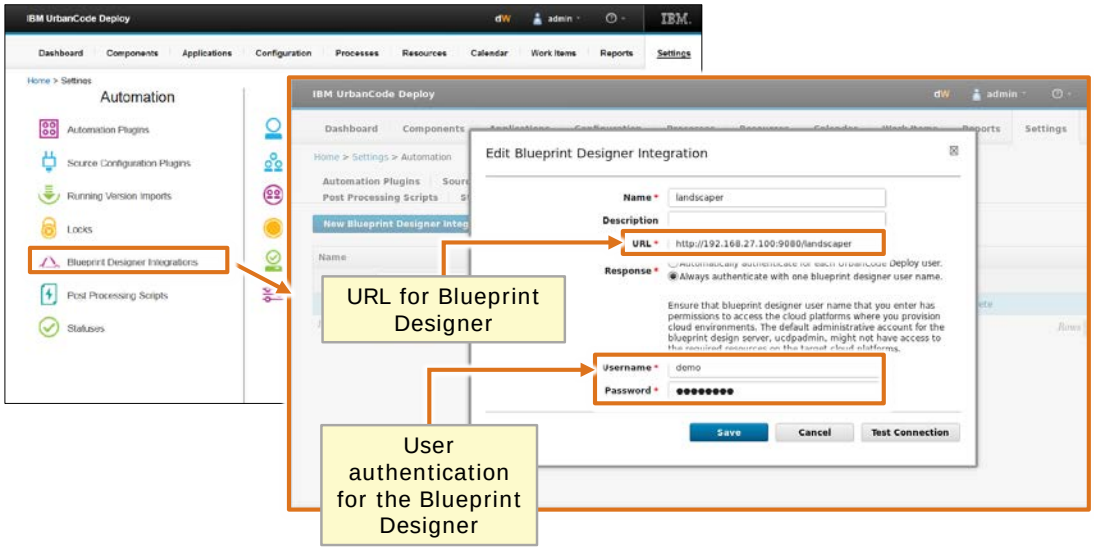
Demo user access

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

Server connections (1 of 2)

- Connect the IBM UrbanCode server to the blueprint design server



Settings and Configuration

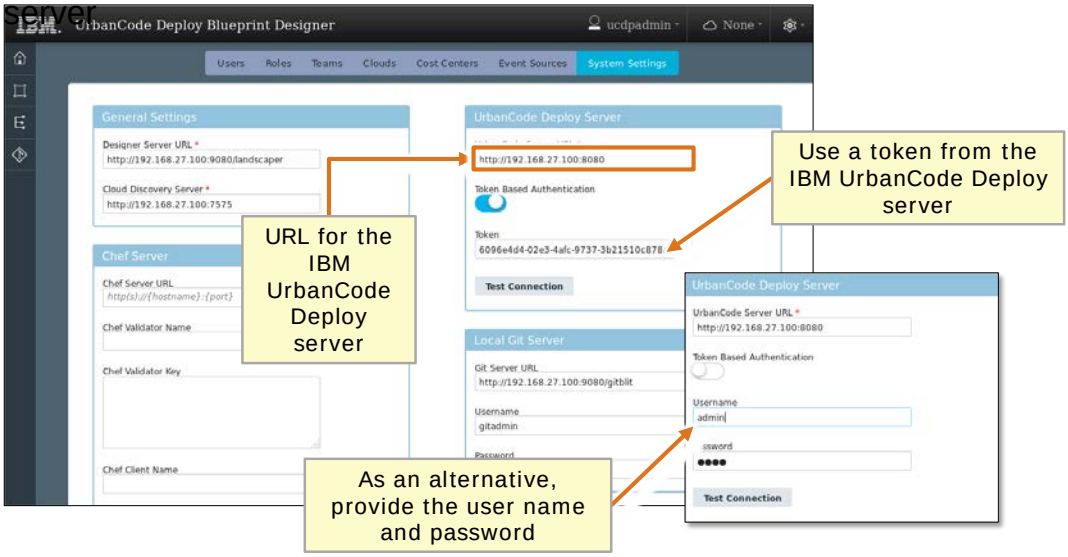
Server connections

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.

IBM Training

Server connections (2 of 2)

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



Settings and Configuration

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.

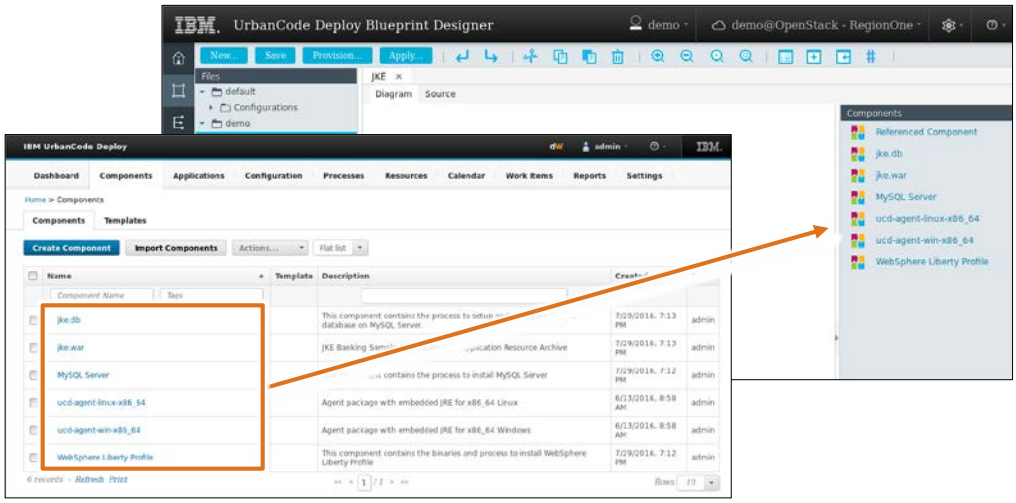
Tokens provide authorization for agents, agent relays, users, and external systems or applications from the server. Agents use tokens when they run process steps and communicate with the IBM UrbanCode Deploy server and external services. Users can use tokens with the command-line interface (CLI) client 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 Training

IBM UrbanCode Deploy components

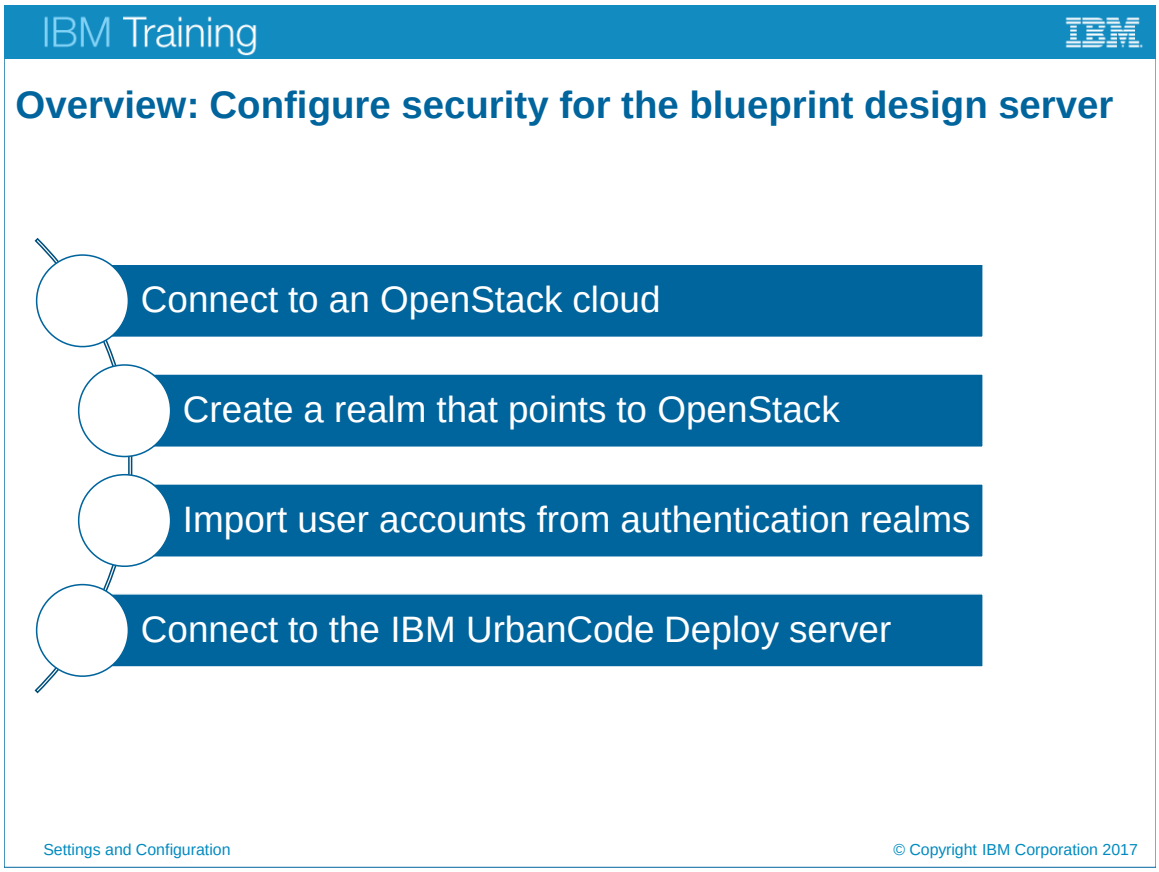
- The components in IBM UrbanCode Deploy are available in the Blueprint Designer



Settings and Configuration

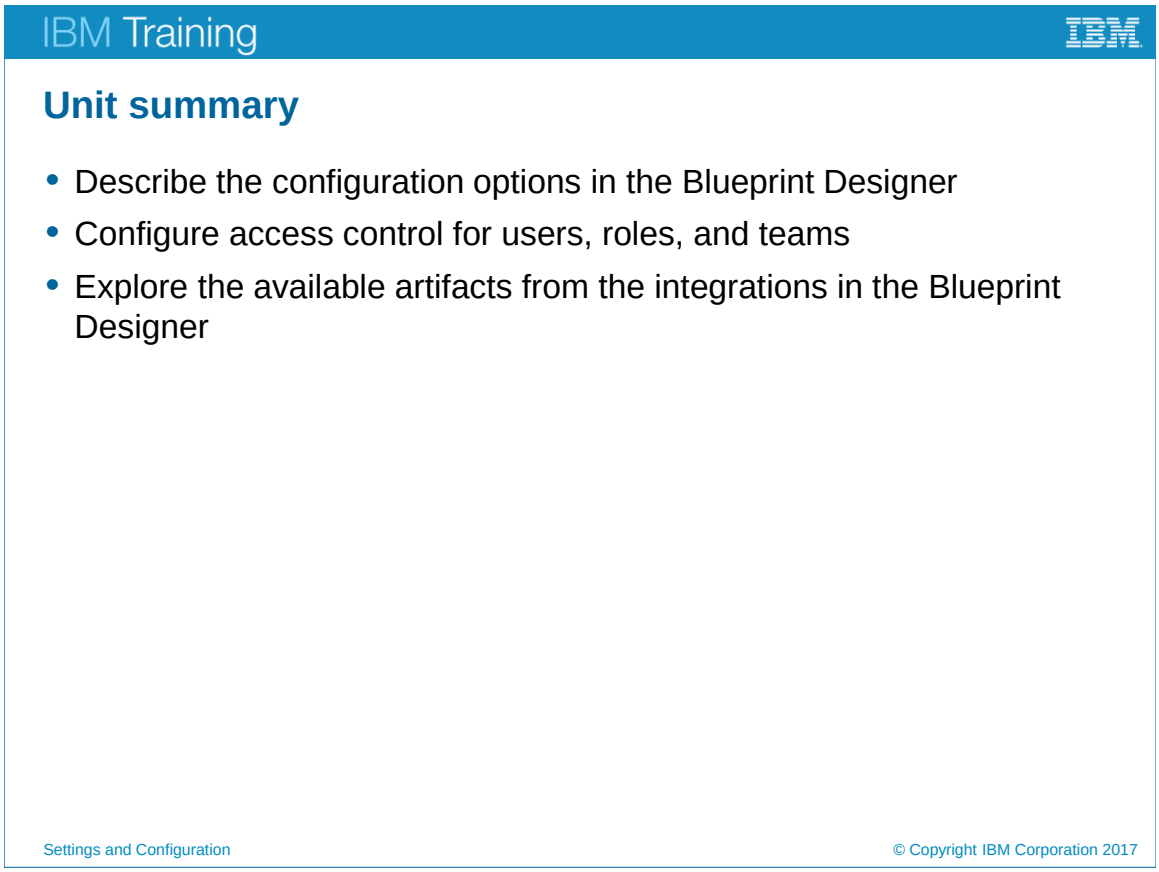
IBM UrbanCode Deploy components

You can now model the application layer of a blueprint.



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

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.

IBM Training

About the exercises

Login instructions to start with the lab:

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

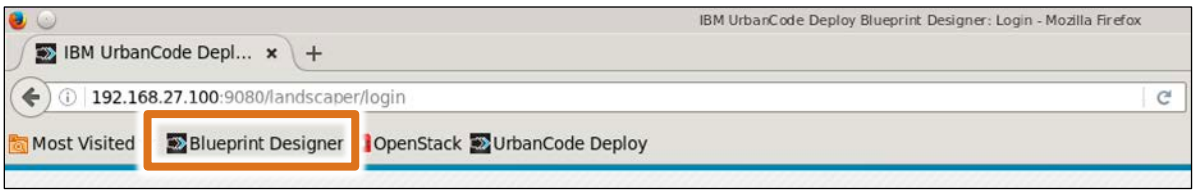
About the exercises

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.

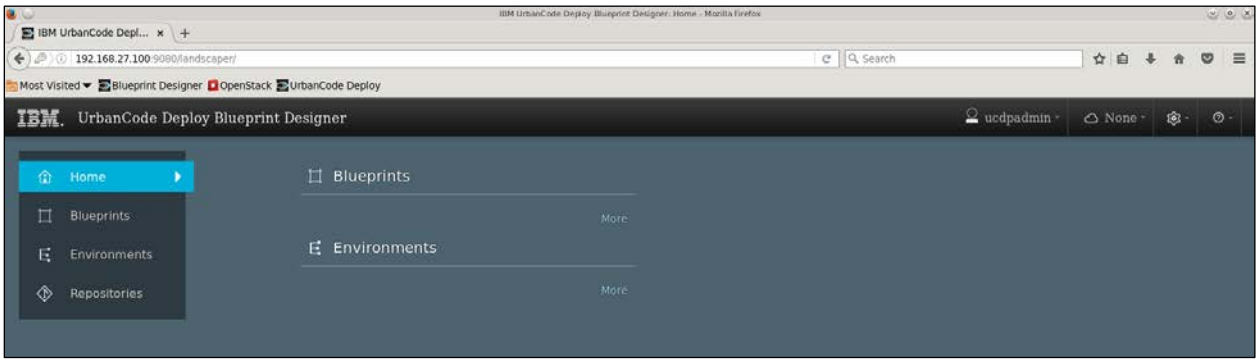


   Alternatively, you may type the URL http://192.168.27.100:9080/landscaper/login, in the address field.

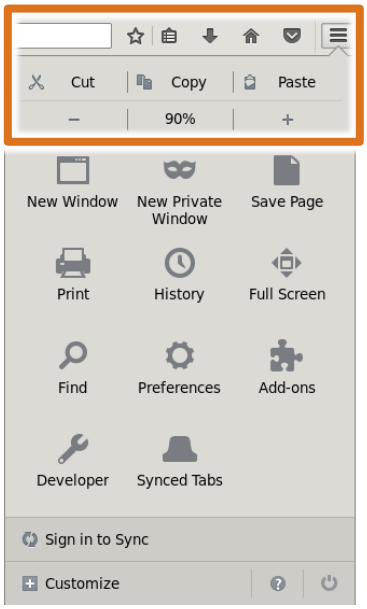
   The Blueprint Designer page is displayed.

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%.



IBM Training

Exercises: Configuring the blueprint design server integrations

In the following exercises, you will:

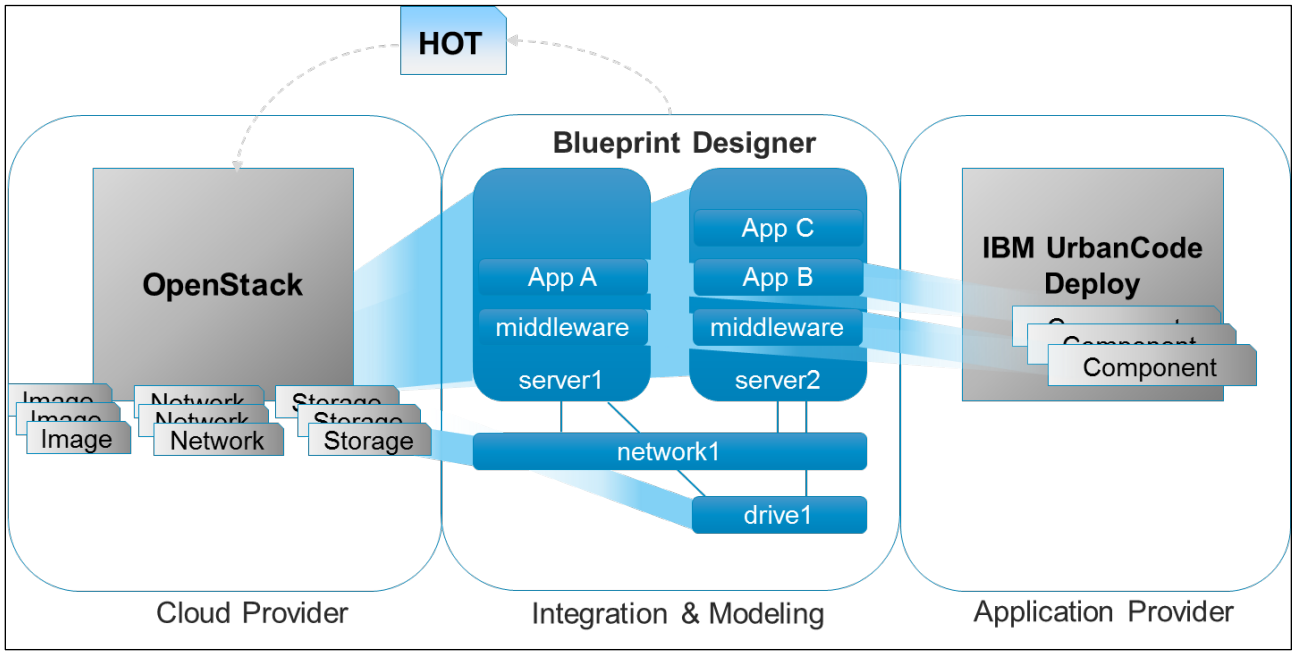
- Connect the blueprint design server to an OpenStack cloud and the IBM UrbanCode Deploy server
- Import users from the OpenStack Keystone service and grant privileges to teams and users in the Blueprint Designer
- Explore the OpenStack dashboard and view the cloud resources

Settings and Configuration

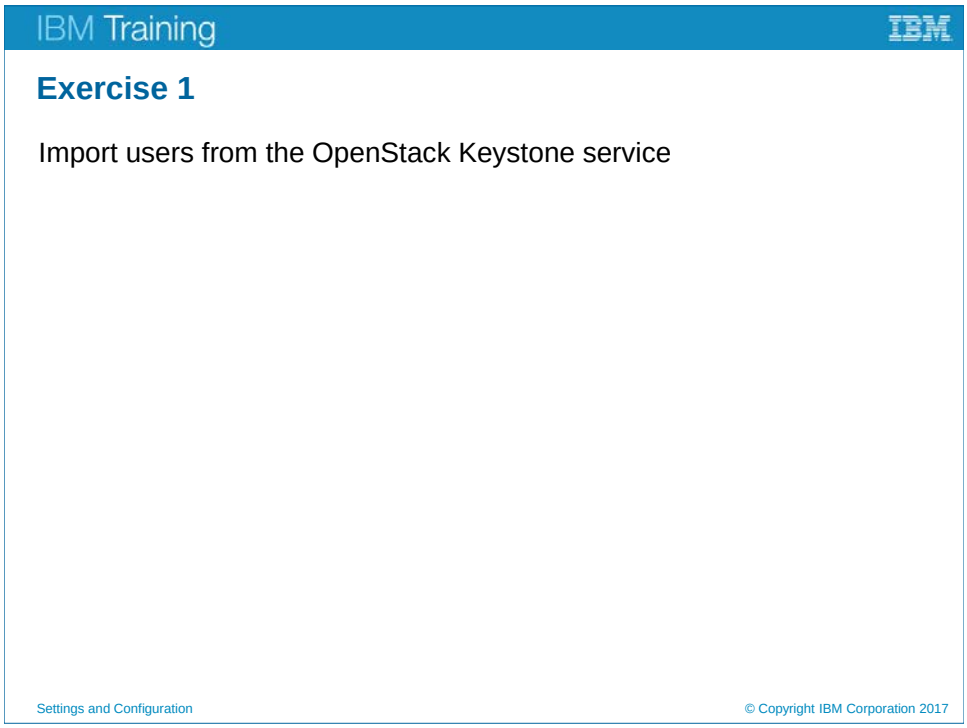
Exercises: Configuring the blueprint design server integrations

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.



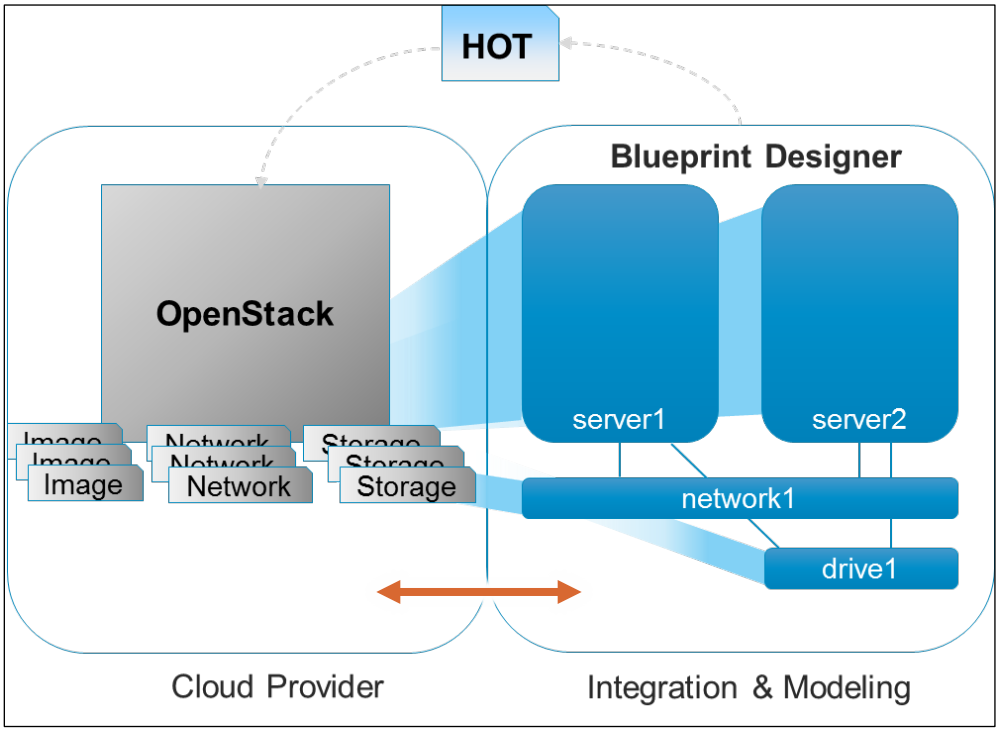
Although this lab focuses on using OpenStack as the cloud provider, you can also use other cloud types, such as Azure, Amazon EC2, vCenter, and Softlayer.



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.

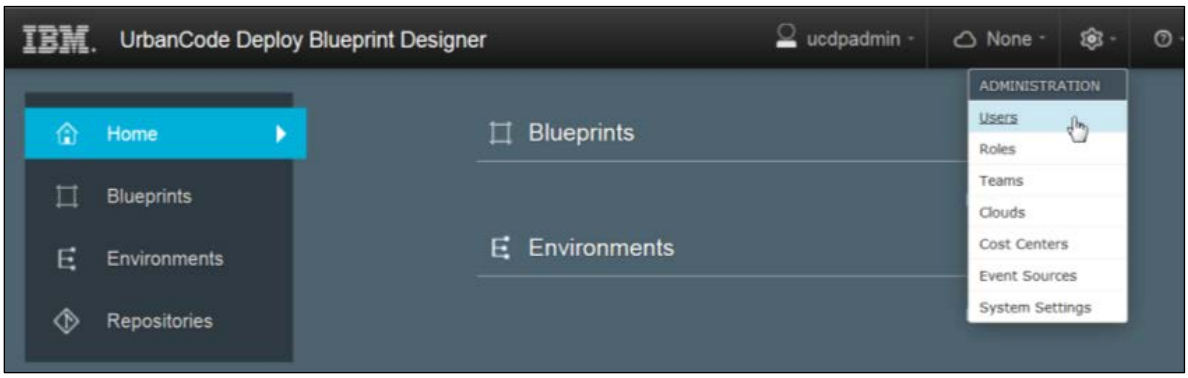
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 1:
Tasks and results

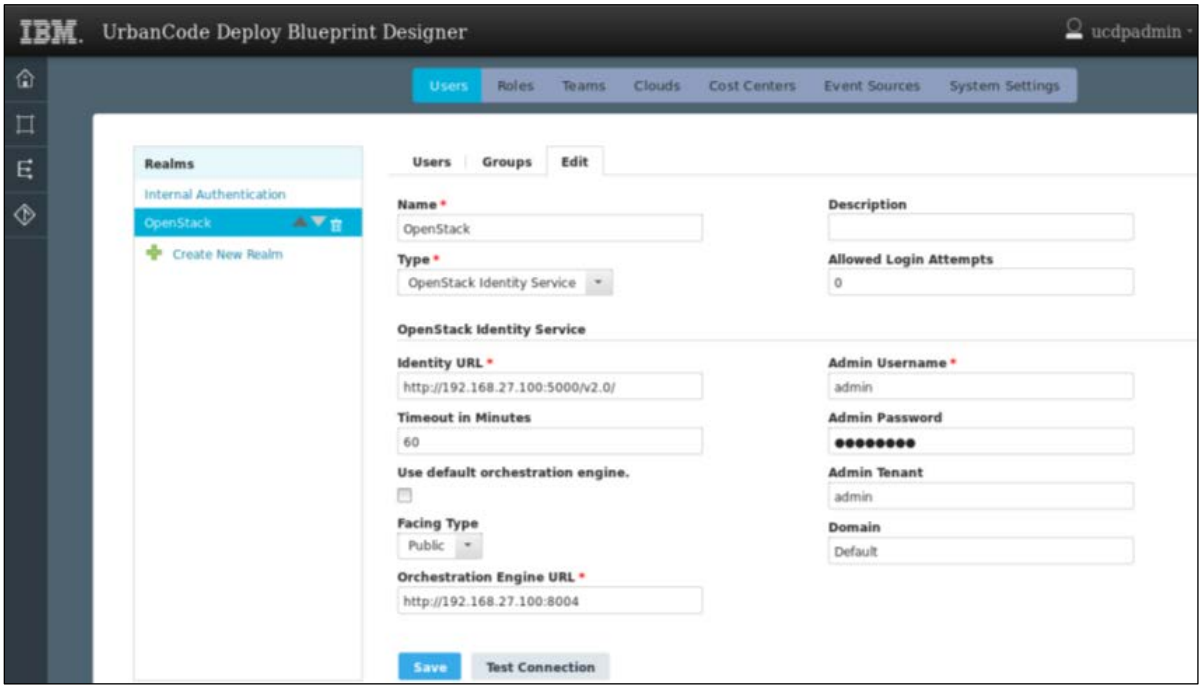
Task 1. View the authentication realm settings.

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.



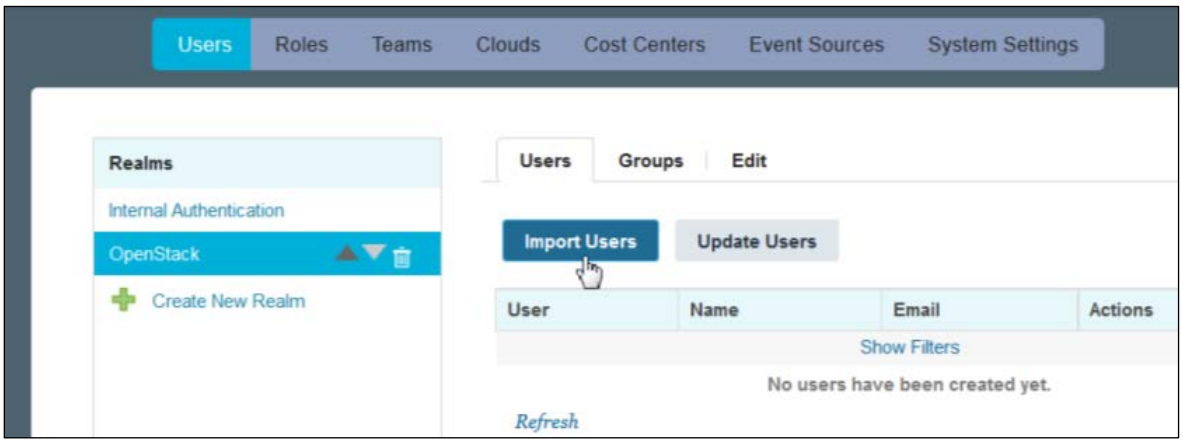
In this exercise, the settings are already configured for this cloud server.

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

Task 2. Import users from the OpenStack Keystone server.

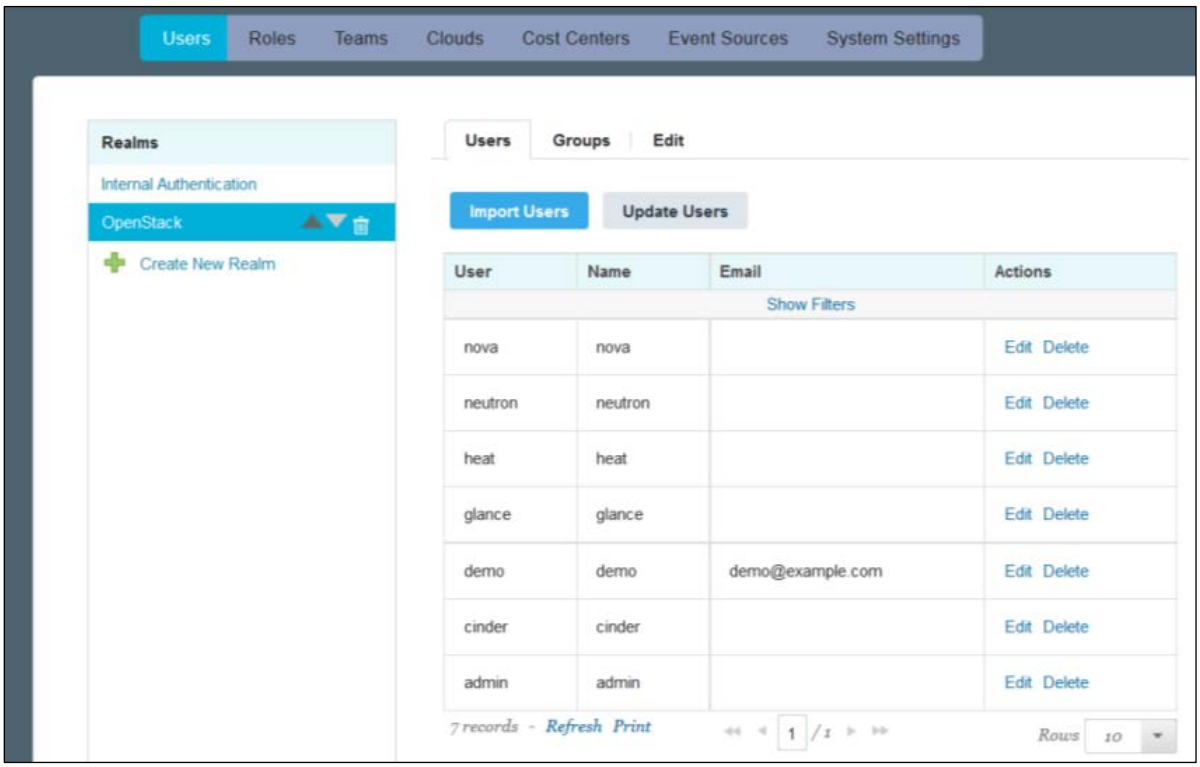
When you import the users from an authentication realm, the blueprint design server creates a cloud connection that is based on that authentication realm and cloud projects that are based on the tenants or projects on the cloud.

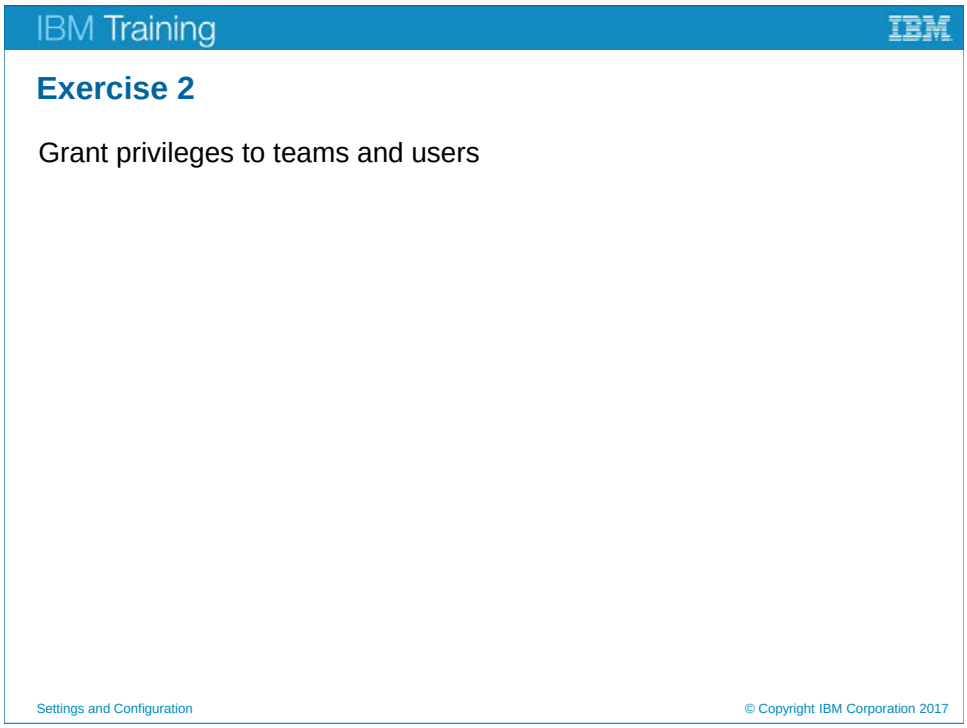
1. For the OpenStack realm, click the **Users** tab, and then click **Import Users**.



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.





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.

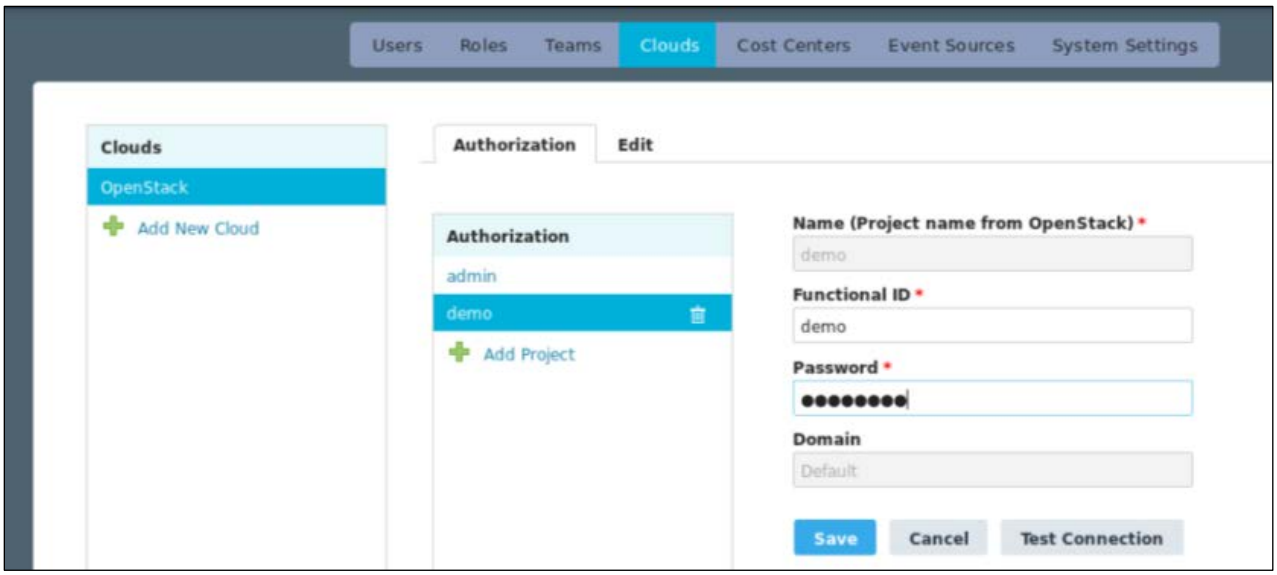
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

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`.



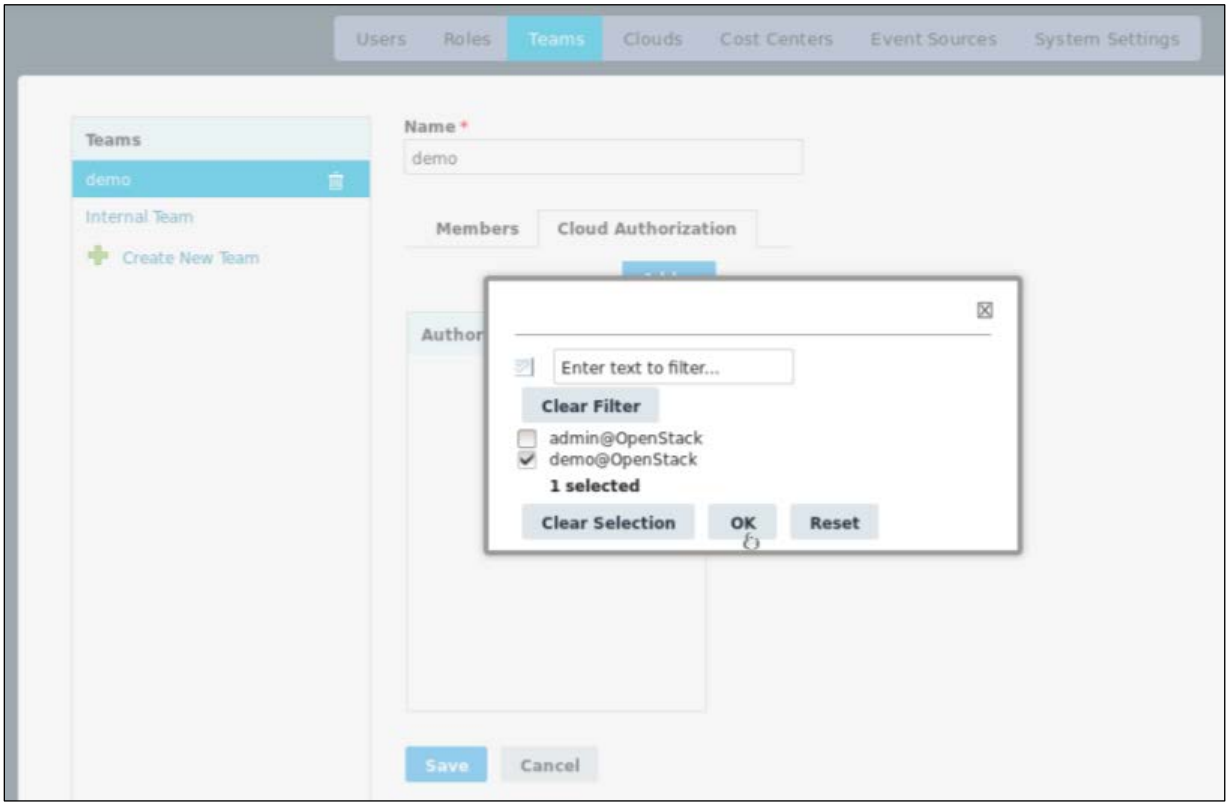
5. Click **Test Connection** to verify the credentials, and then close the Connection Successful window.
6. Click **Save**.

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.

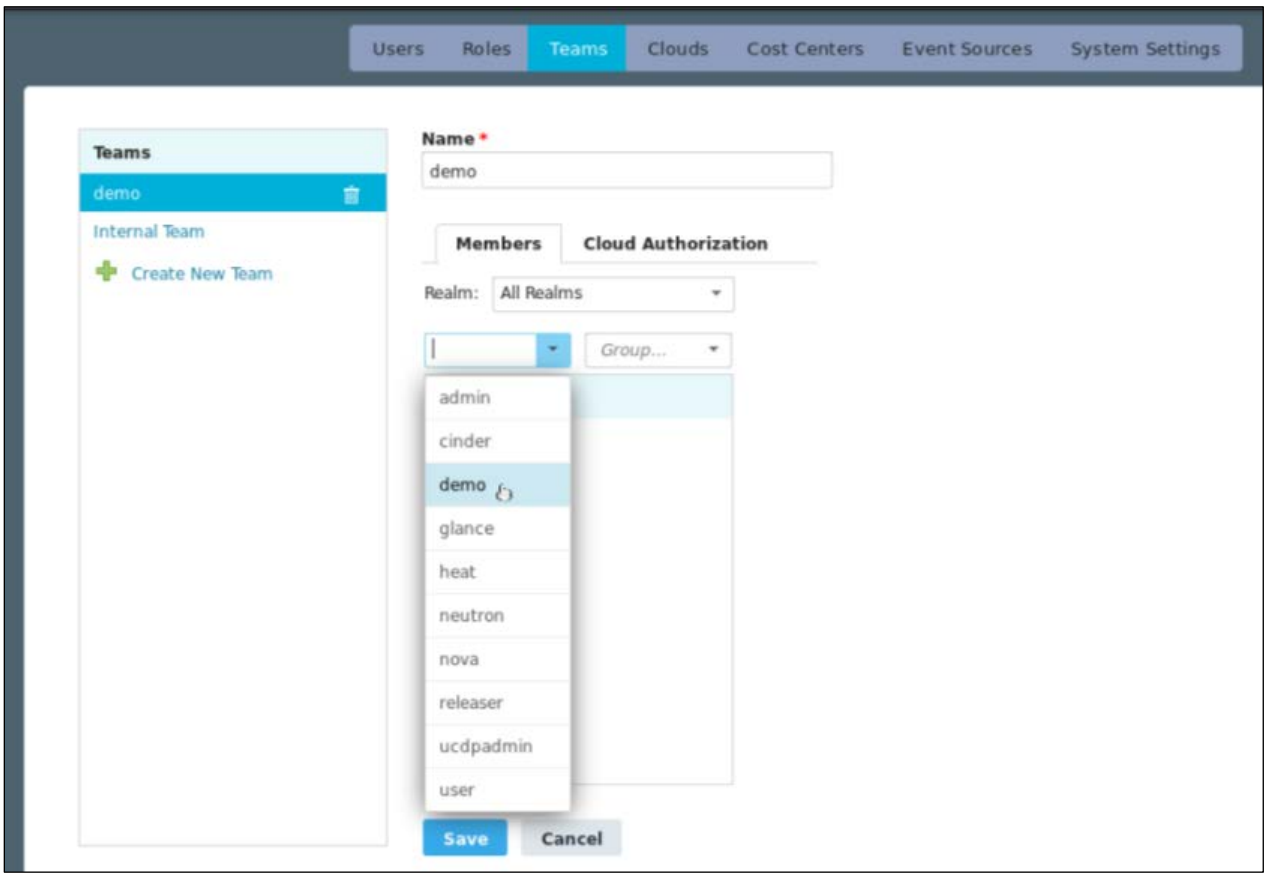
3. Click **Add,** and select **demo@OpenStack**.



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

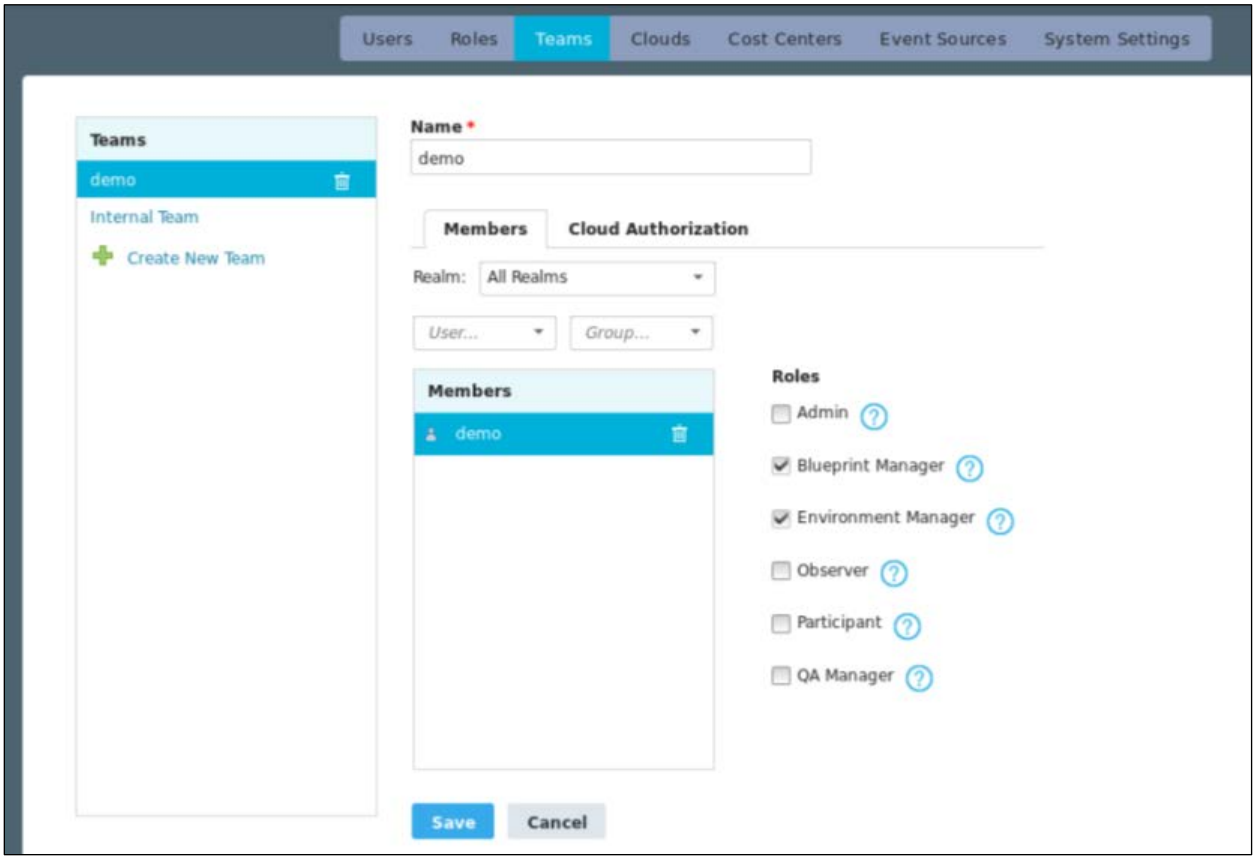
Task 3. Grant Blueprint Designer access to the demo user.

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.



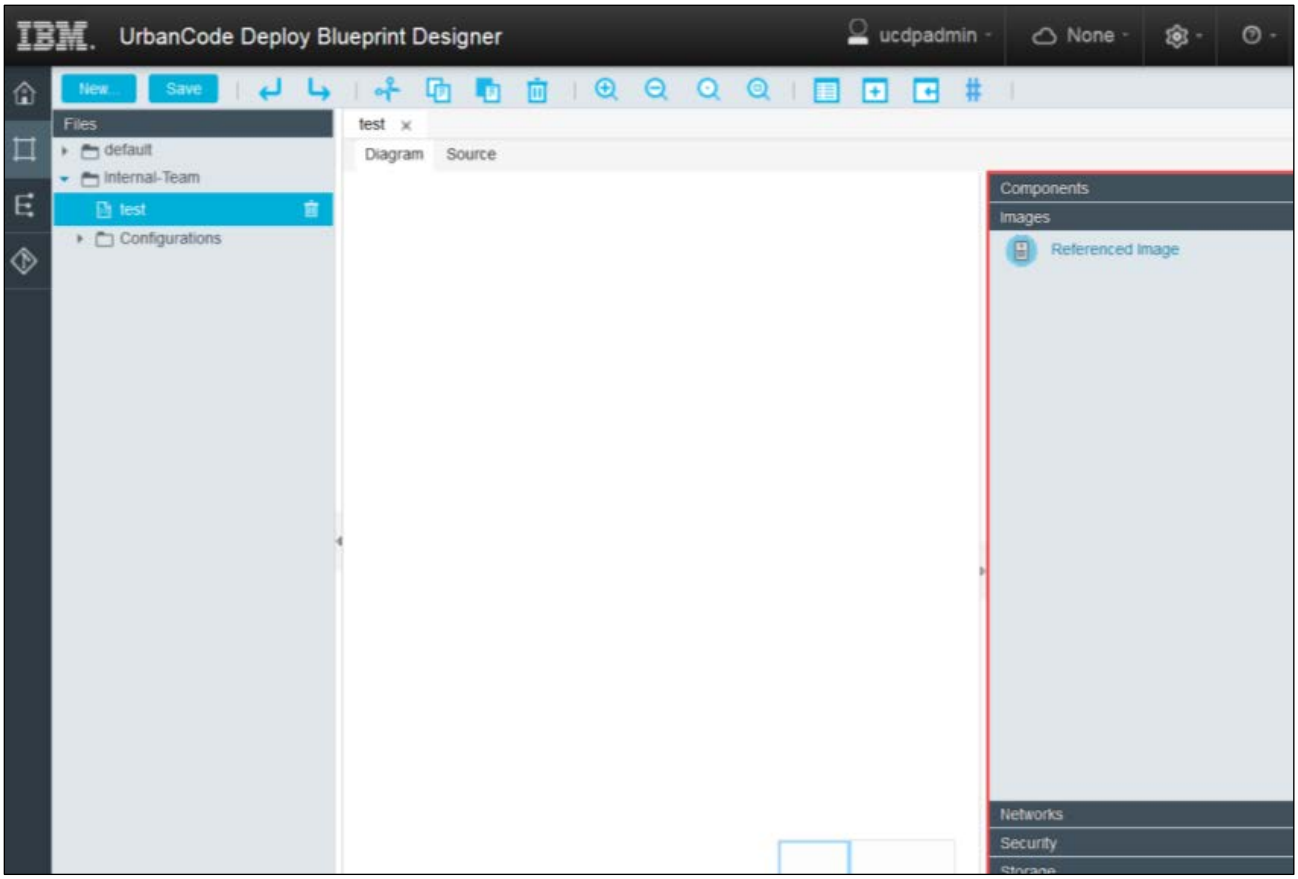
3. In the Roles section, select **Blueprint Manager** and **Environment Manager** for the demo user.

4. Click **Save**.



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.

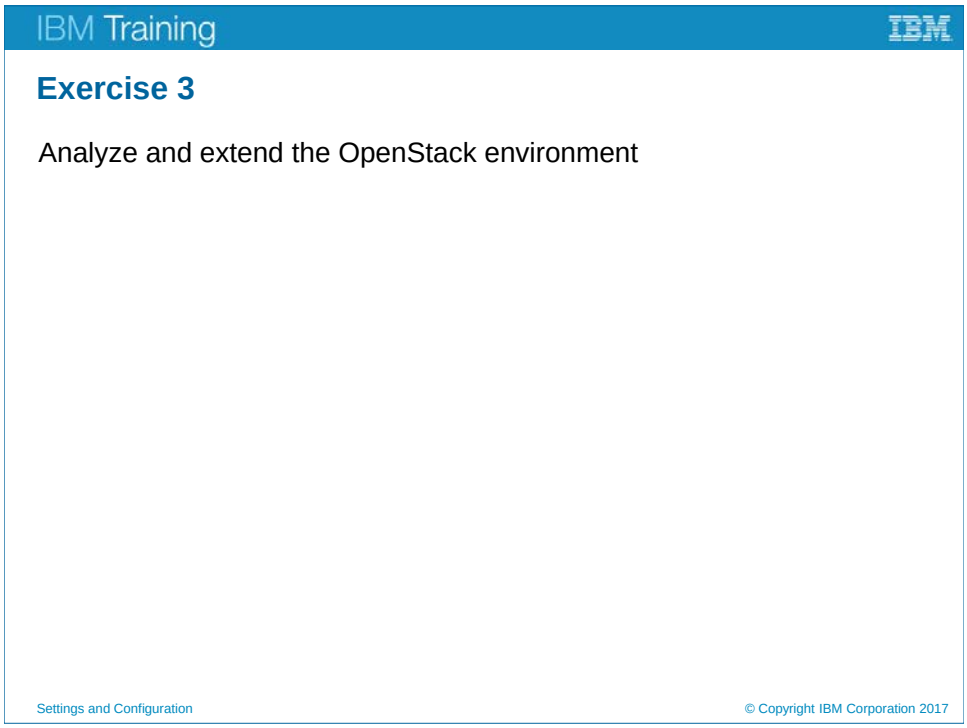
   Note: As the **ucdpadmin** user, you do not have authorization to access the OpenStack cloud. In the second lab, you create a blueprint as the demo user. Currently, while logged in as the ucdpadmin, you would not see any artifacts from the OpenStack cloud if you were to look at the resource drawers to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

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

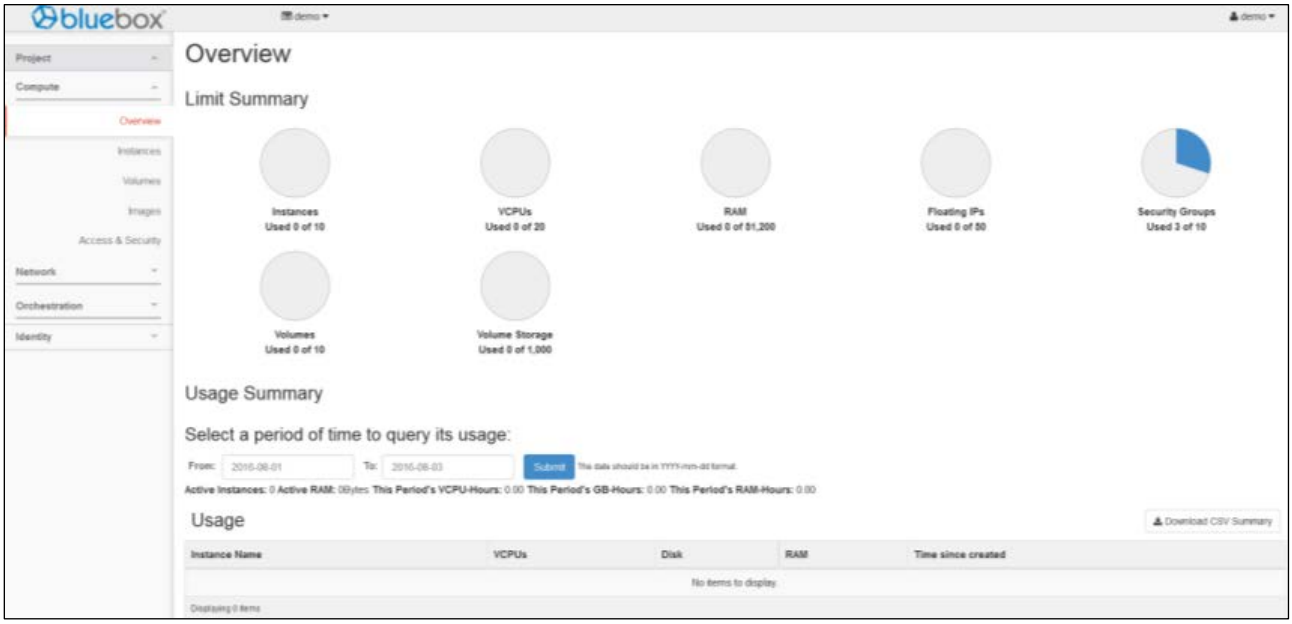
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 3: Tasks and results

Task 1. Analyze and extend the OpenStack environment.

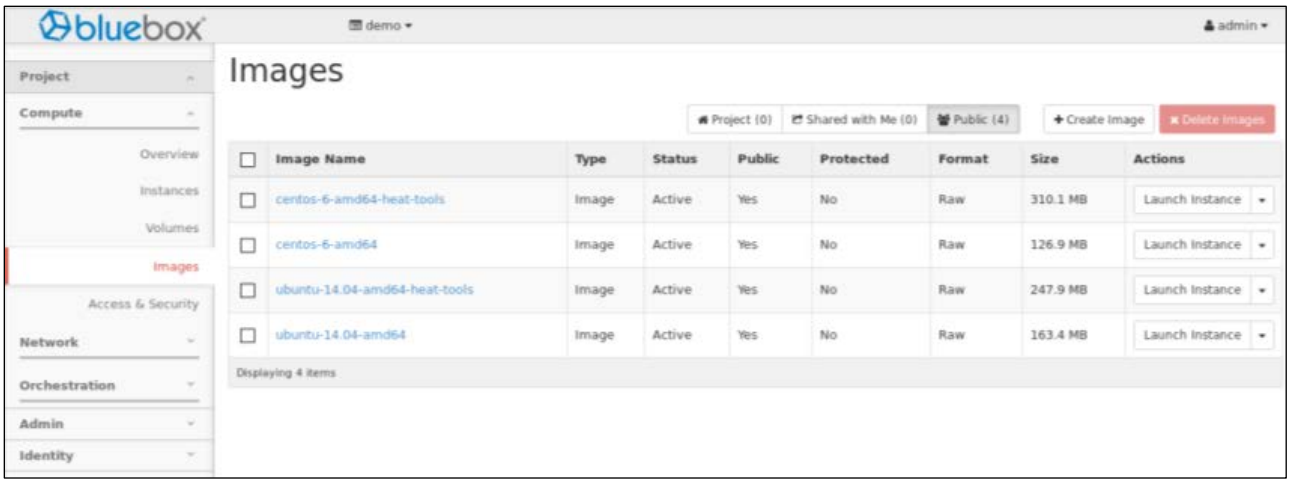
1. From the browser, open a new tab and click the **OpenStack** bookmark, or type the URL, http://192.168.27.100/dashboard/auth/login, in the address field.
2. To log in to the default domain, type `demo` in the **User Name** field and `labstack` in the **Password** field.

   The dashboard opens to the overview page for the project.

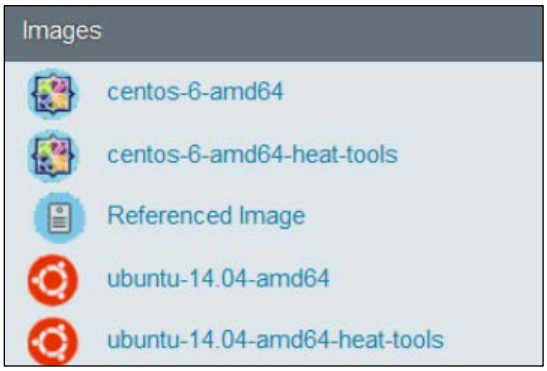


3. In the panel on the left, click **Images**.

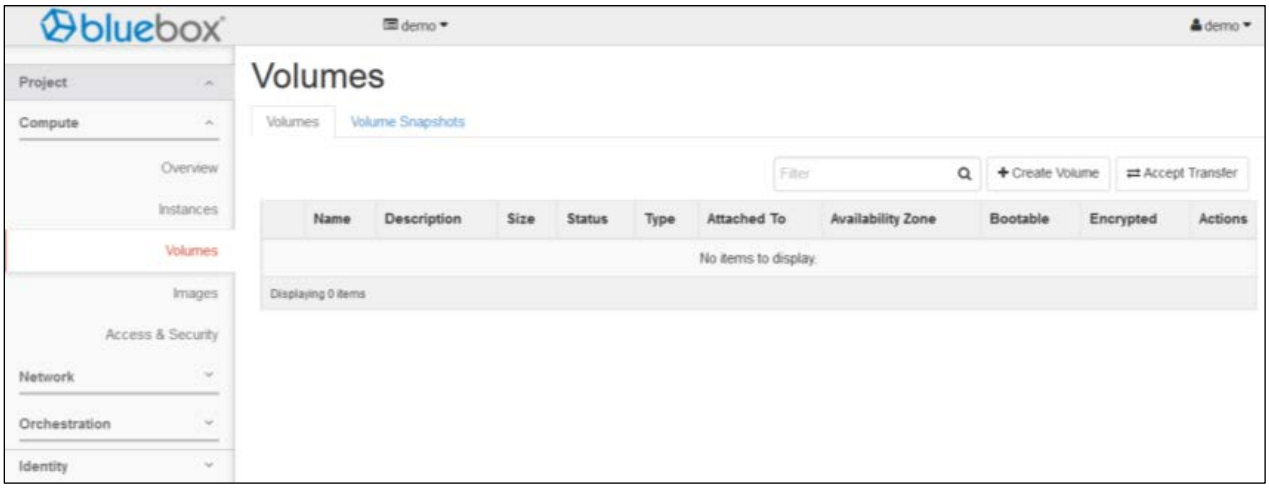
   The images section stores and retrieves virtual machine disk images. OpenStack makes use of this during instance provisioning. The four images in the following screen capture are the same as the ones that are made available to the images drawer of the Blueprint Designer:



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



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



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.

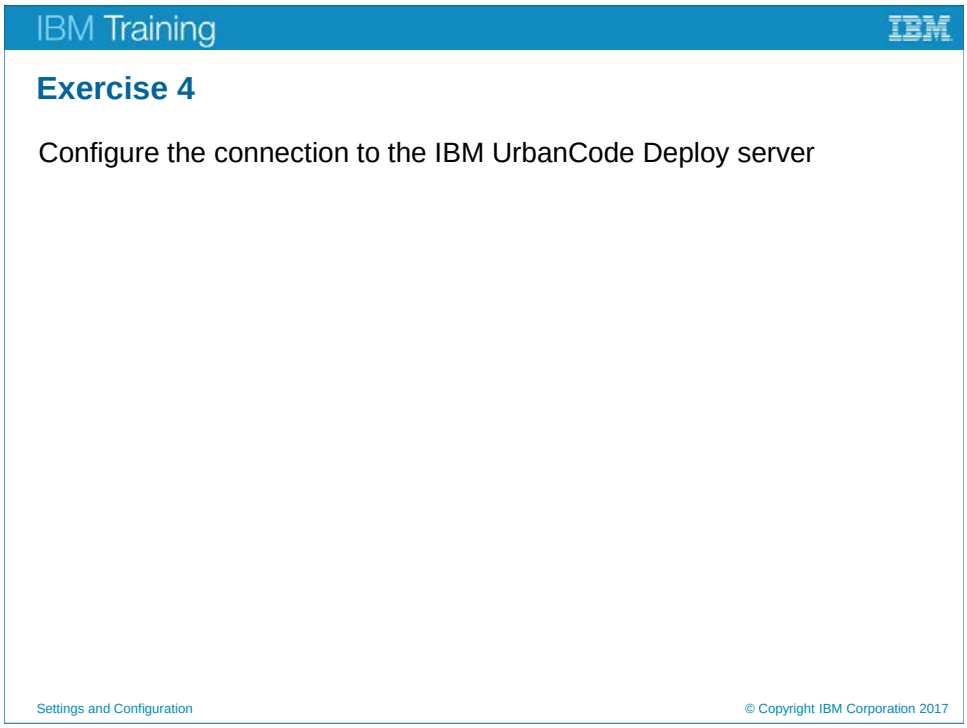
Note: In the Blueprint Designer, the Storage drawer shows the cloud storage volumes.

5. 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:

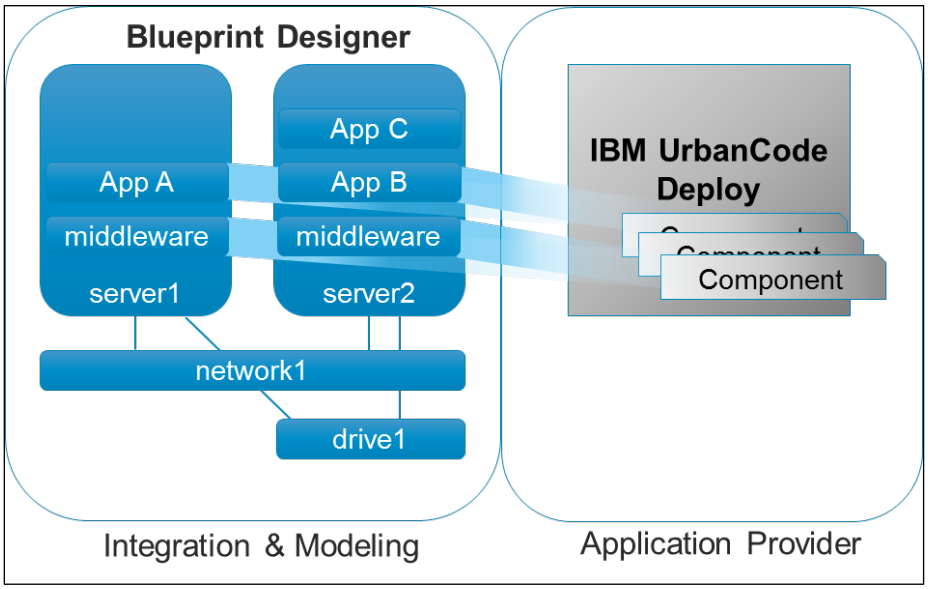


Exercise 4: Configure the connection to the IBM UrbanCode Deploy server

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:

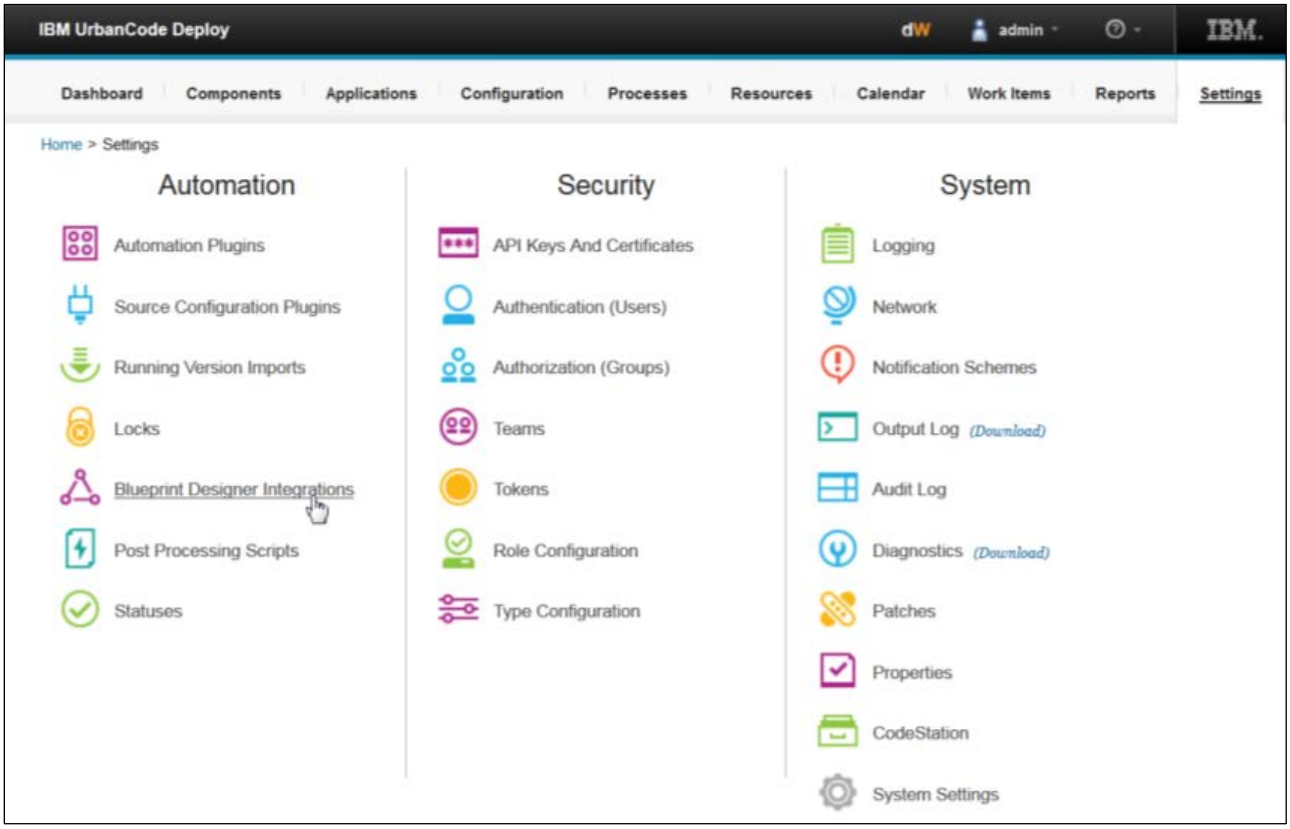
- View the integration for the blueprint design server in IBM UrbanCode Deploy.
- View the integration for the IBM UrbanCode Deploy server in the Blueprint Designer.

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**.



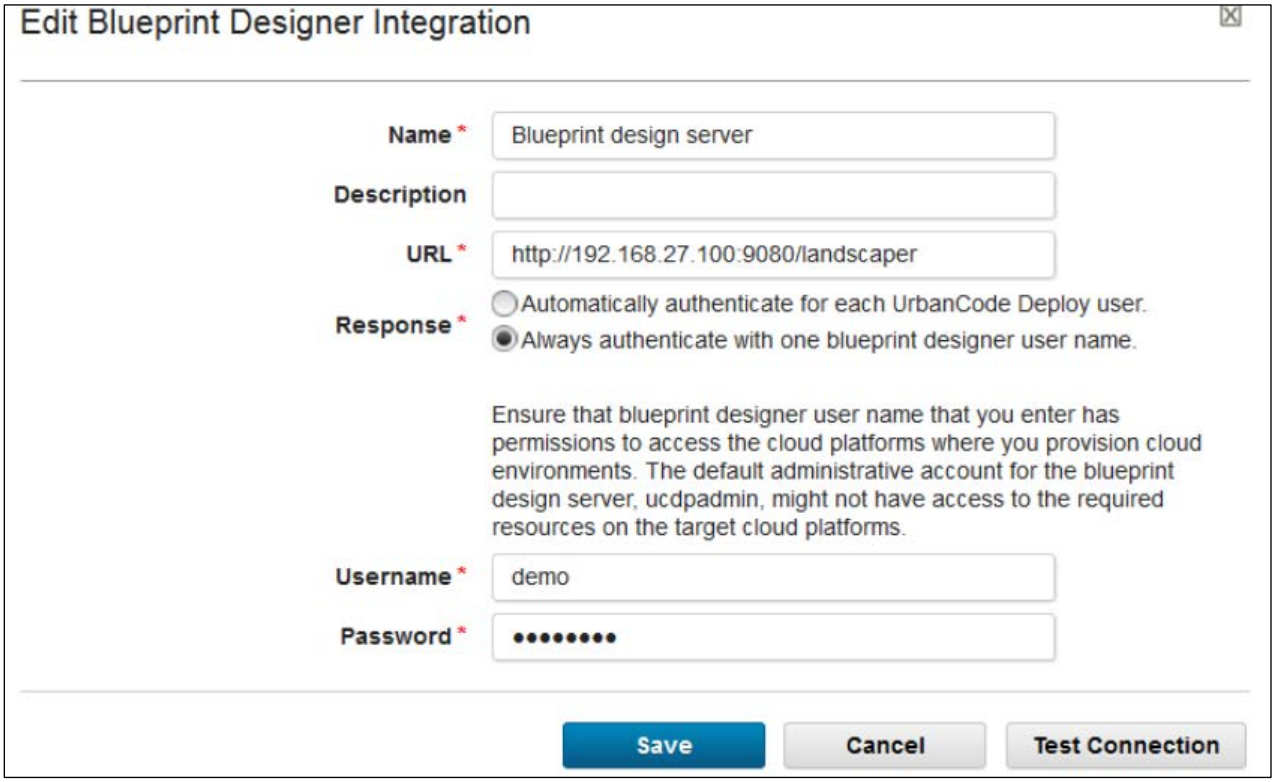
Under the Name column on the Automation page, you see an existing integration, called *landscaper*. This provides the integration information for the Blueprint Designer.

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**.

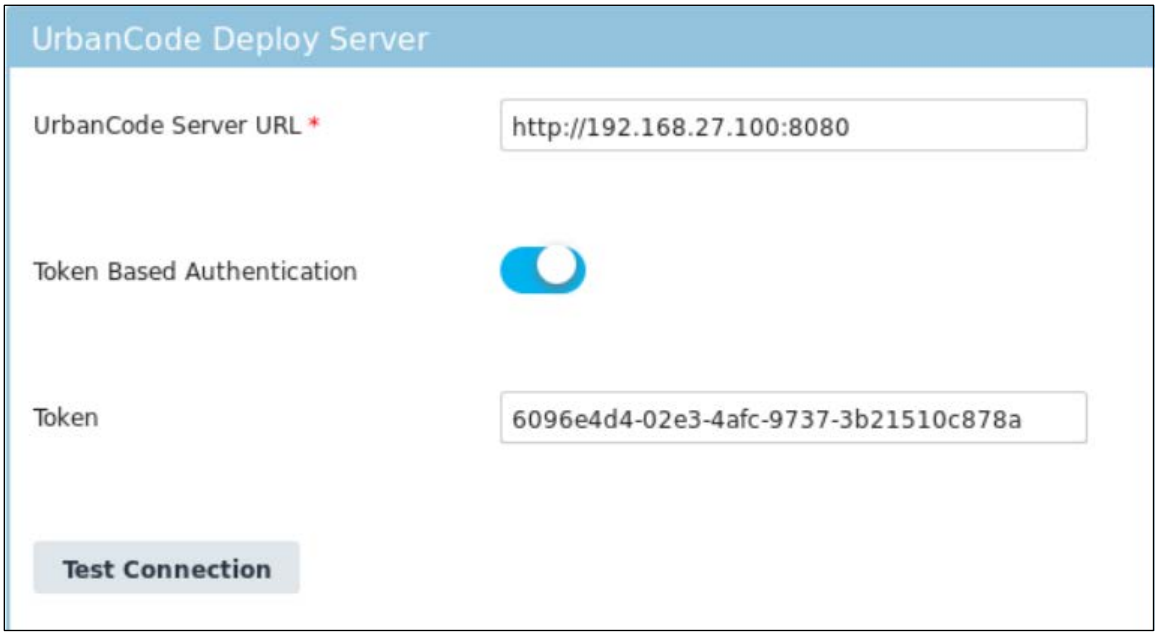


Task 2. View the integration for the IBM UrbanCode Deploy server in the Blueprint Designer.

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`.

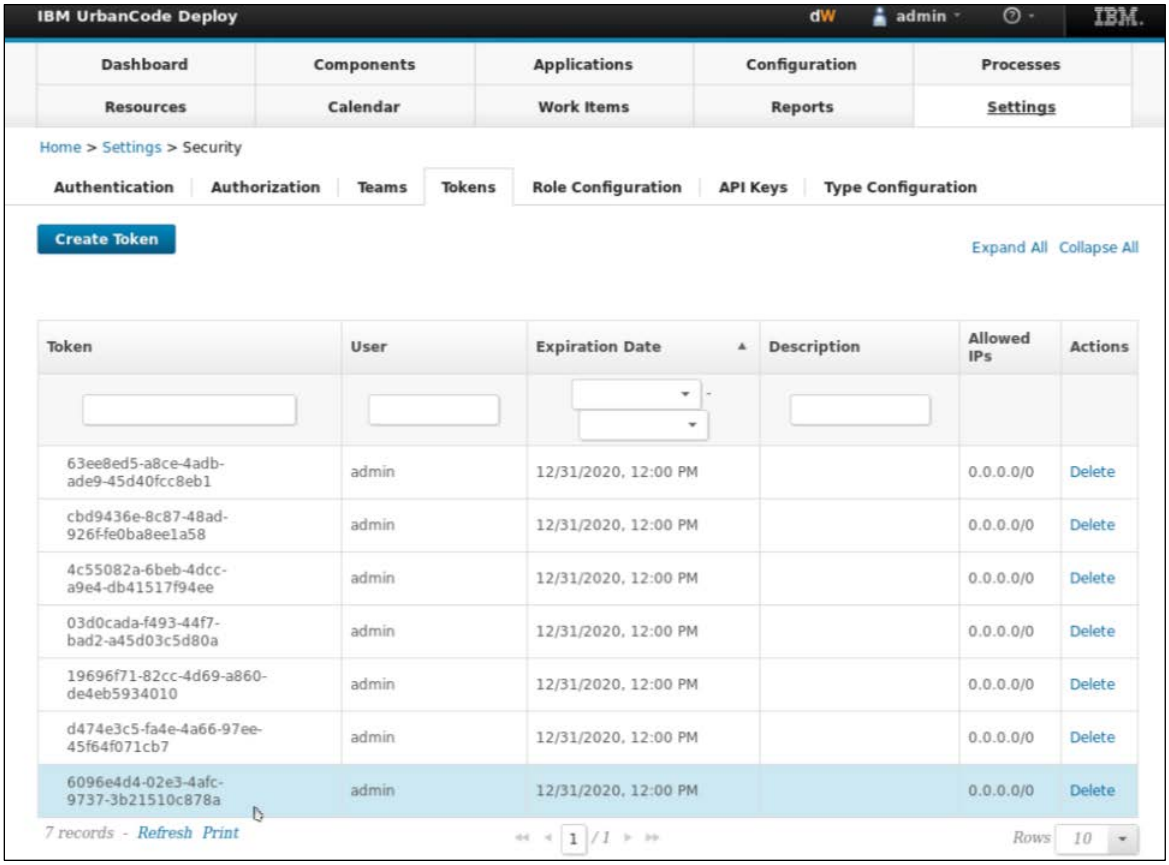
2. At the upper-right corner of the page, click the **Settings** icon  , and click **System Settings** from the drop-down list.

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



   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.



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

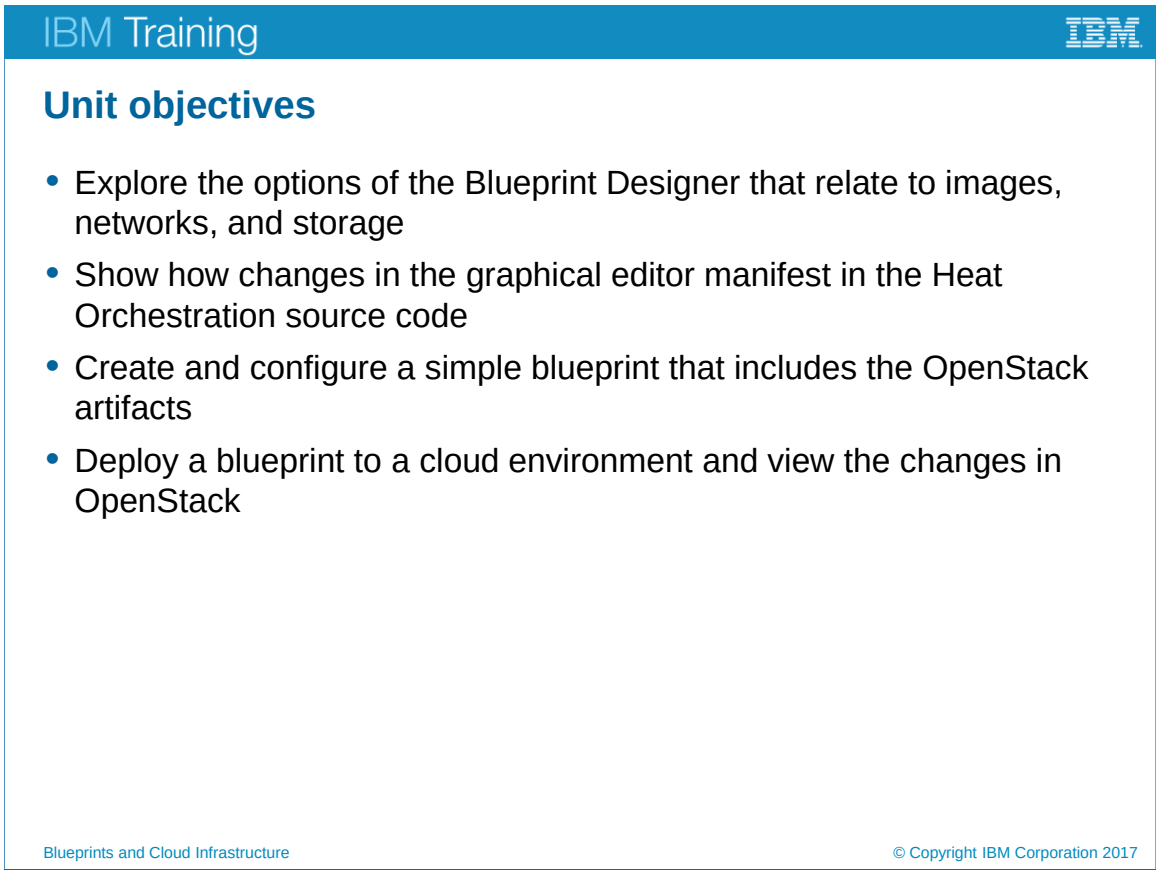
In this unit, you viewed the integrations between the blueprint design server with both the OpenStack cloud and the IBM UrbanCode Deploy server. The integration with OpenStack provides resources that you can use to create a blueprint that describes an environment, such as images, networks, and storage 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.

IBM Training

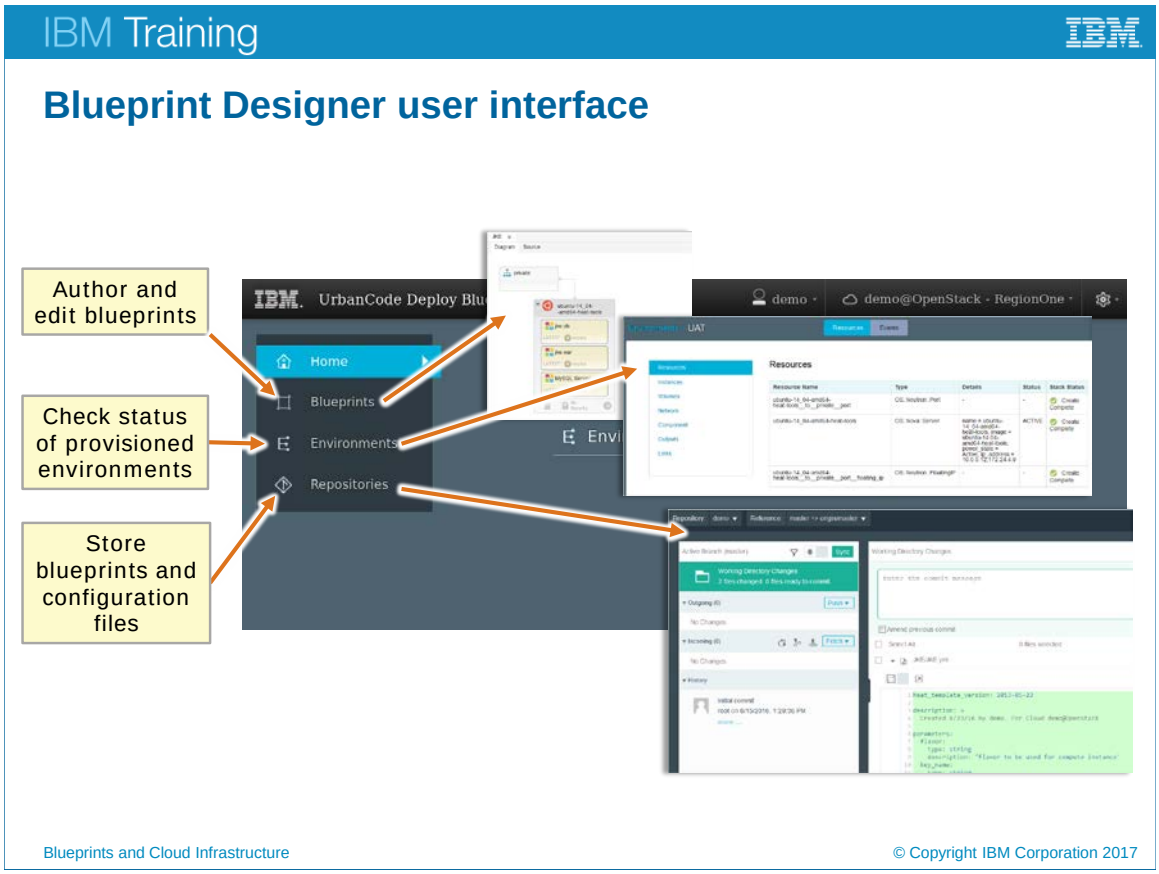
Blueprints and Cloud Infrastructure

IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1



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.



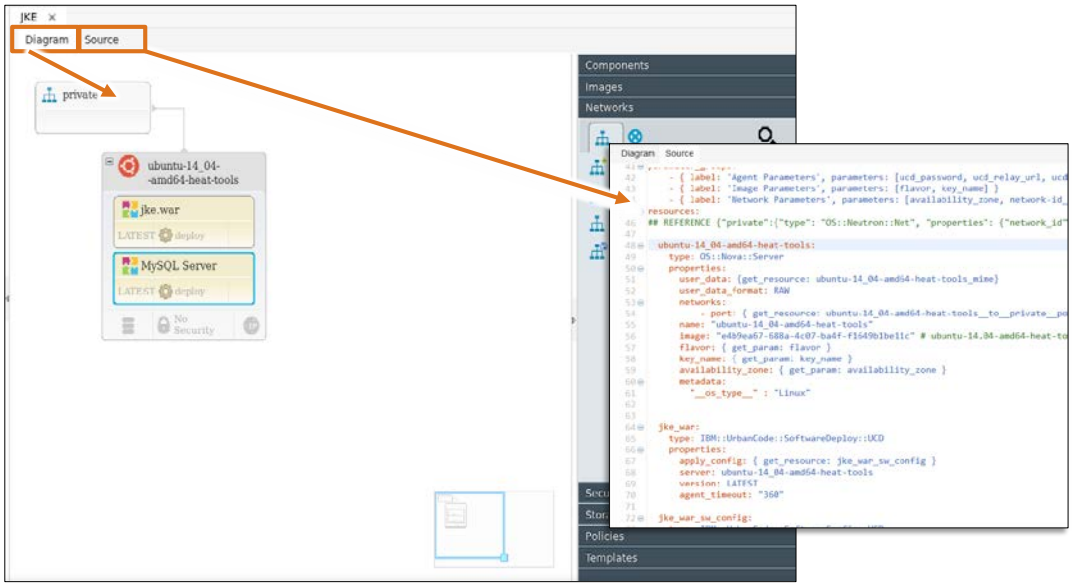
Blueprint Designer user interface

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.

IBM Training

Diagram and Source editor in Blueprint Designer



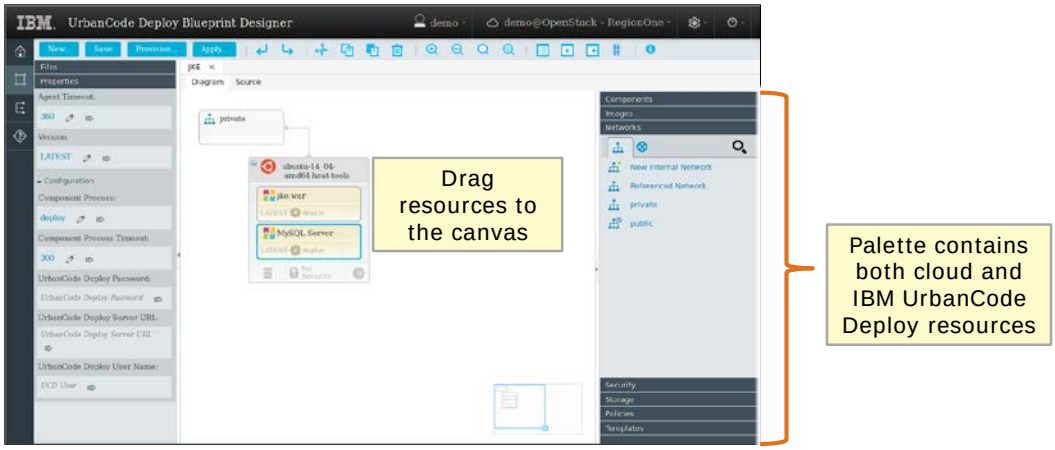
Blueprints and Cloud Infrastructure

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

A graphical depiction of the modeled template



Blueprints and Cloud Infrastructure

A graphical depiction of the modeled template

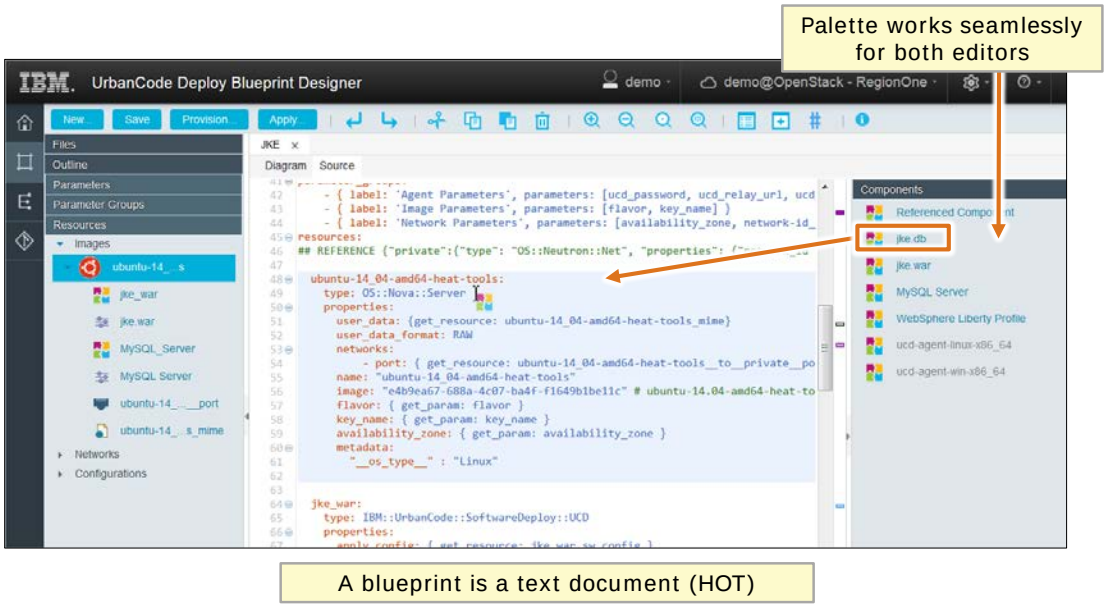
The palette at the right side of the editor shows resources that you can add to the blueprint. The palette is filtered based on the cloud system that you are connected to, and the palette is divided into multiple drawers:

- 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.

IBM Training

Source code editor



Blueprints and Cloud Infrastructure

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.

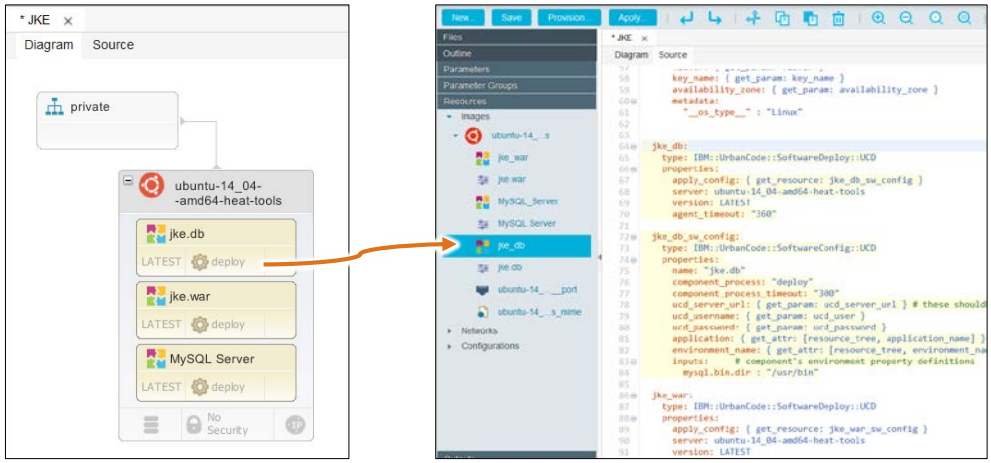
The left side of the editor shows the following sections when the Source editor is open:

- **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.

IBM Training

Preserving object selection and state

- The editors are synchronized and preserve object selection and state



GUI

HOT source code

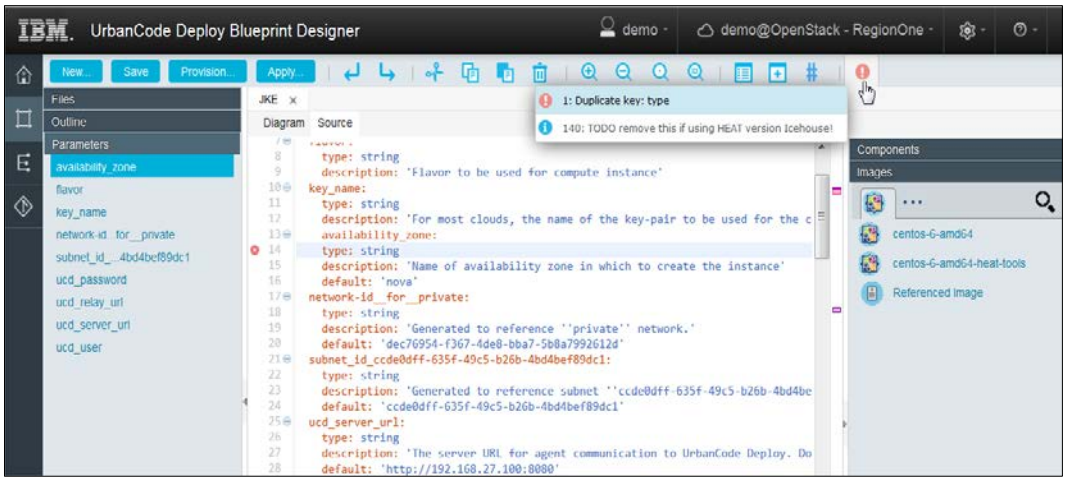
Blueprints and Cloud Infrastructure

Preserving object selection and state

The editors are synchronized and preserve object selection and state.

IBM Training

The Source editor includes code validation



Blueprints and Cloud Infrastructure

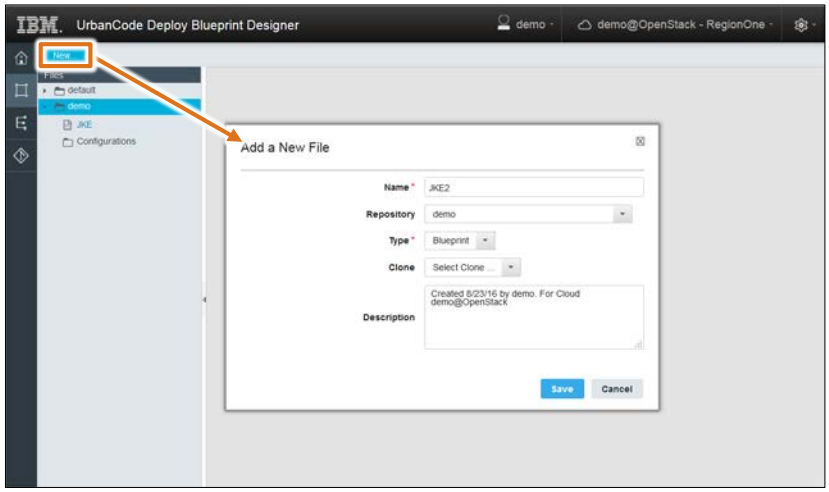
The Source editor includes code validation

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.

IBM Training

Blueprints

- Blueprints store information that is used in provisioning cloud environments



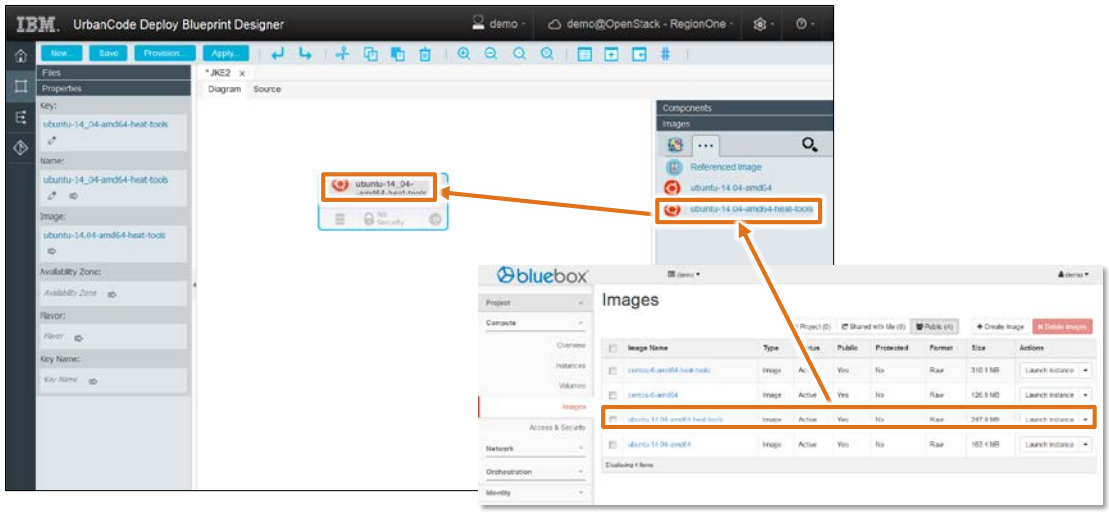
Blueprints and Cloud Infrastructure

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.

IBM Training

The Images drawer



Blueprints and Cloud Infrastructure

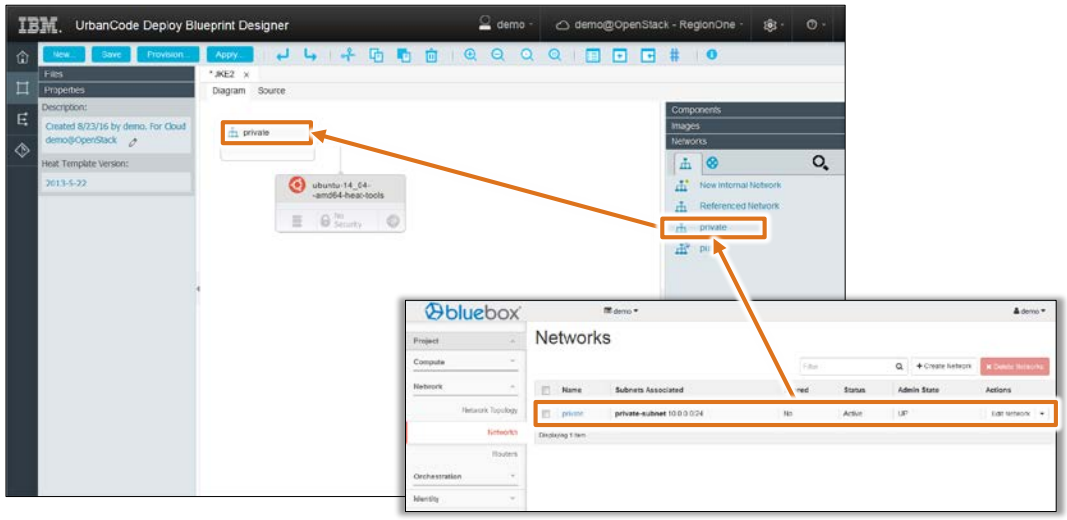
The Images drawer

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.

IBM Training

The Networks drawer



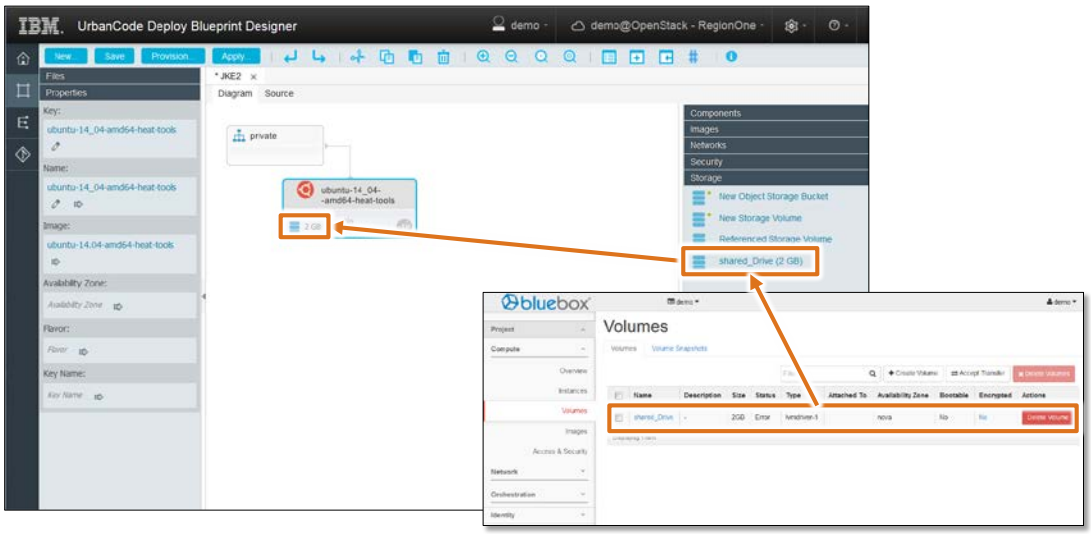
Blueprints and Cloud Infrastructure

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.

IBM Training

The Storage drawer



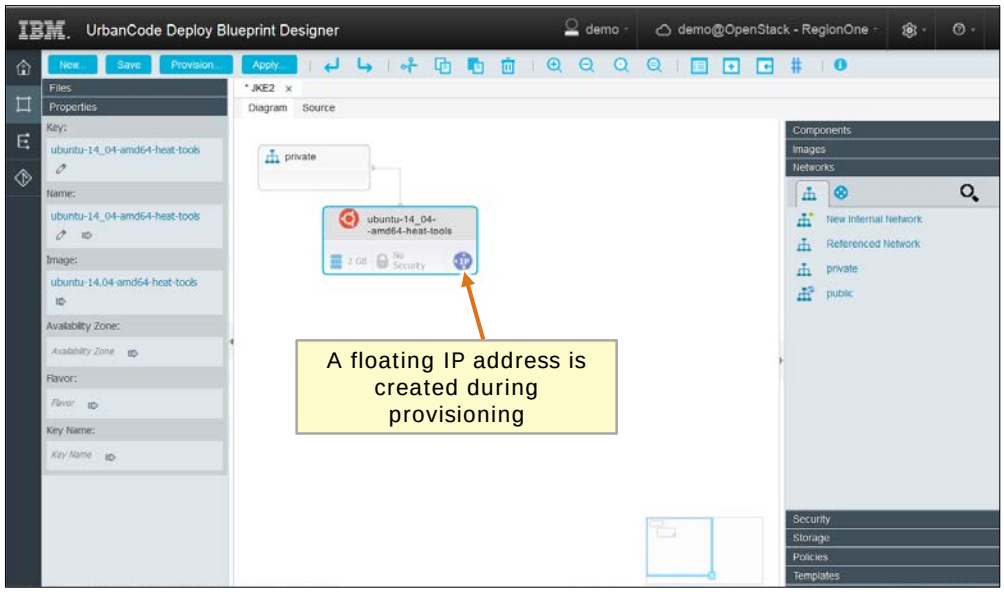
Blueprints and Cloud Infrastructure

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.

IBM Training

Access network and resources



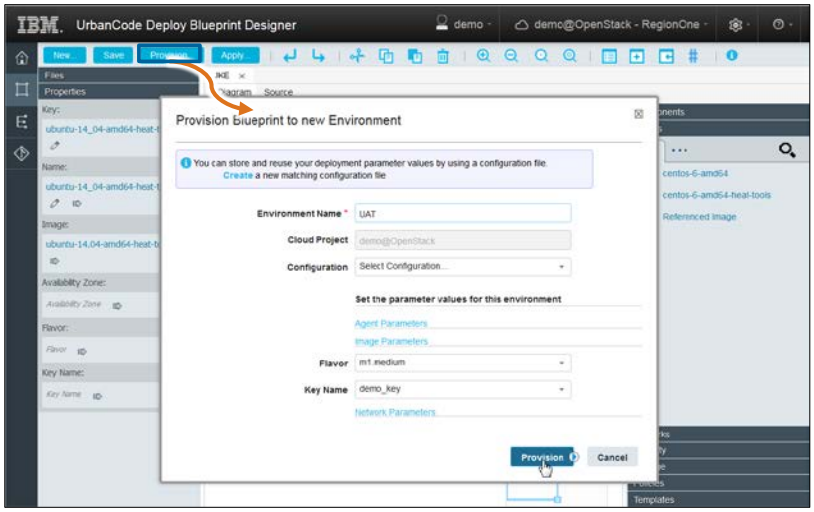
Blueprints and Cloud Infrastructure

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.

IBM Training

Provisioning of environments



Blueprints and Cloud Infrastructure

Provisioning of environments

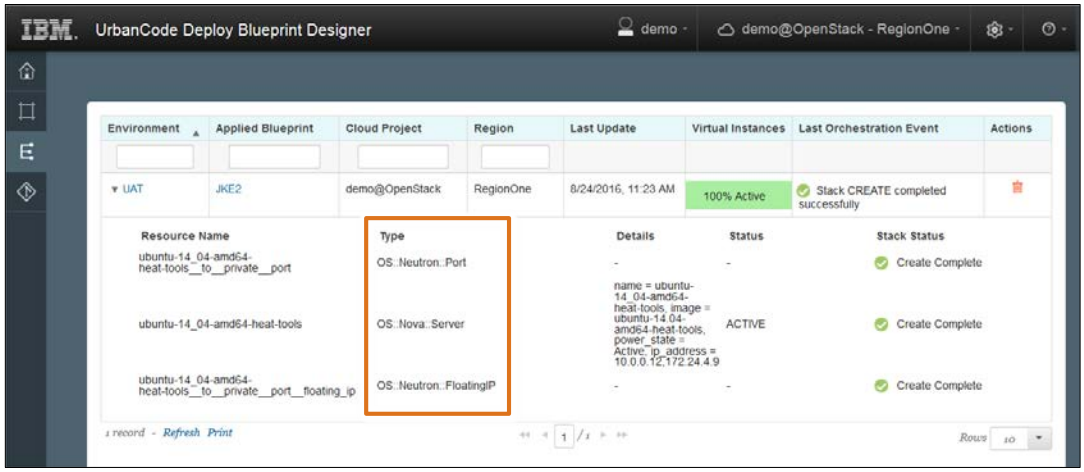
After a blueprint is created, you can provision a cloud environment and set aside resources on the specified cloud that you specified through a blueprint. The provisioning step involves a blueprint, naming the environment to be created, and, optionally, a configuration file. The blueprint is provisioned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

IBM Training

Creation of the virtual environment



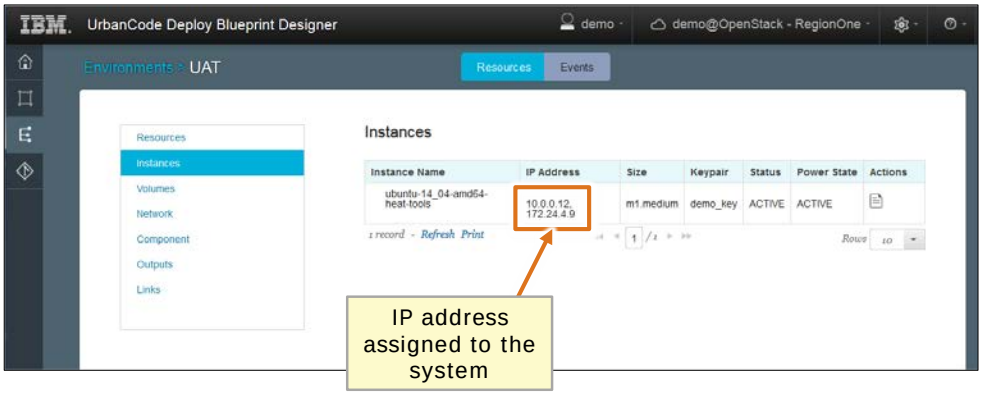
Creation of the virtual environment

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.

IBM Training

Environments

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



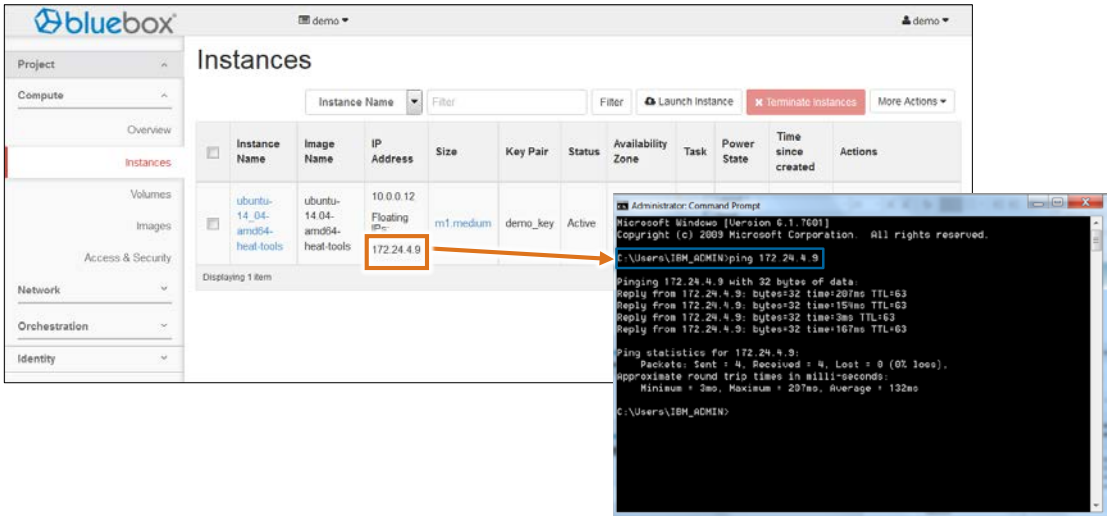
Blueprints and Cloud Infrastructure

Environments

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

IBM Training

OpenStack shows the deployed instance



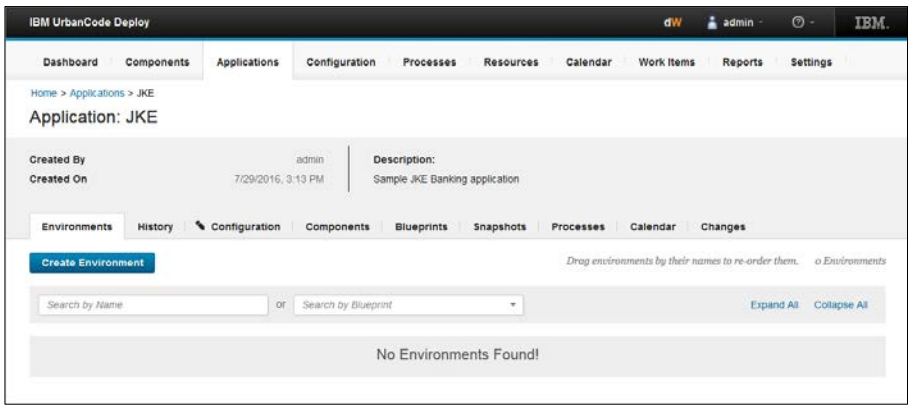
Blueprints and Cloud Infrastructure

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.

IBM Training

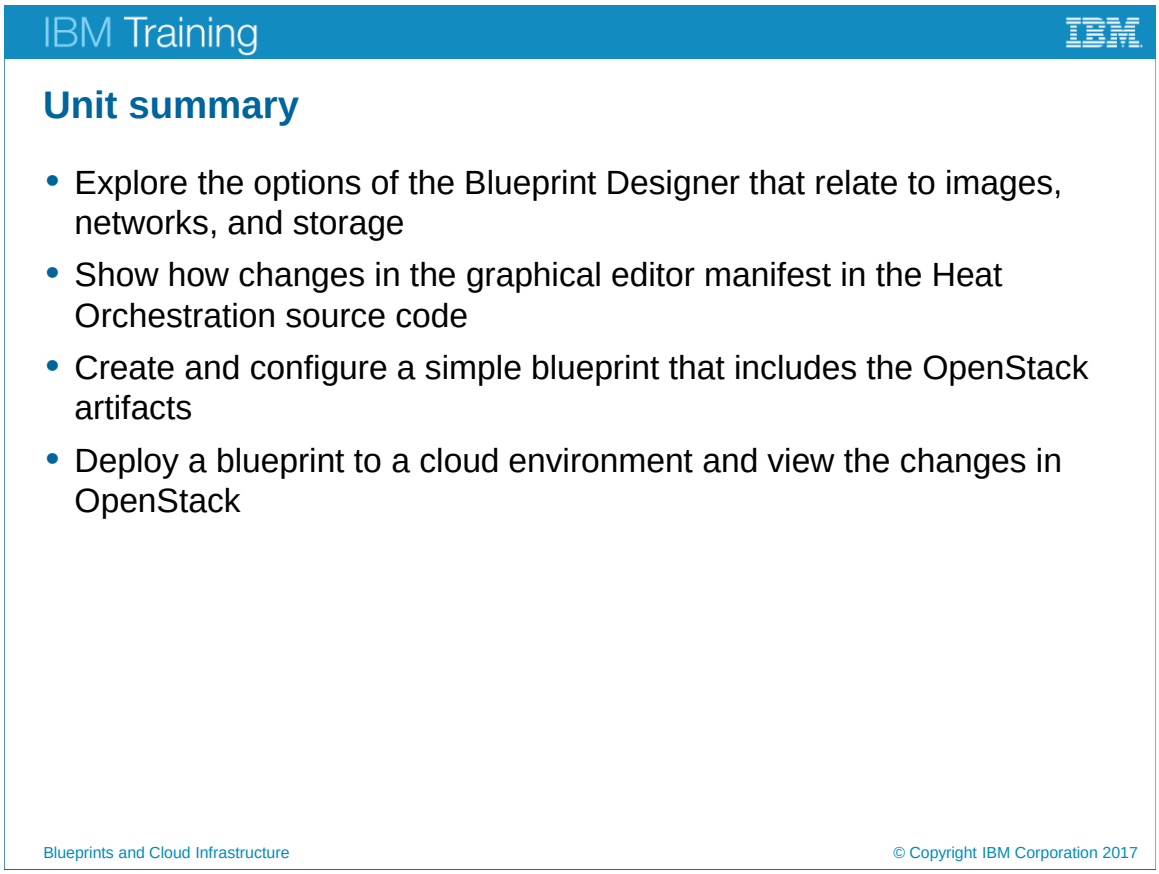
IBM UrbanCode Deploy shows no activity



Blueprints and Cloud Infrastructure

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

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.

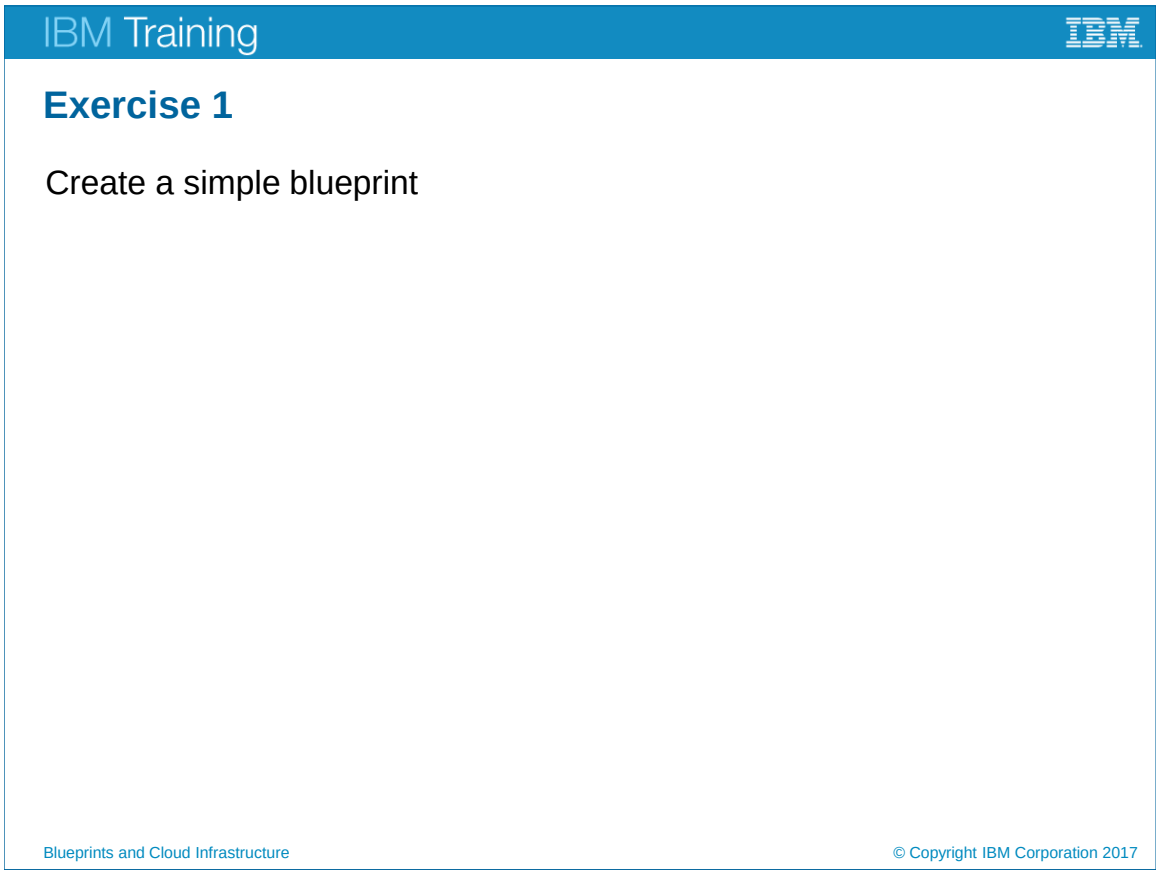
You model the cloud landscape by dragging and dropping the various components that are available in the palette. When you provision a blueprint into an environment, virtual compute resources are created and attached to a network.

IBM Training

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

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:

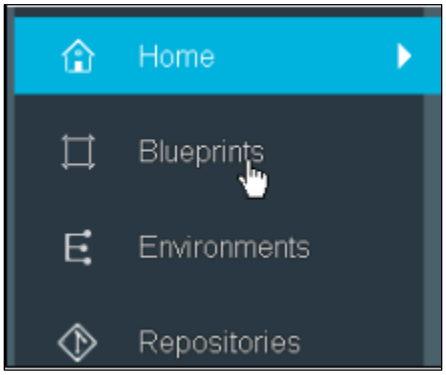
- Create a simple blueprint.
- Add virtual images to the blueprint.
- Add a private network to the blueprint.

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

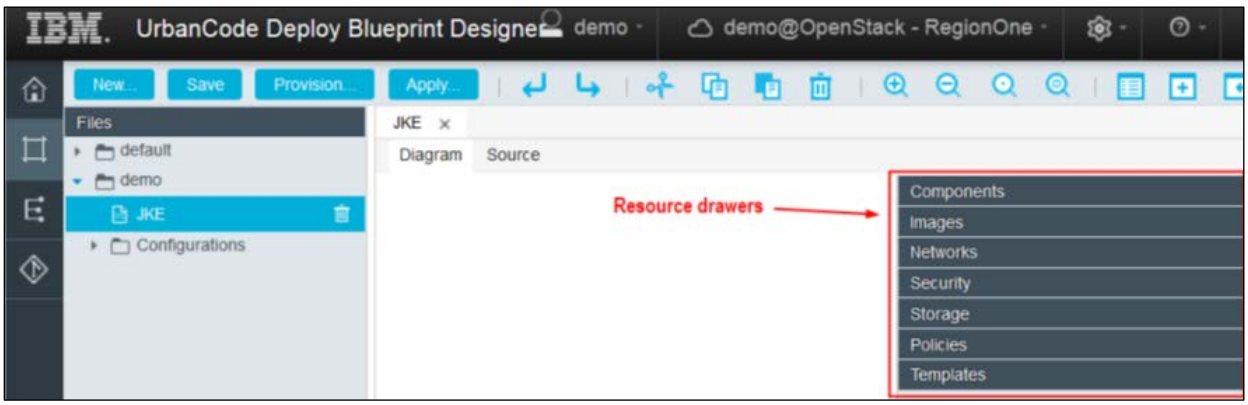
Exercise 1:
Tasks and results

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**.

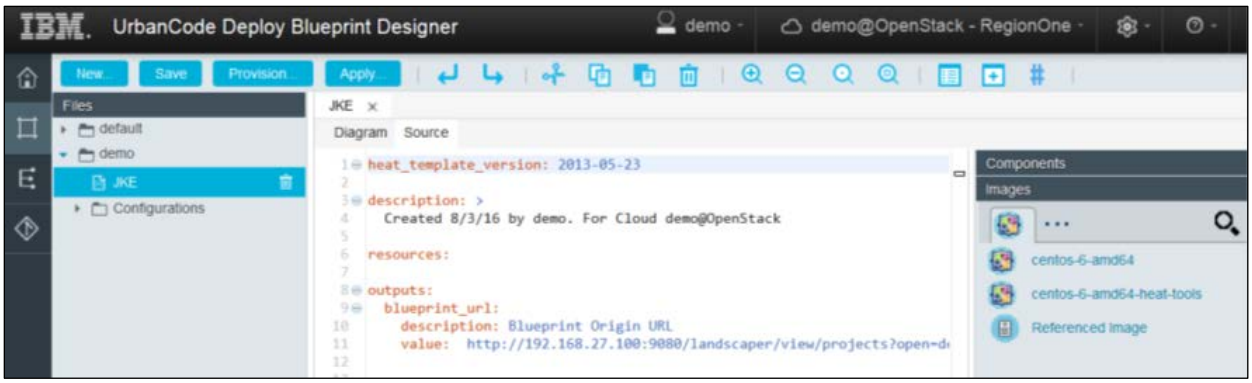


4. In the Files panel, verify that the **demo** folder is selected, and click **New**.
5. Specify the following information for the new blueprint (Heat stack):
   - In the **Name** field, type JKE.
   - In the **Repository** field, select **demo**.
   - In the **Type** field, select **Blueprint**.
6.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



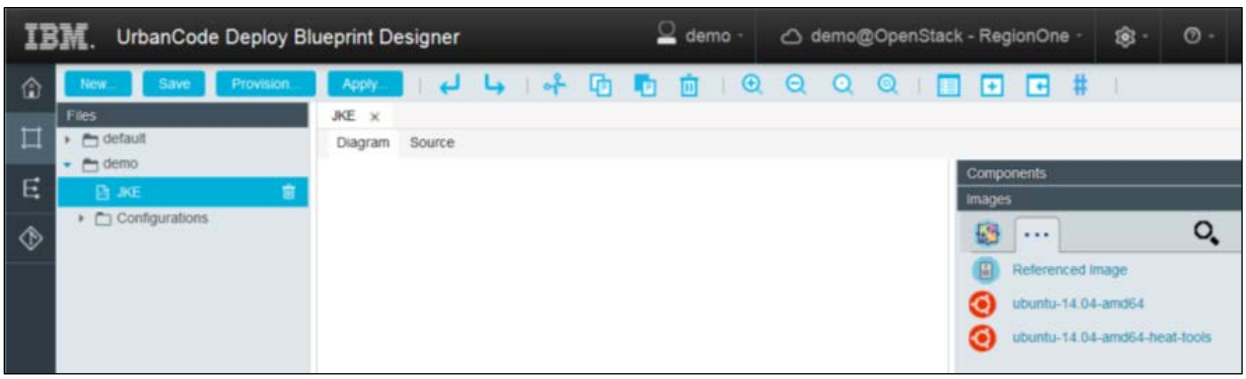
8. Switch back to the **Diagram** tab.

Task 2. Add virtual images to the blueprint.

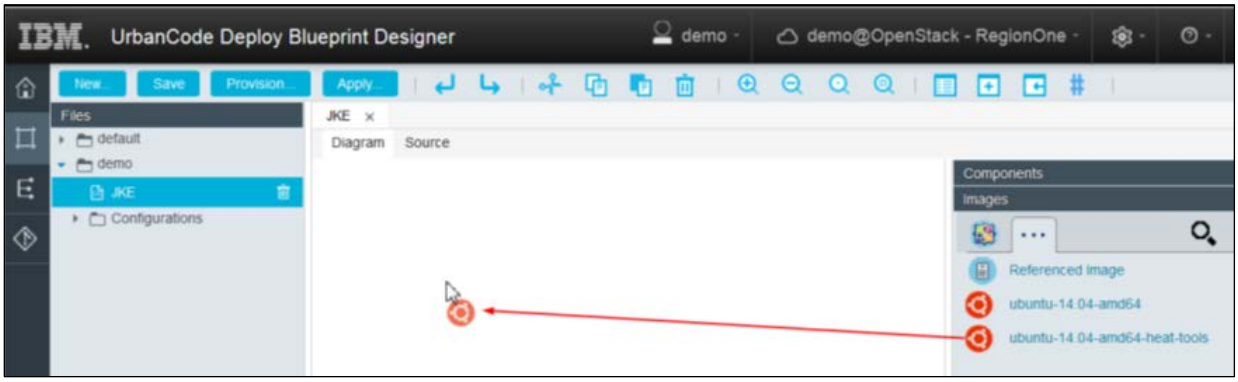
The Images drawer shows the virtual images that are available on the current cloud system, which is the OpenStack cloud that you connected to and explored in the previous lesson.

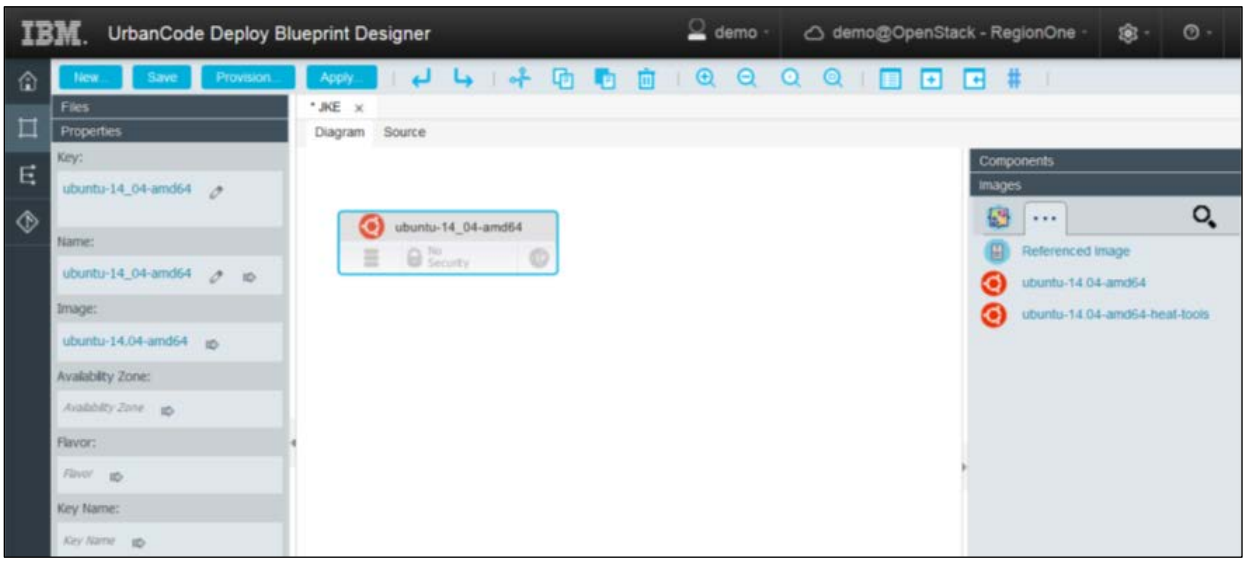
1. In the palette to the right, click the **Images** drawer. You can see that the two Centos images are the same ones that you saw in the OpenStack Images section.
2. Click the second tab with the ellipses (…). You can see the two Ubuntu images that you viewed in the OpenStack Images section in the previous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

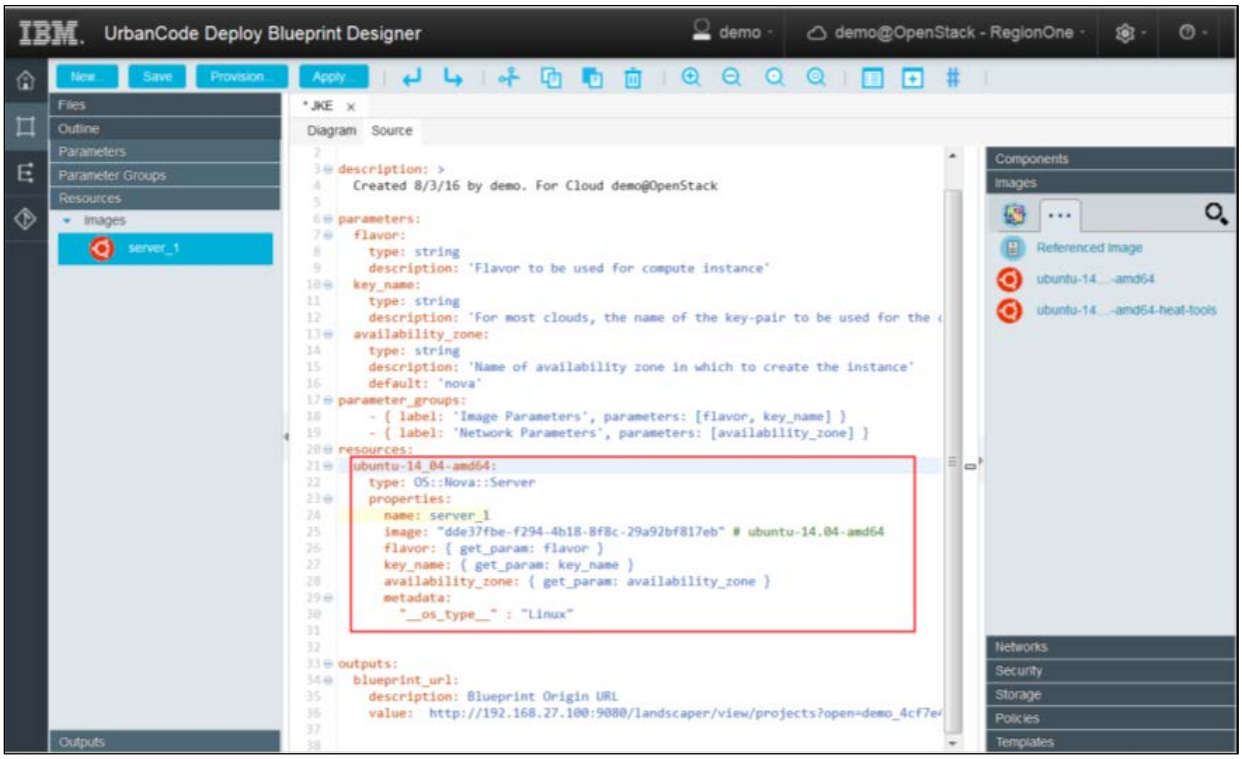




4. In the **Name** field, click the **Edit** icon  , type `server_1`, and press **Enter**.

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.



   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**.

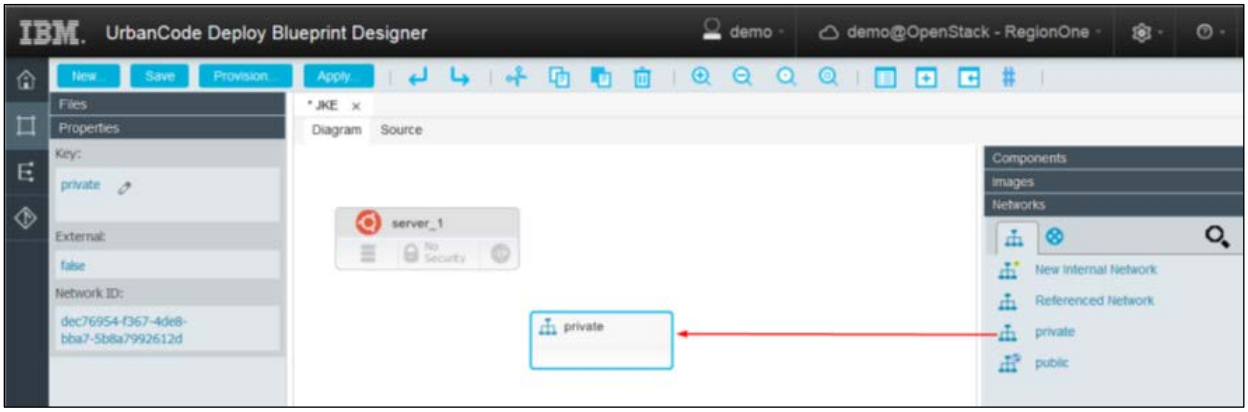
Task 3. Add a private network to the blueprint.

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

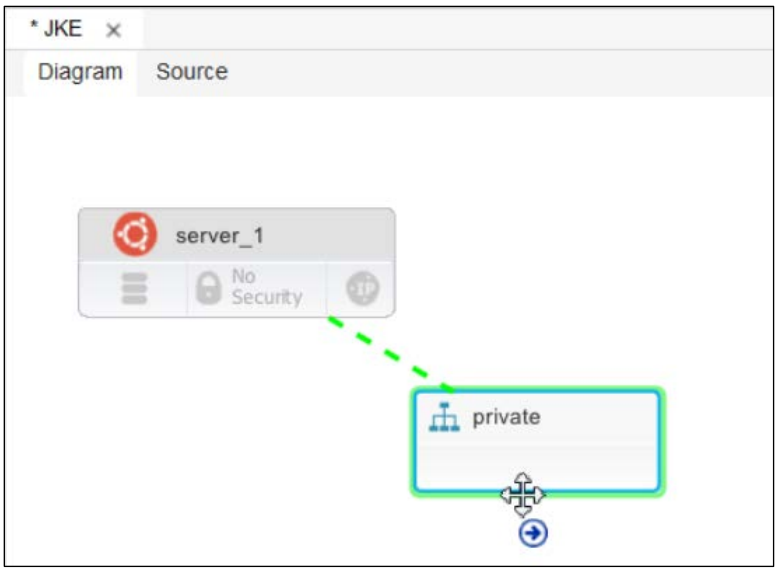
1. Click the **Network** drawer, which is located under the Images drawer.

   The New Internal Network resource represents a new network. The Referenced Network resource represents a network for which you provide the information later.

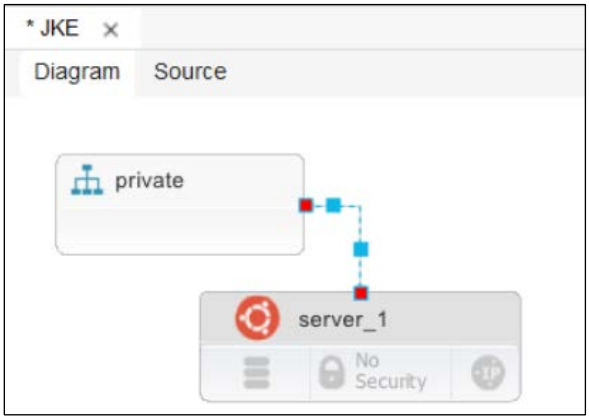
2. Drag the **private** network to the canvas.



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.

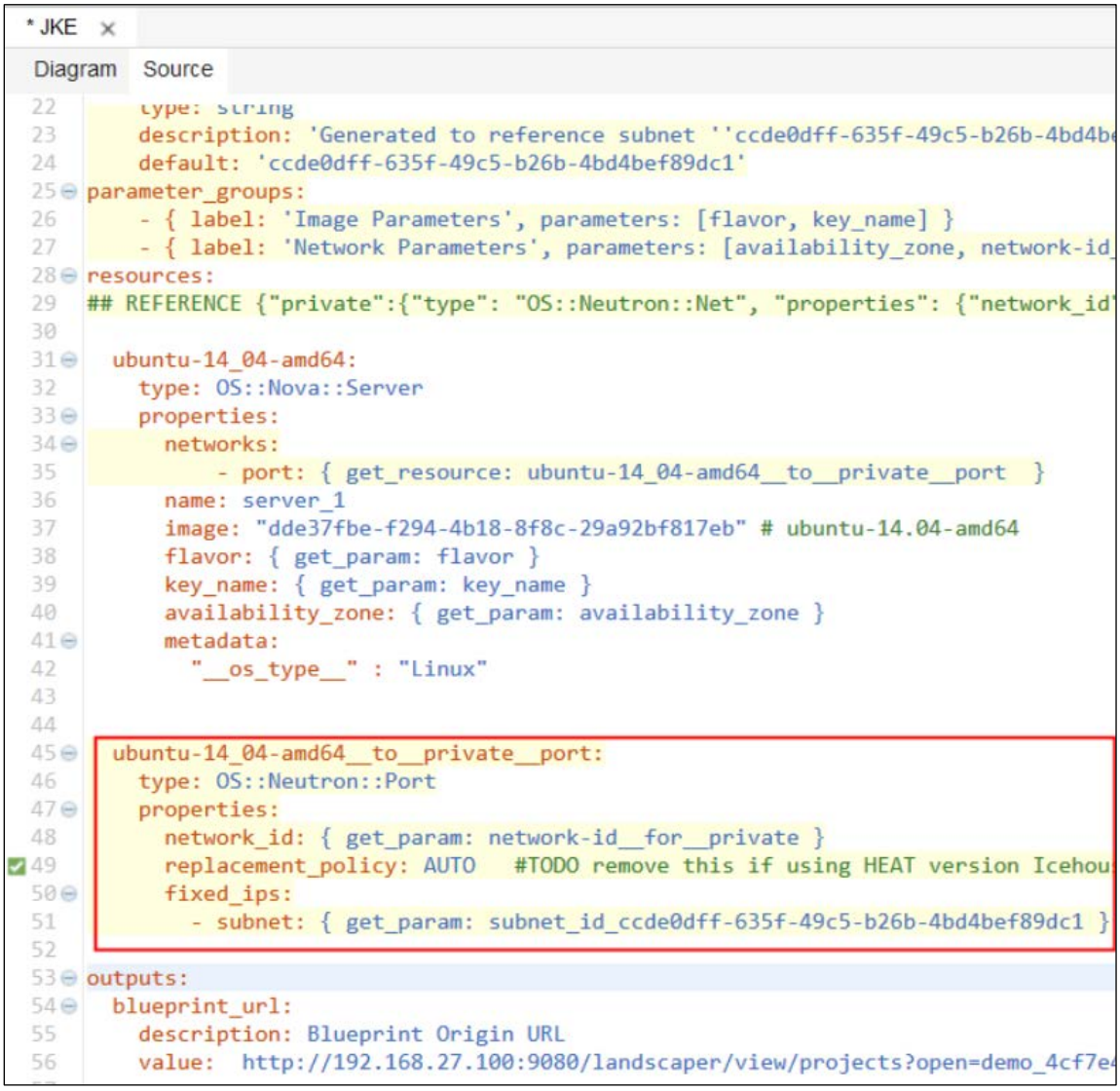


After the connection is complete, a line connects the two resources. This compute resource will be the host for the MySQL database and the WebSphere Liberty Profile web server components (from IBM UrbanCode Deploy) in subsequent labs.



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.

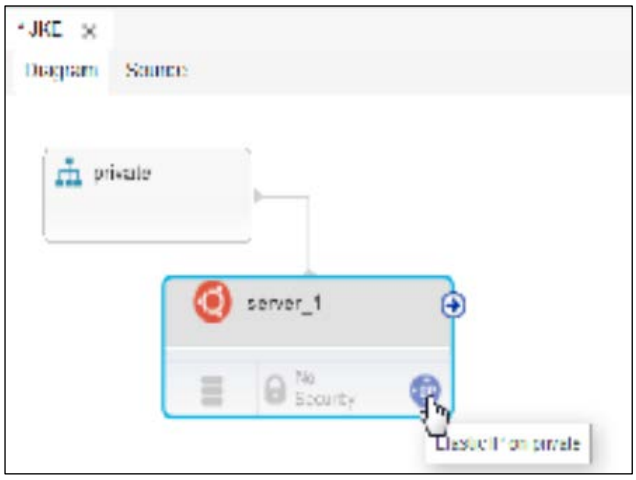
4. Switch to the **Source** tab to see this addition to the template:



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.

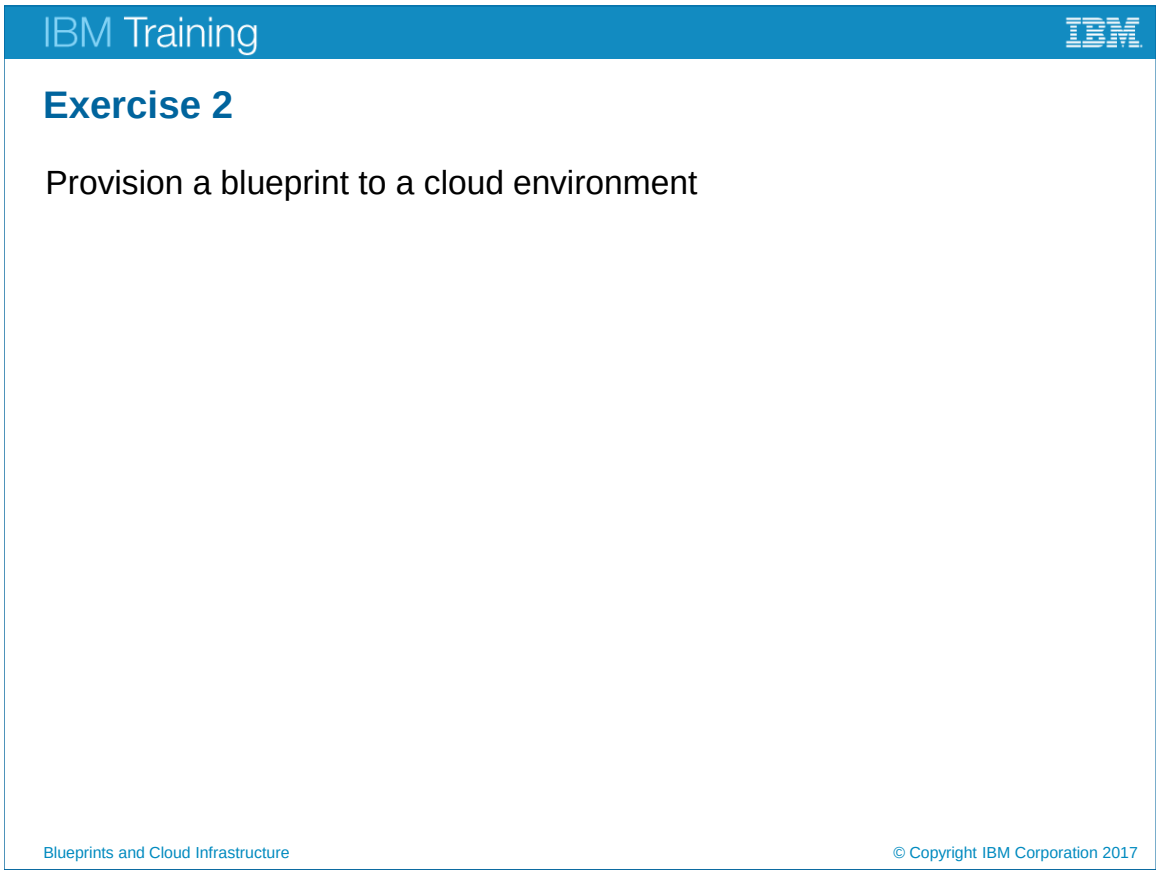
6. Click the **IP** icon  in the lower-right corner of the `server_1` image. The icon color turns blue.



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

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.

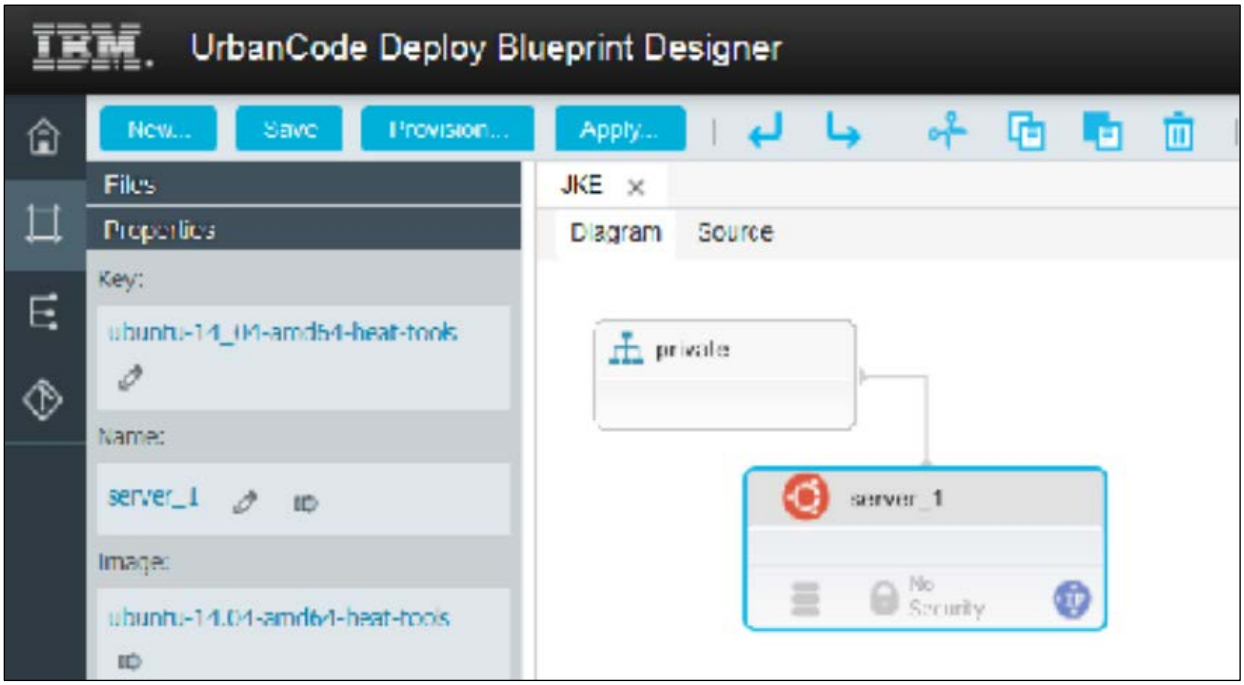
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 2: Tasks and results

Task 1. Provision the blueprint to a cloud environment.

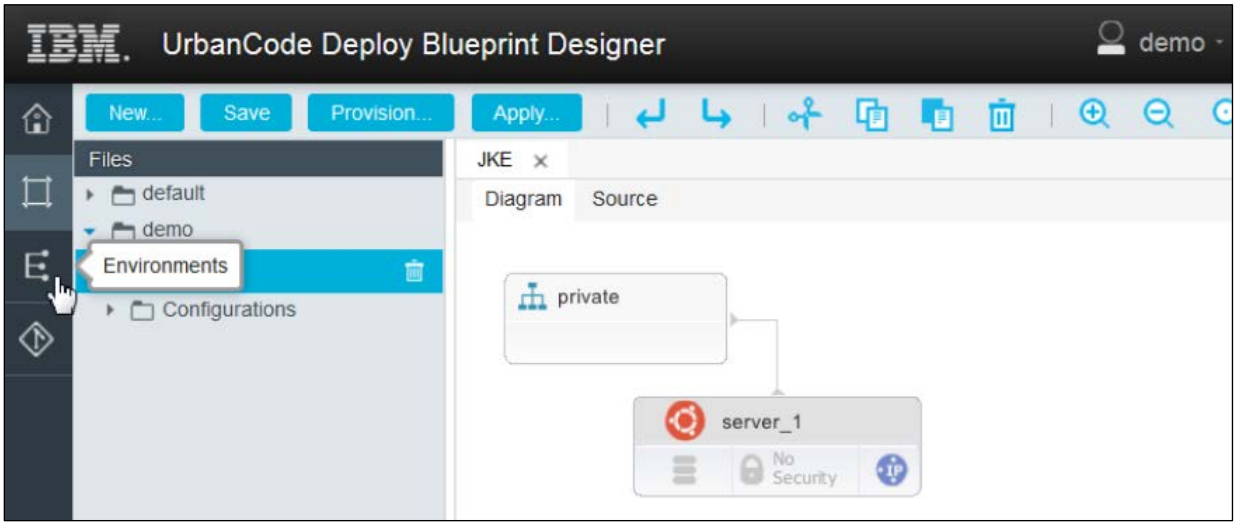
When you provision a cloud environment, you set aside resources on the cloud that you specified through a blueprint.

1. Click **Provision**.

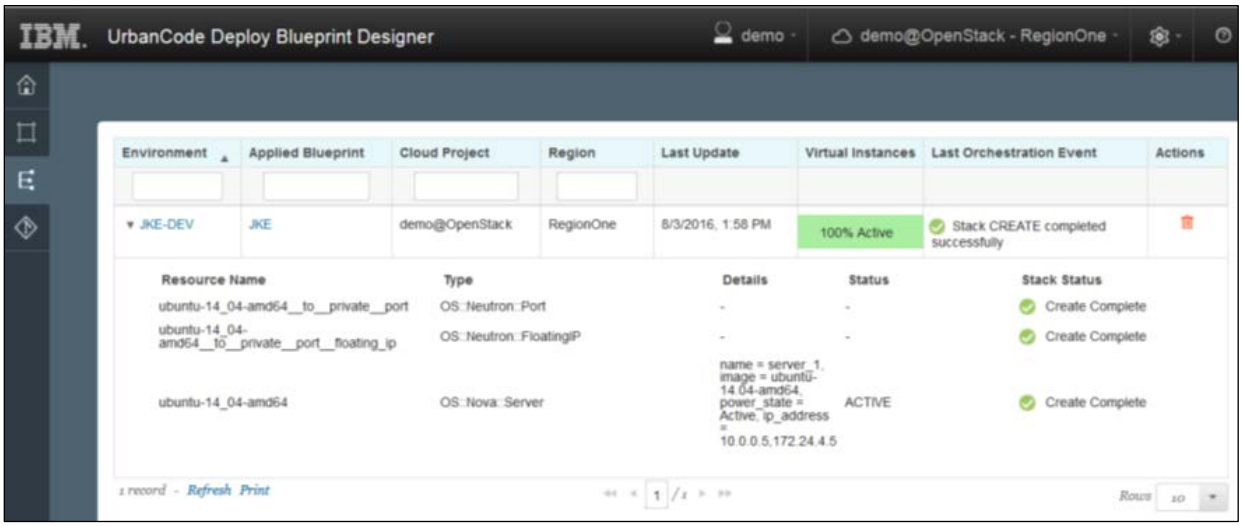


2. In the “Provision Blueprint to new Environment” window, configure as follows:
   - **Environment Name**: `JKE-DEV`
   - Click **Image Parameters**.
     - In the **Flavor** field, select **m1.medium.** The flavor represents the size of a provisioned image.
     - In the **Key Name** field, select **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**.
3. Click **Provision**.

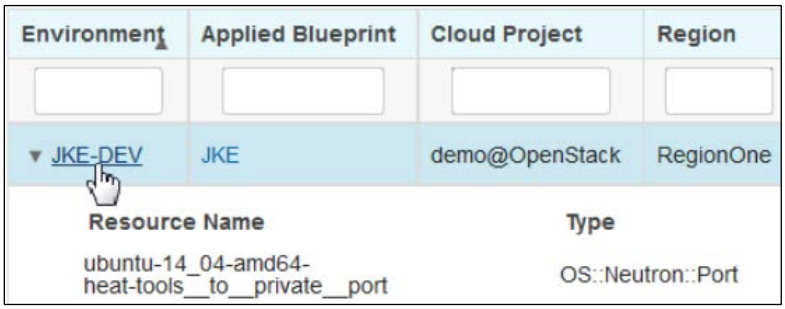
4. On the left side of the page, click the **Environments** tab.



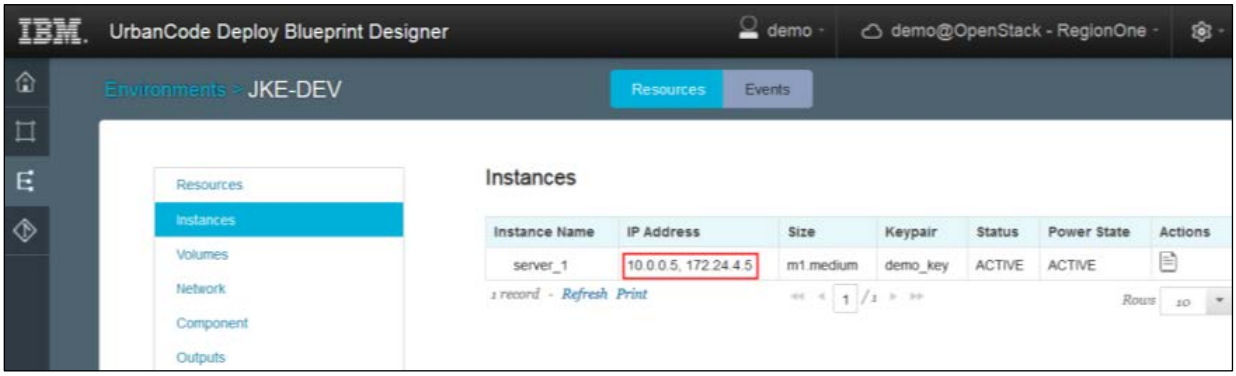
5. Click the twisty to expand the list of resources and observe the progress of the deployment.



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



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



   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.

IBM Training

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

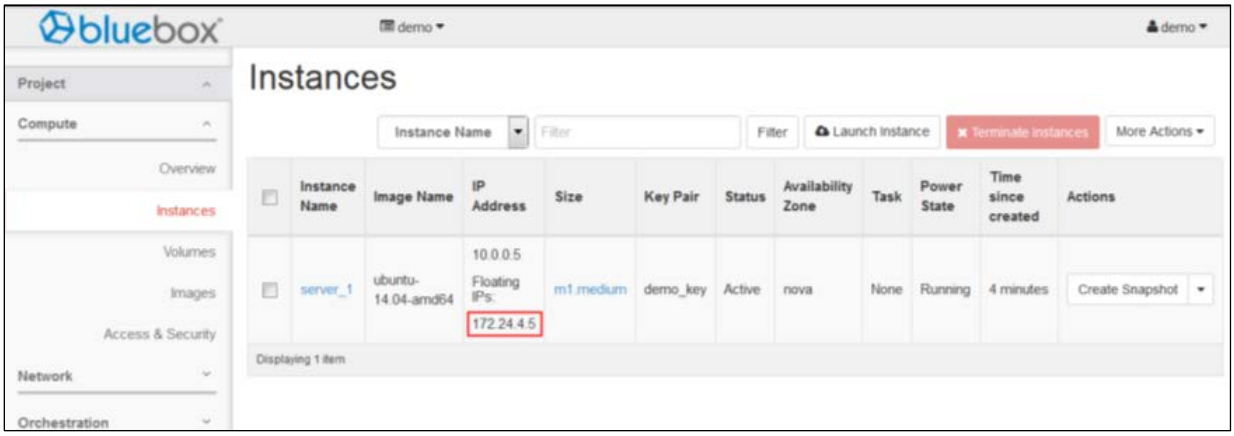
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

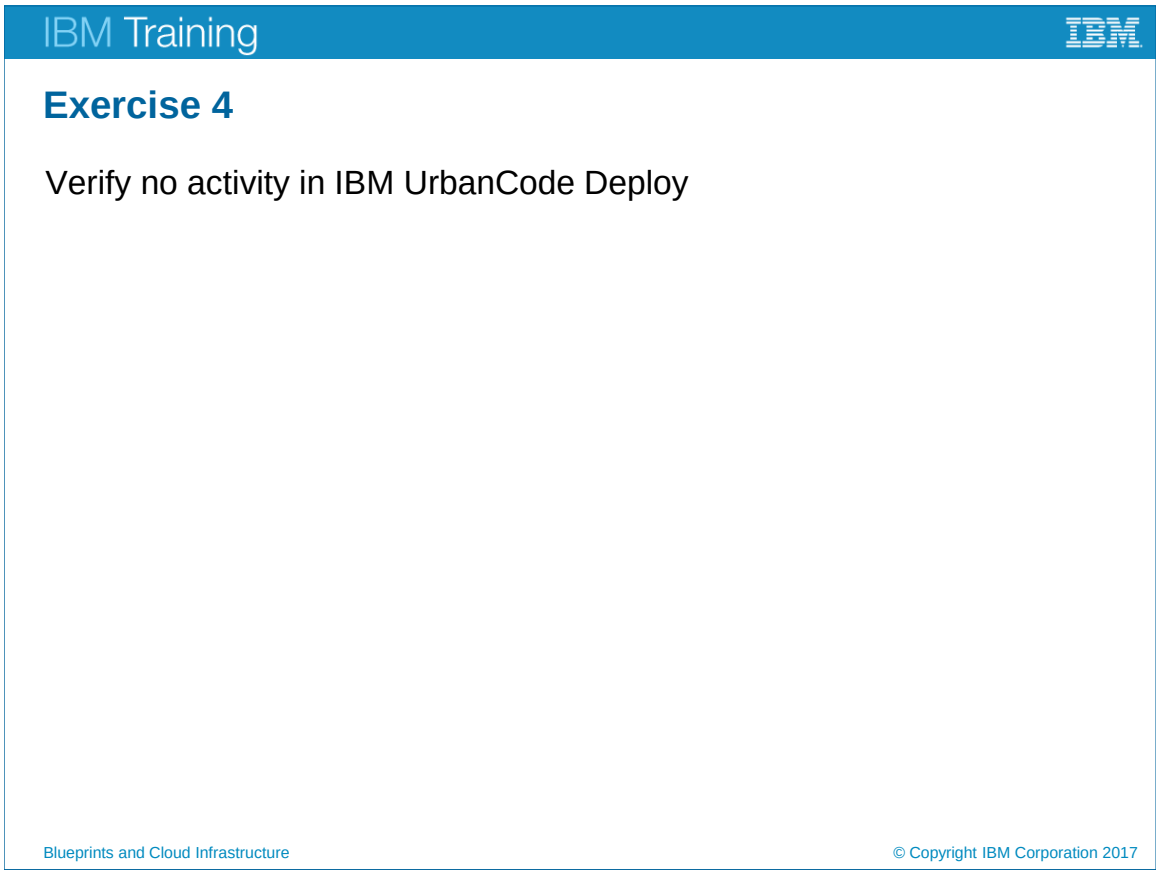
Exercise 3:
Tasks and results

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.





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:

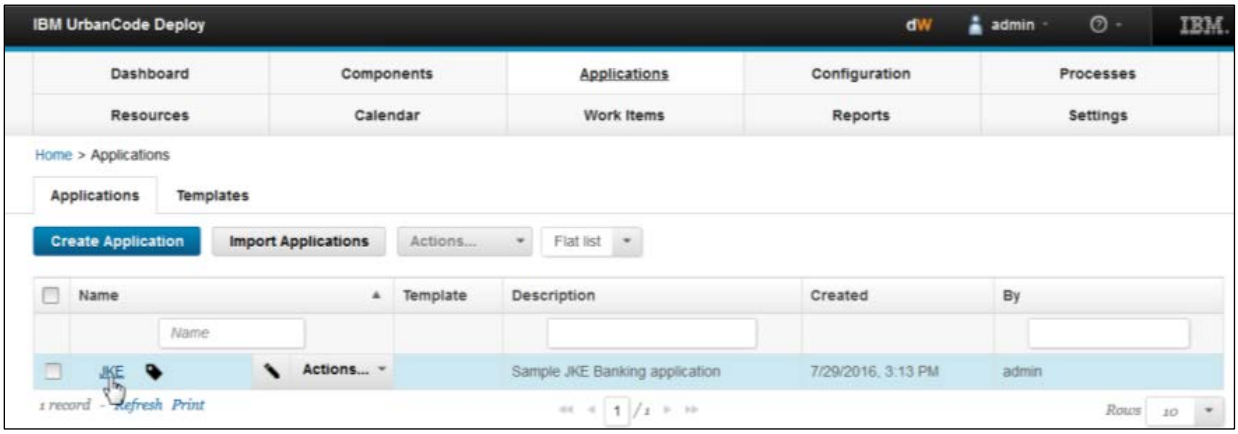
- Verify that there is no activity in IBM UrbanCode Deploy

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

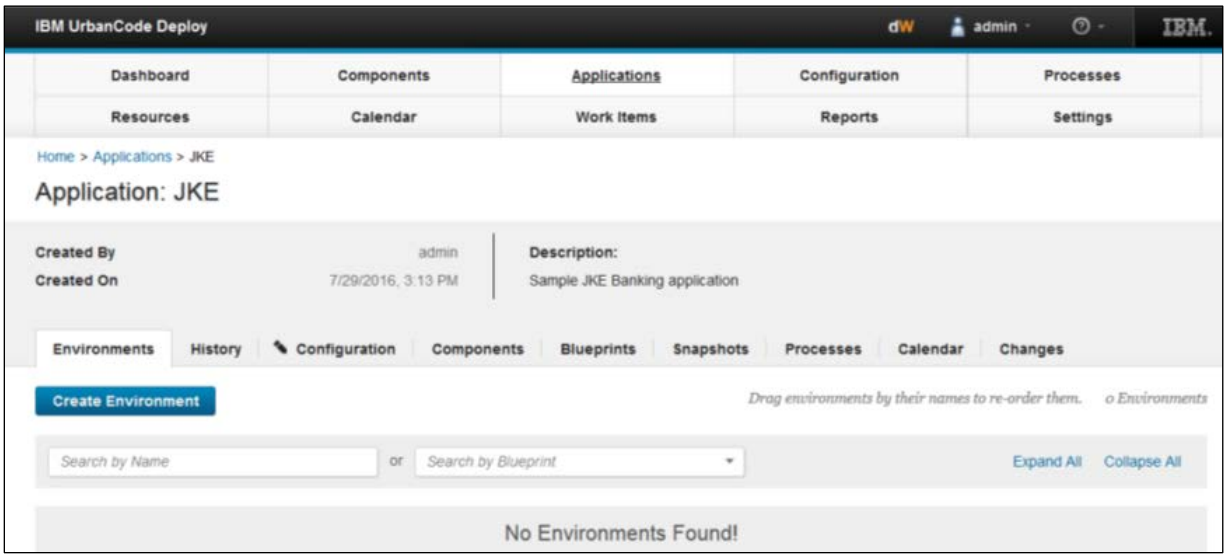
Exercise 4:
Tasks and results

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**.



Observe that no environments exist for the application.



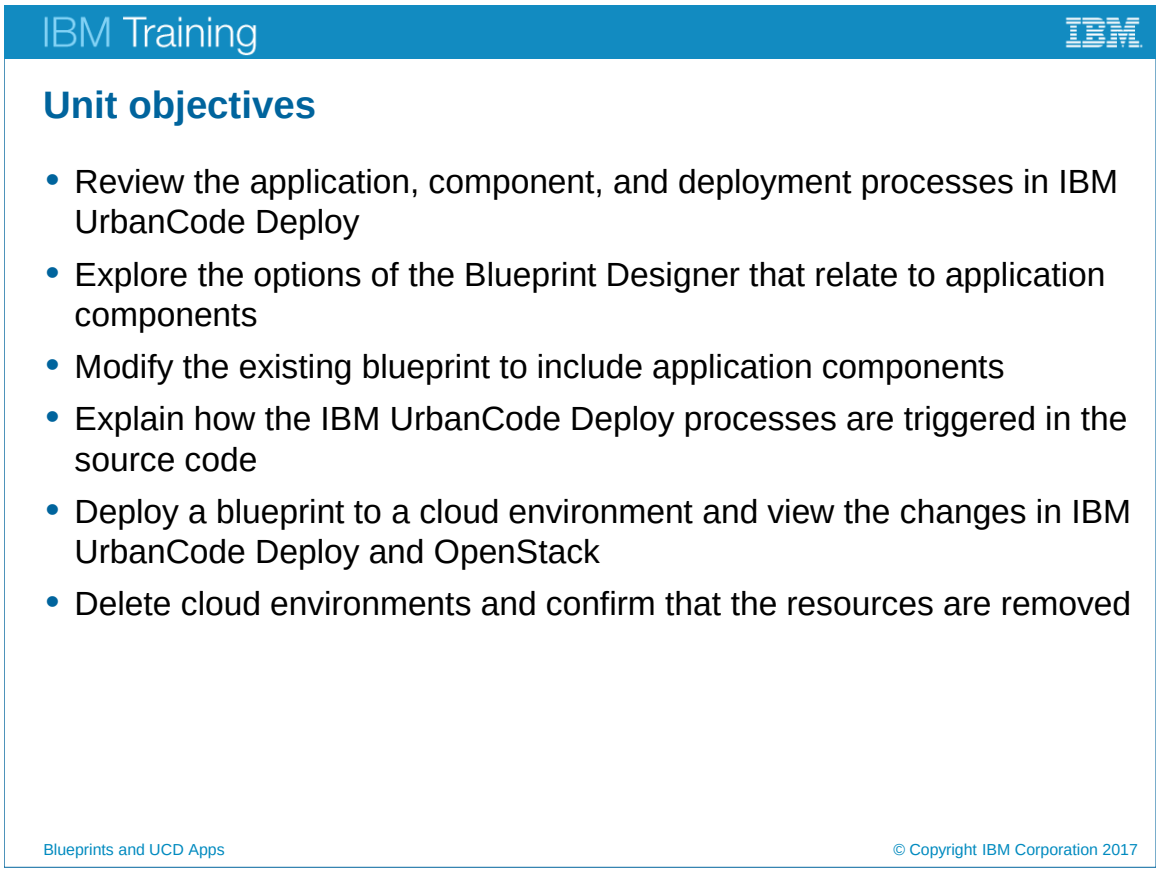
In the next unit, you install components to the OpenStack cloud. You will come back to this page to see the running environment.

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.

IBM Training

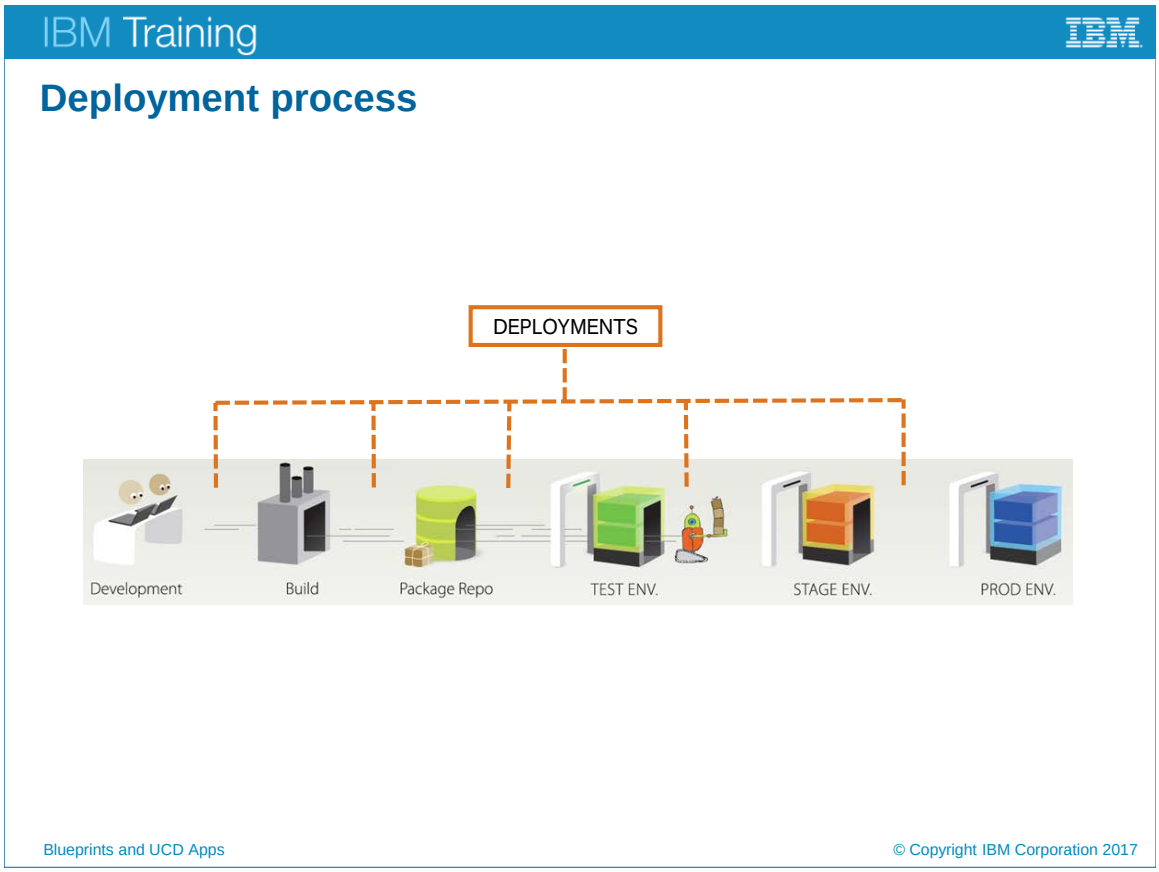
Blueprints and UCD Apps

IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1



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.

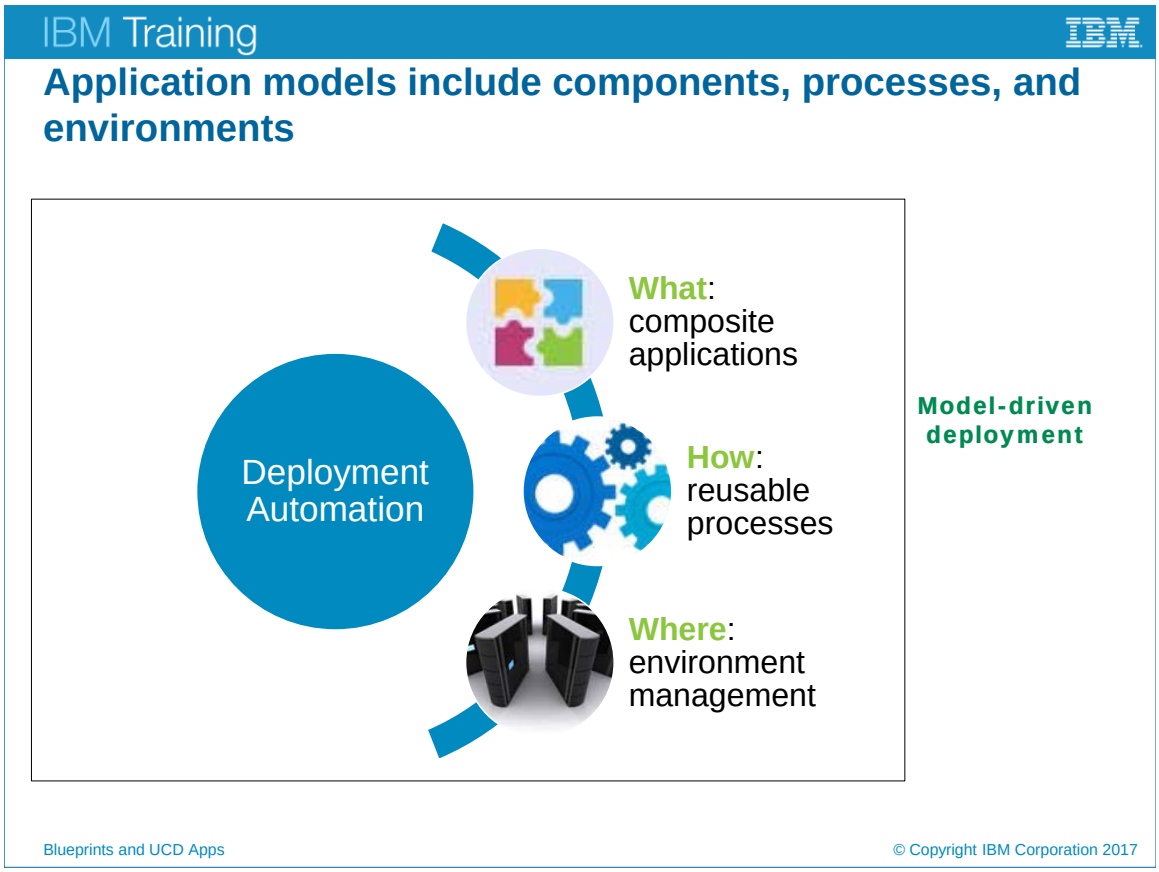


Deployment process

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:

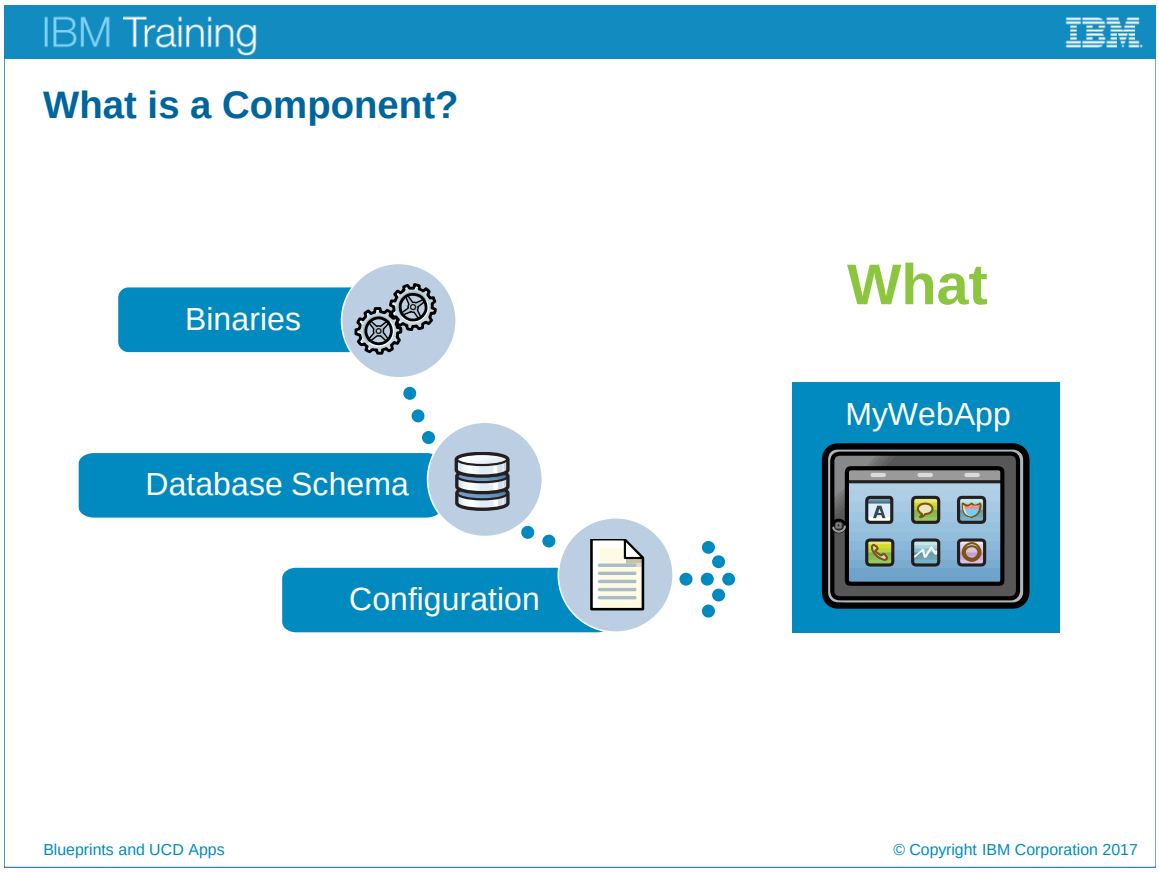
- **Development (DEV):** In the *Development* environment, developers build and deploy code in a test lab, and the development team tests the application at the most basic level. When the application meets certain criteria for advancement, it moves to the next environment.
- **System Integration Testing (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

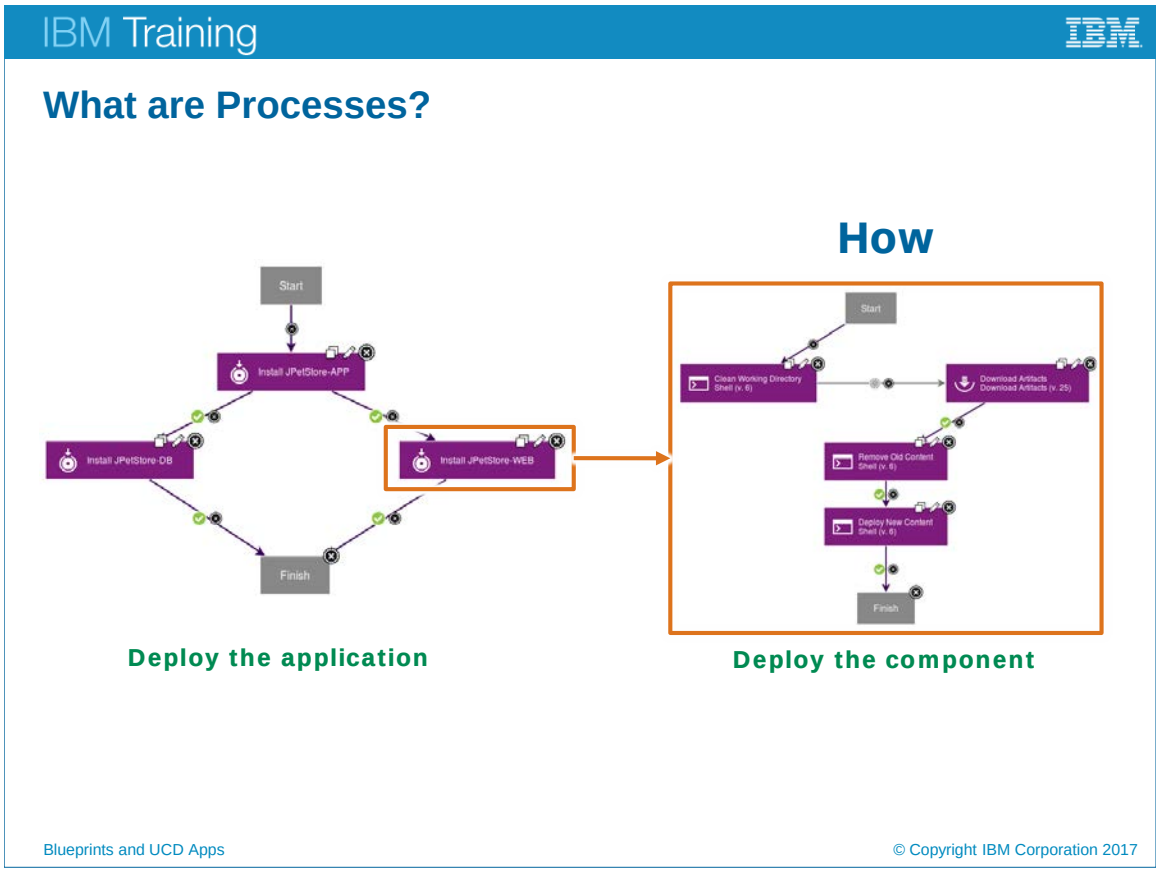
IBM UrbanCode Deploy focuses on model-driven deployment. You model a plan to run software deployments, and then you reuse these processes to create automation.

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?

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.

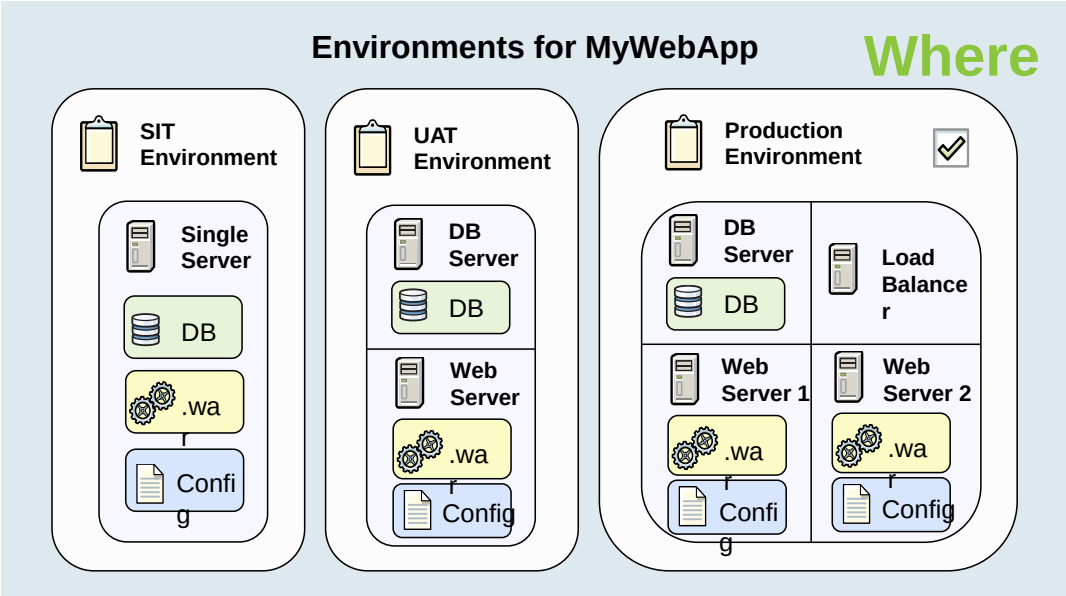


What are Processes?

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.

IBM Training

Environment



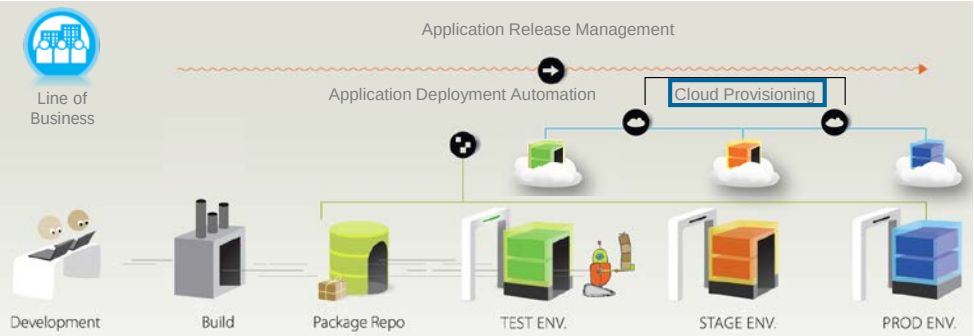
Blueprints and UCD Apps

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.

IBM Training

Provisioning infrastructure is the current bottleneck in the delivery pipeline



Blueprints and UCD Apps

Provisioning infrastructure is the current bottleneck in the delivery pipeline

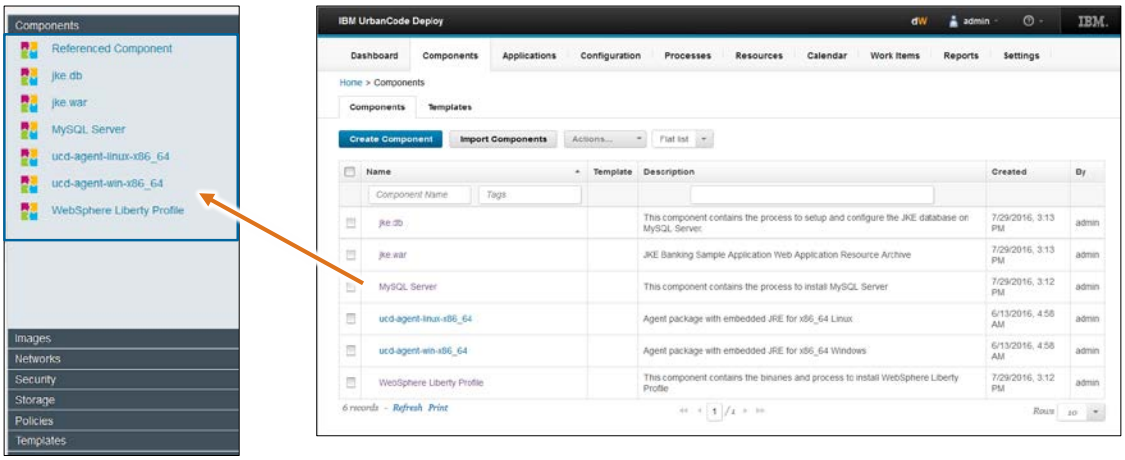
Development and testing teams are now optimized with agile, service virtualization, and deployment automation. Developers are increasingly becoming DevOps practitioners, demanding access to the full-environment from application-layer and down. This shift puts pressure on full-stack provisioning, which 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.

IBM Training

The Components drawer



Blueprints and UCD Apps

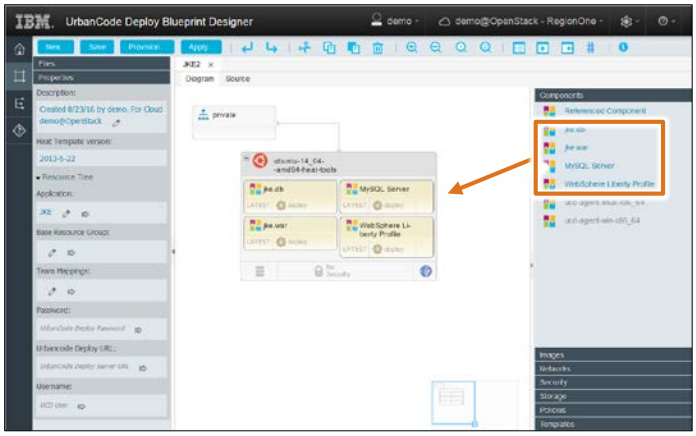
The Components drawer

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.

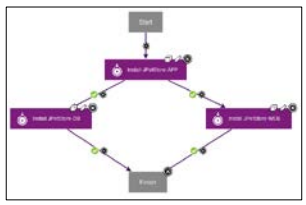
The components that you can see in the blueprint editor depend on your permissions. If your blueprint design server account is connected to an account on the IBM UrbanCode Deploy server, you see the components that you have access to on that server.

IBM Training

Components adding to a blueprint



Application process in IBM UrbanCode Deploy

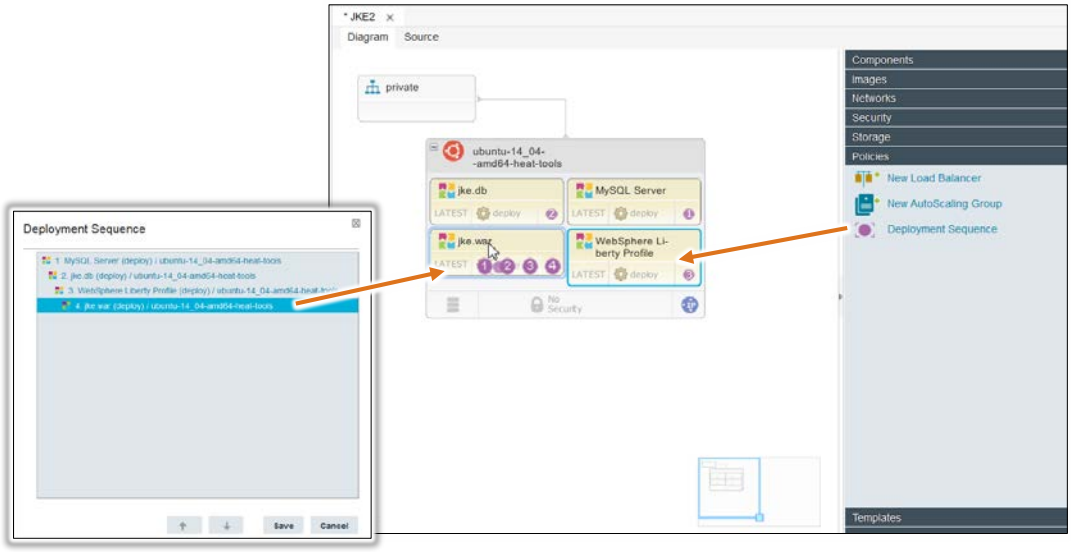


Components adding to a blueprint

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

Deploying sequence for components



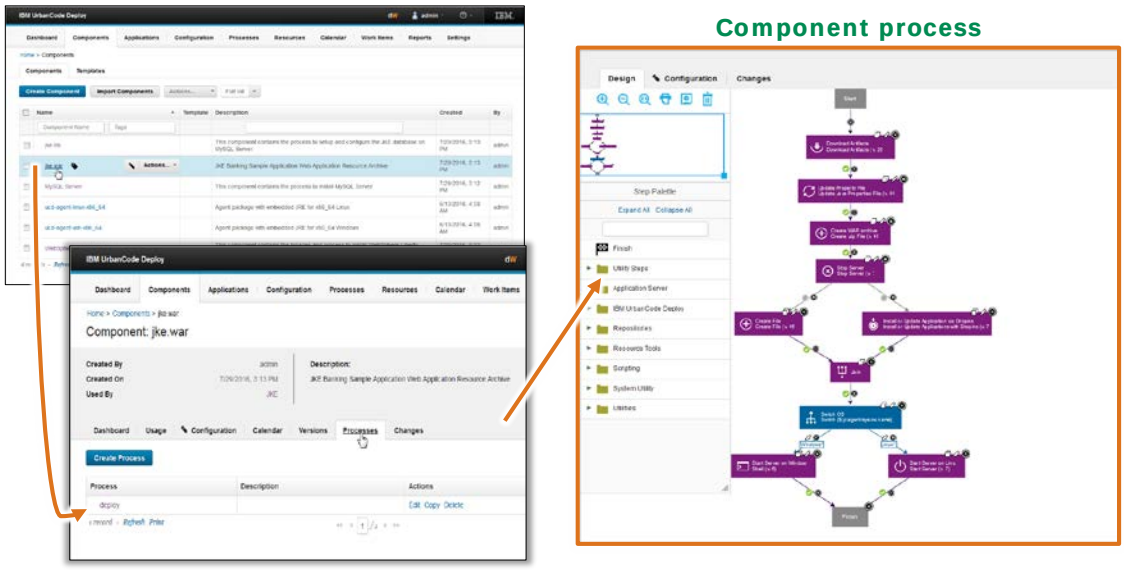
Blueprints and UCD Apps

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.

IBM Training

Component process in IBM UrbanCode Deploy



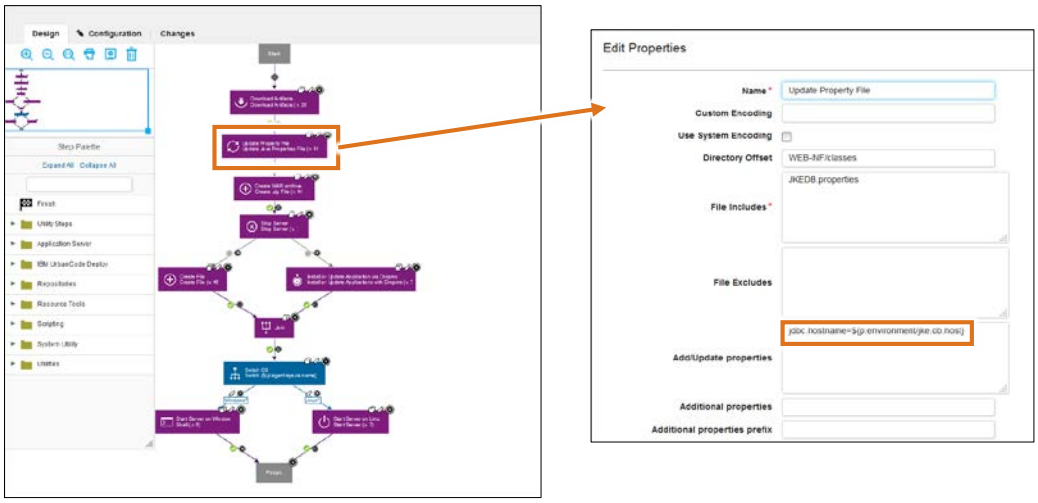
Blueprints and UCD Apps

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.

IBM Training

Process steps and their environment properties



Blueprints and UCD Apps

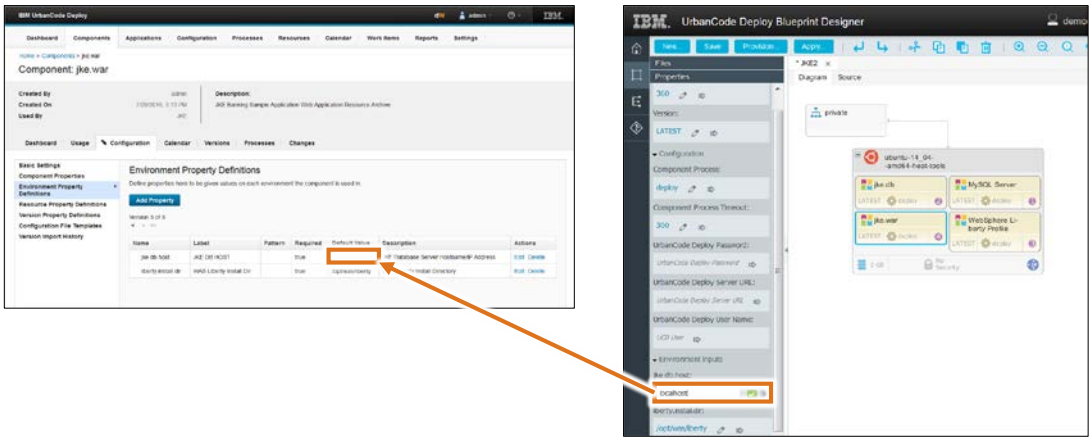
Process steps and their environment properties

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

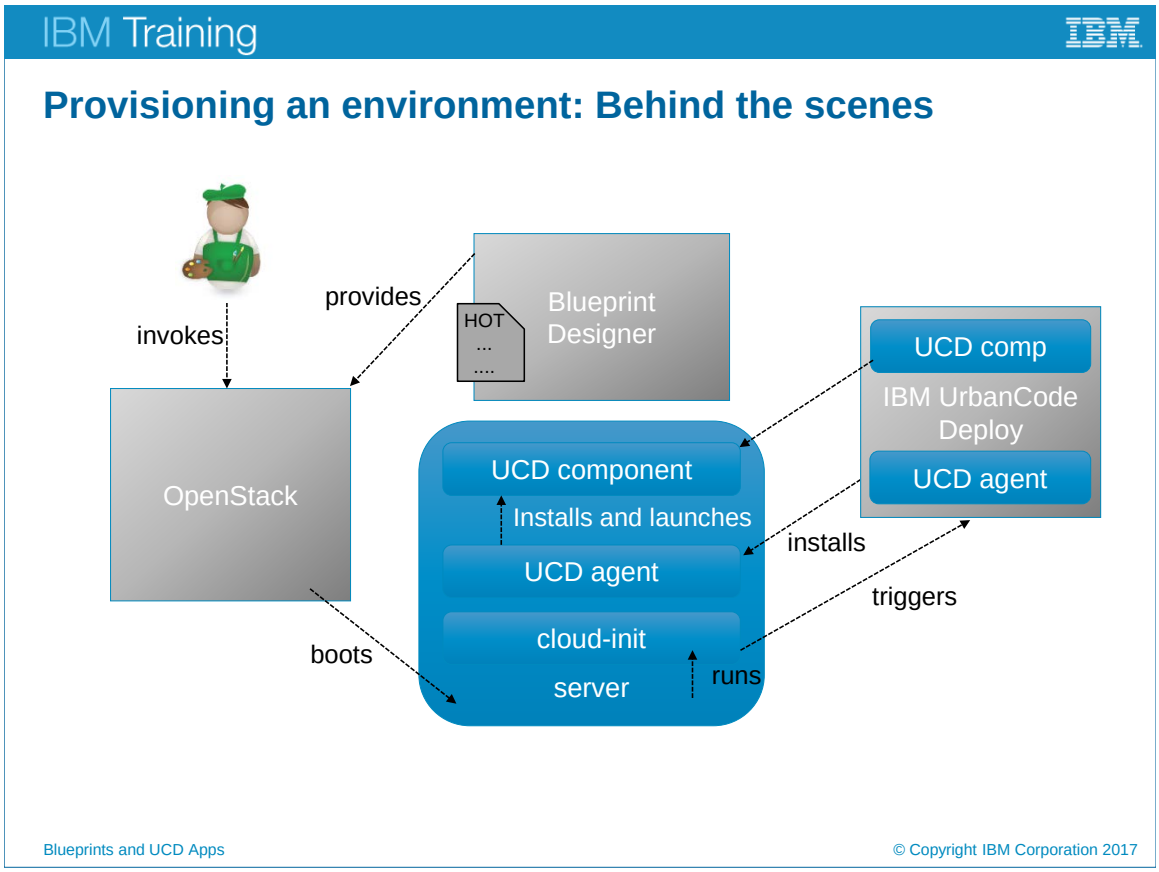
Setting properties in the Blueprint Designer



Blueprints and UCD Apps

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

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.

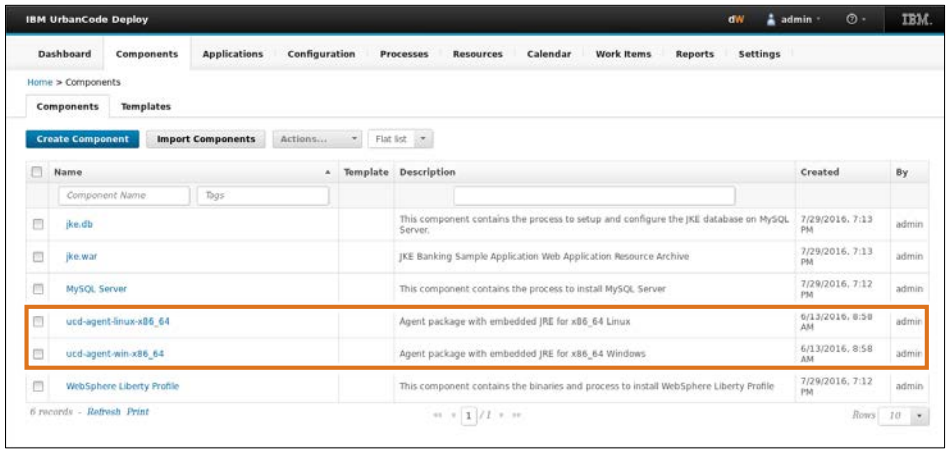
After the operating system is running, it runs cloud-init. Cloud-init is a set of scripts and utilities that cloud systems use to initialize and configure instances.

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

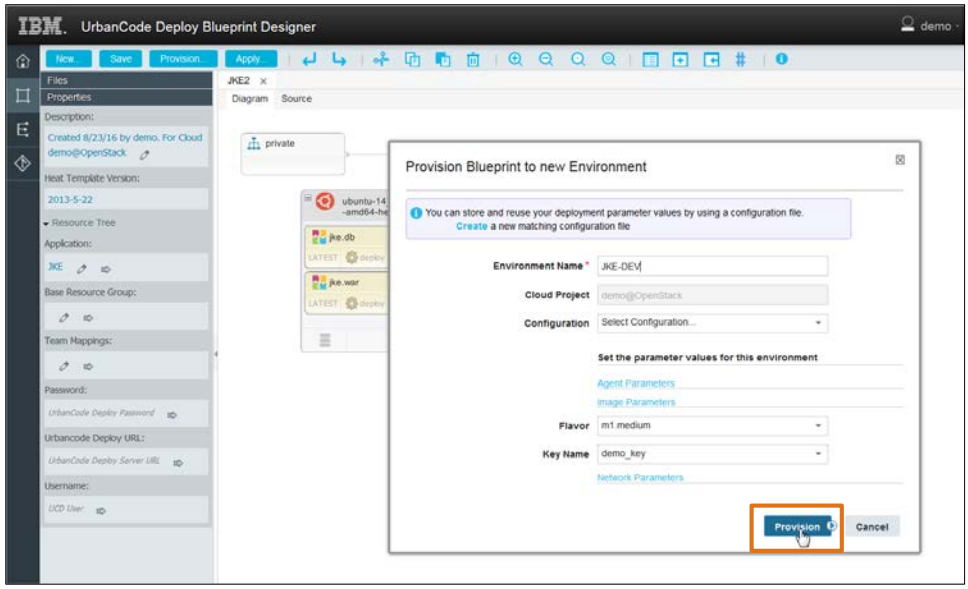


Agents on the IBM UrbanCode Deploy server

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

IBM Training

Provisioning of blueprint from the Blueprint Designer

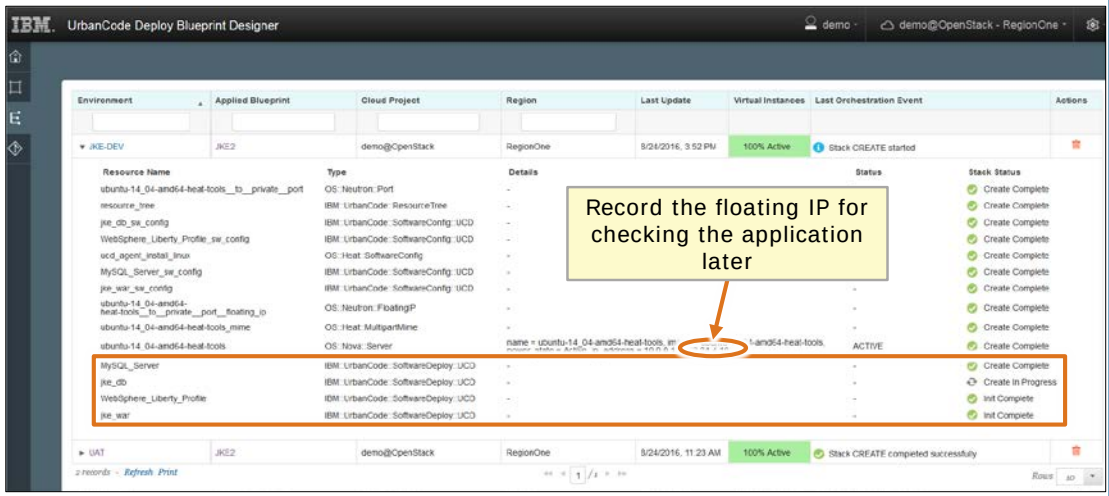


Provisioning of blueprint from the Blueprint Designer

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

Deployment of the components begins



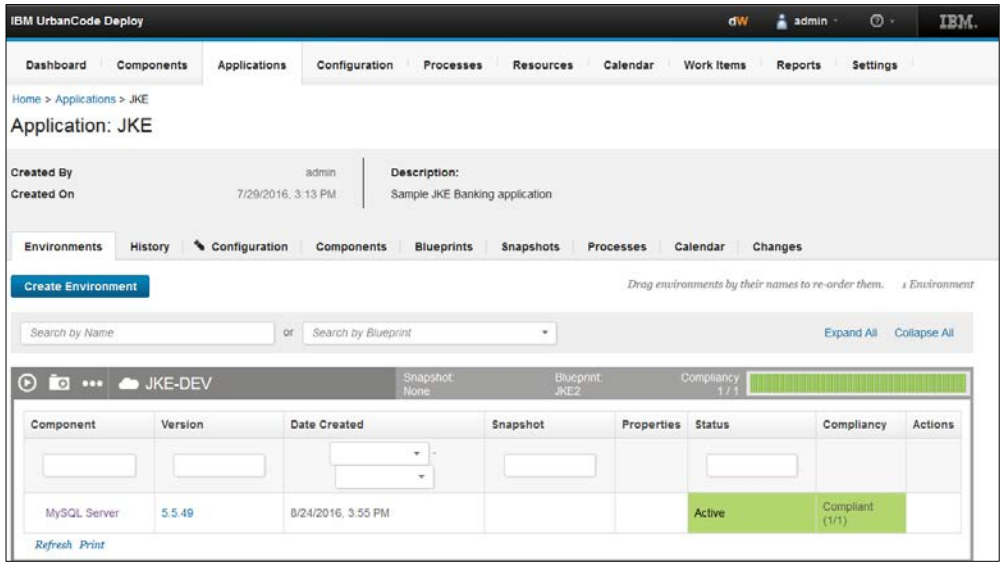
Blueprints and UCD Apps

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

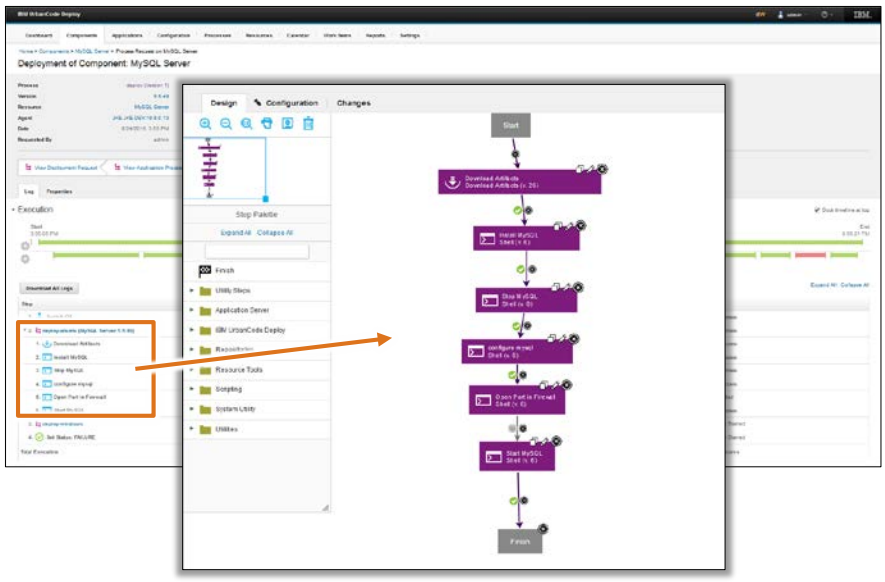


Components in IBM UrbanCode Deploy on running of deployment process

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



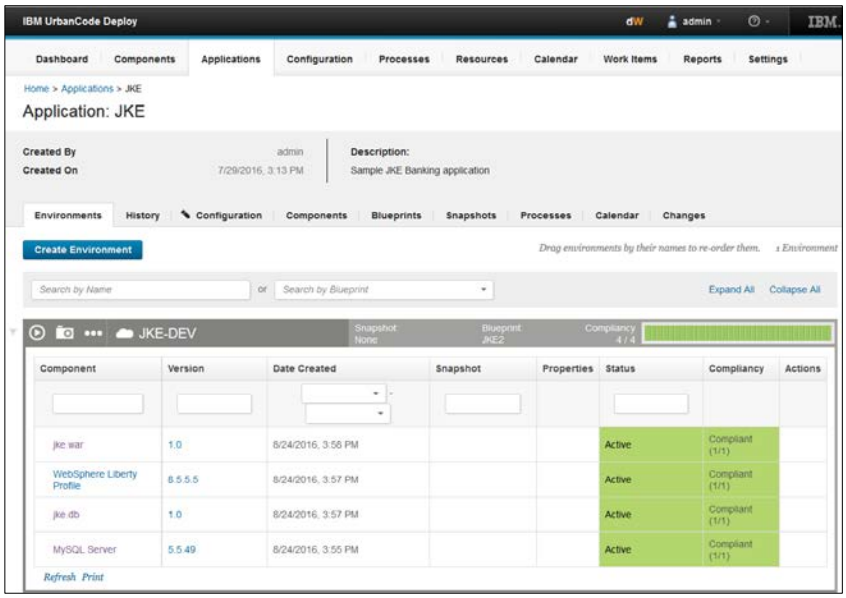
Blueprints and UCD Apps

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

Components deployed on the environment



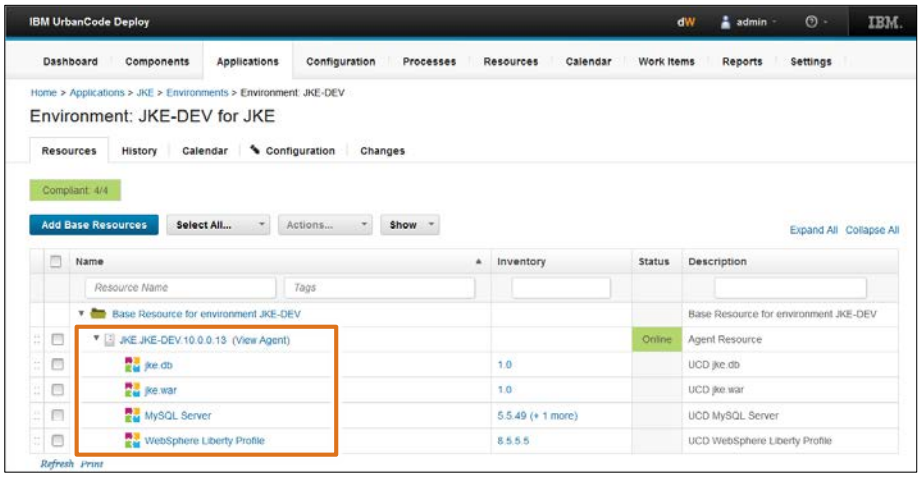
Blueprints and UCD Apps

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

Components are deployed to environments by agents

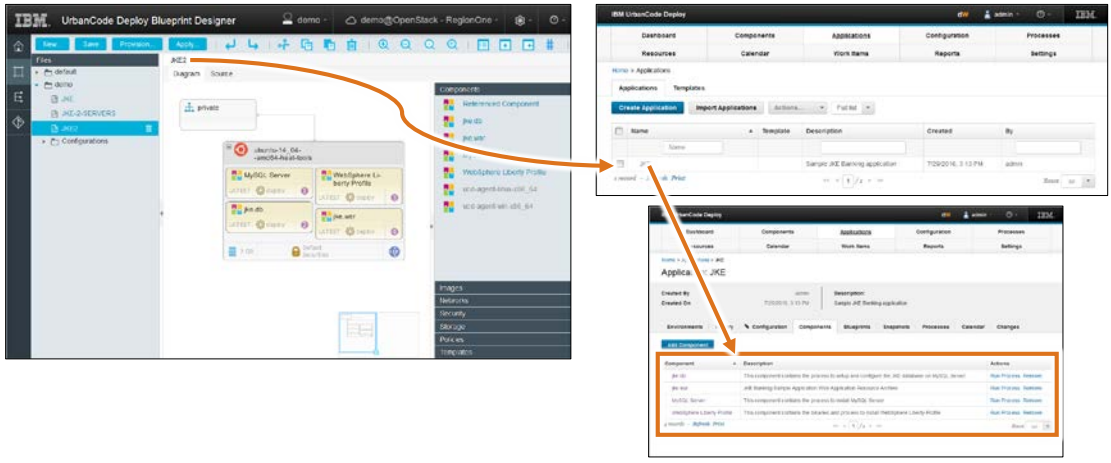


Components are deployed to environments by agents

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



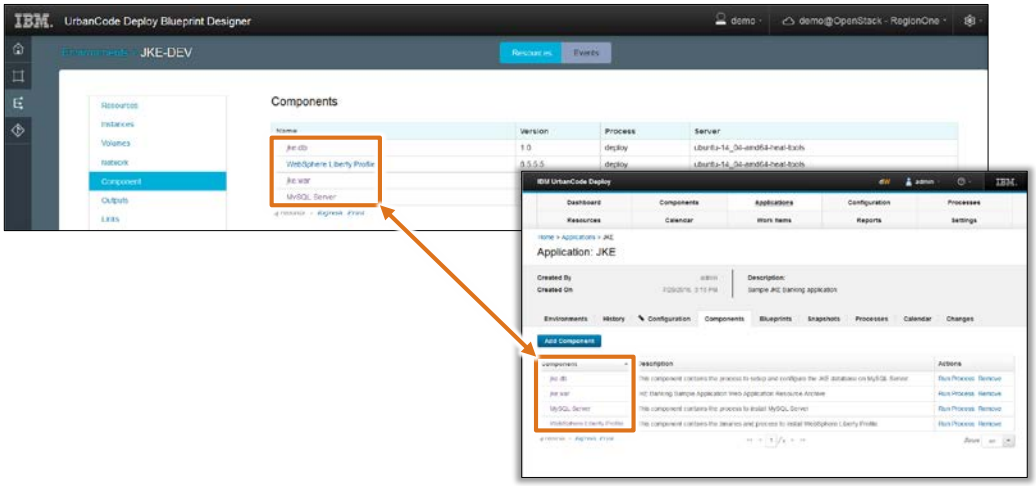
Blueprints and UCD Apps

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

Components deployed shown on the Environment tab



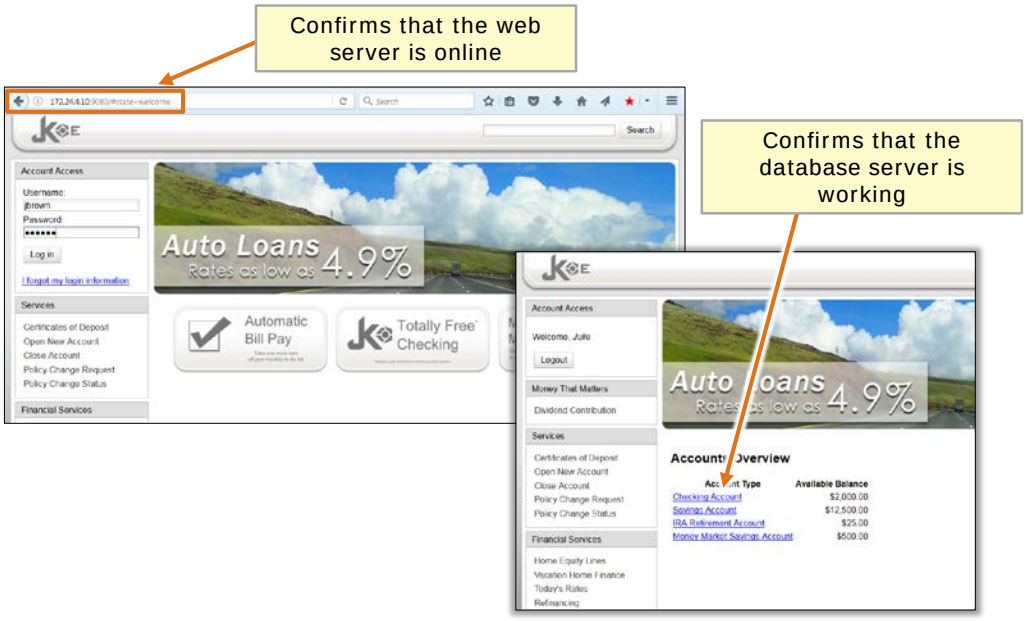
Blueprints and UCD Apps

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

Successfully provisioned environment



Blueprints and UCD Apps

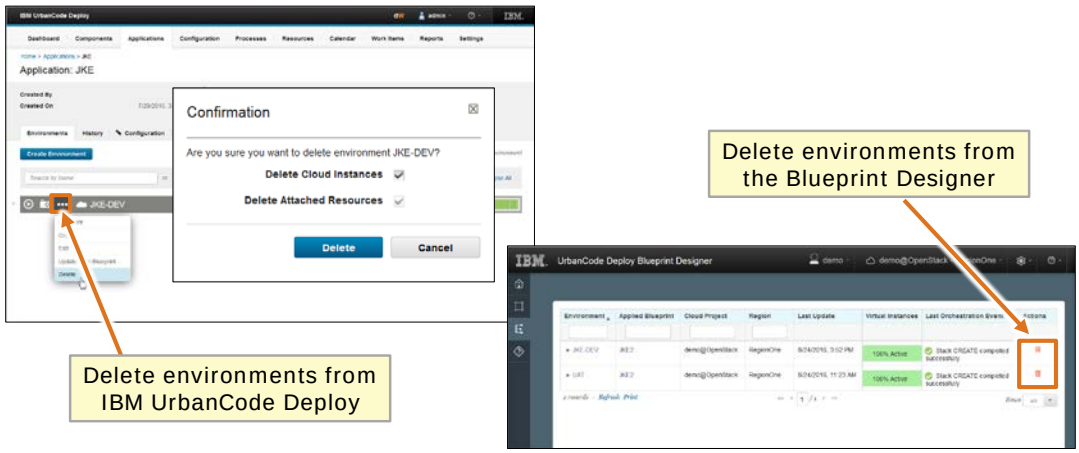
Successfully provisioned environment

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.

IBM Training

Deleting an environment from either UrbanCode Deploy or Blueprint Designer



Blueprints and UCD Apps

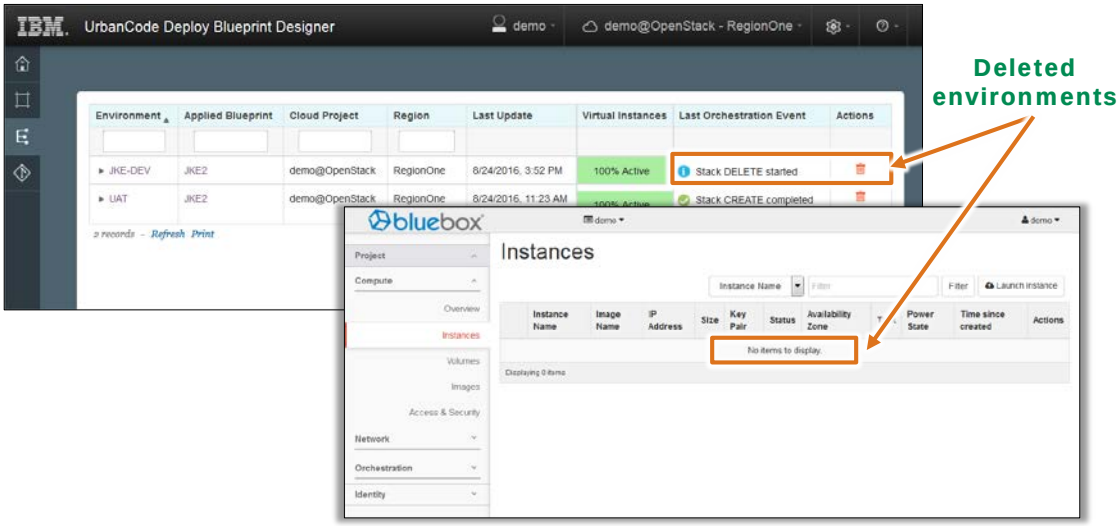
Deleting an environment from either UrbanCode Deploy or Blueprint Designer

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

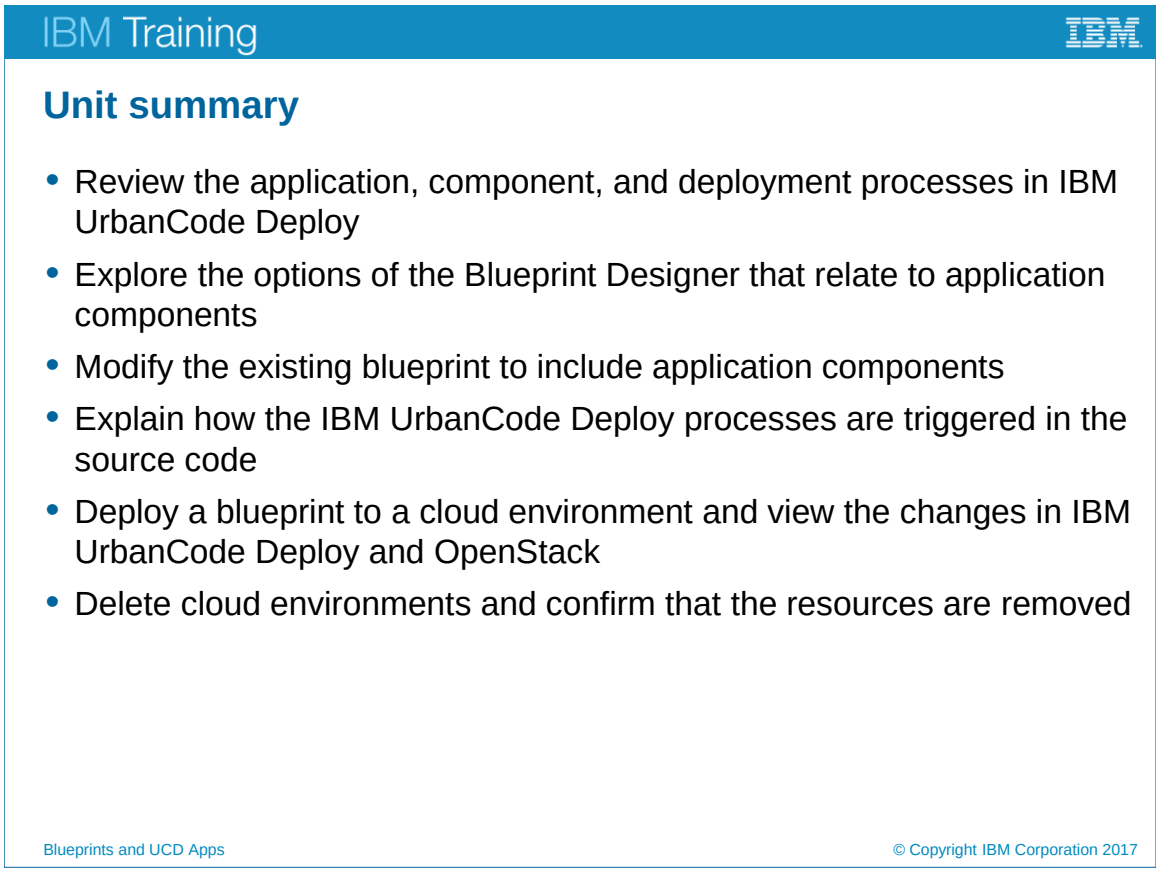
Environments are deleted across all tools



Blueprints and UCD Apps

Environments are deleted across all tools

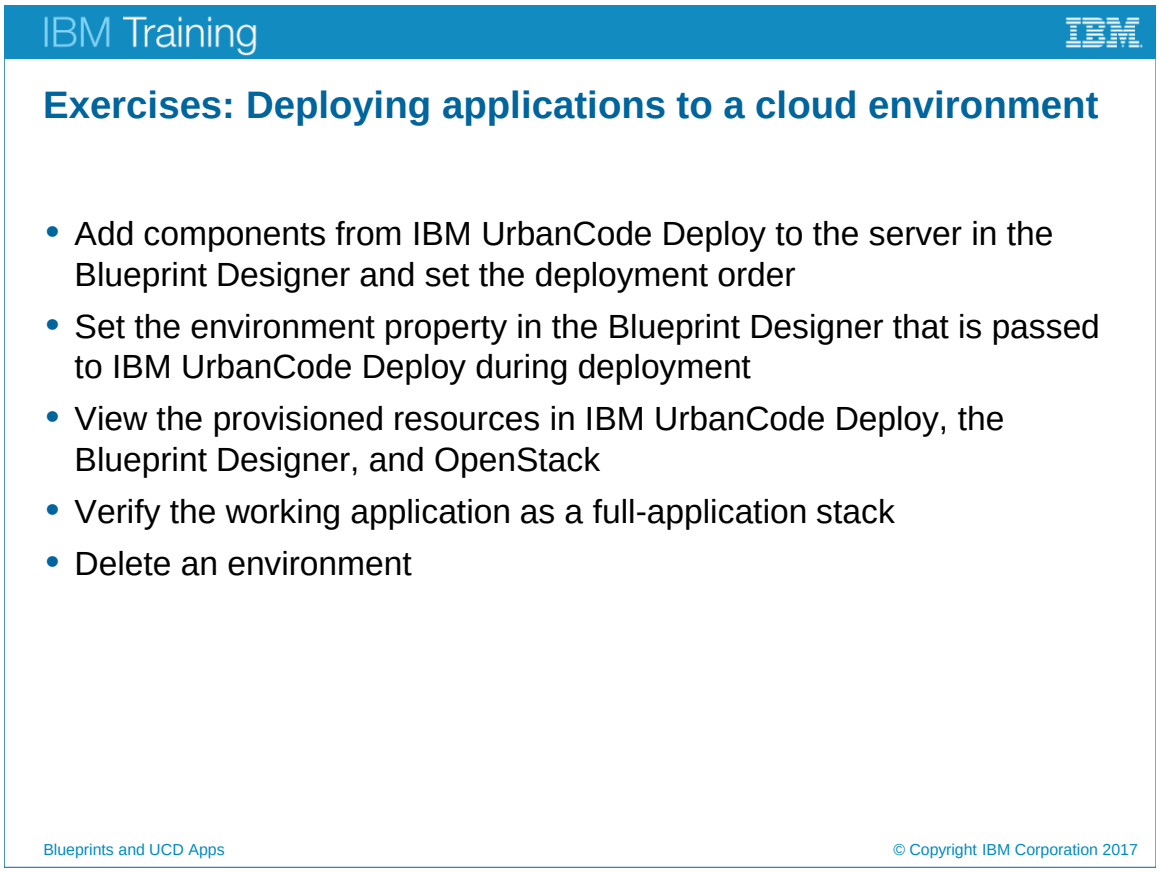
The instances on the cloud are deleted, and the resources in the resource tree are deleted.



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

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.

IBM Training

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.

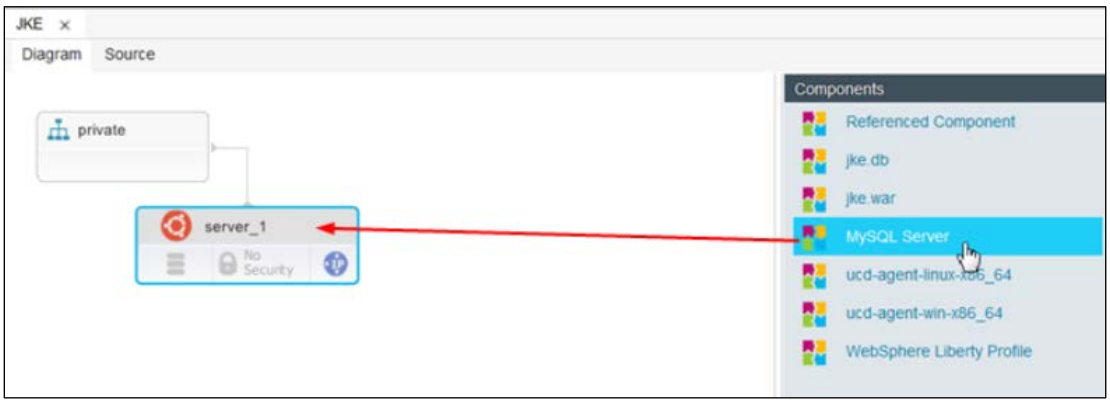
1. From the browser, switch to the **Blueprint Designer** bookmark.
2. Ensure that you are logged in as the **demo** user by typing `demo` in the **User Name** field and `labstack` in the **Password** field.
3. From the home page, click the existing **JKE** blueprint.

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.

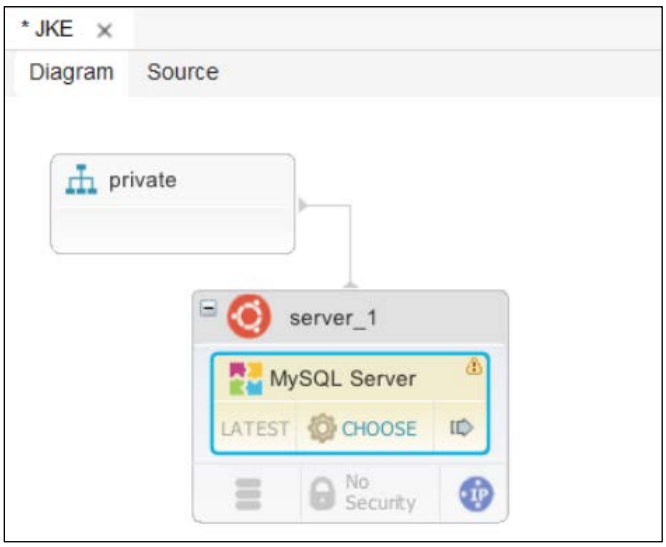
You can now instruct the blueprint design server to make use of the predefined components in IBM UrbanCode Deploy by configuring the server with MySQL and WebSphere Liberty.

1. 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.



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

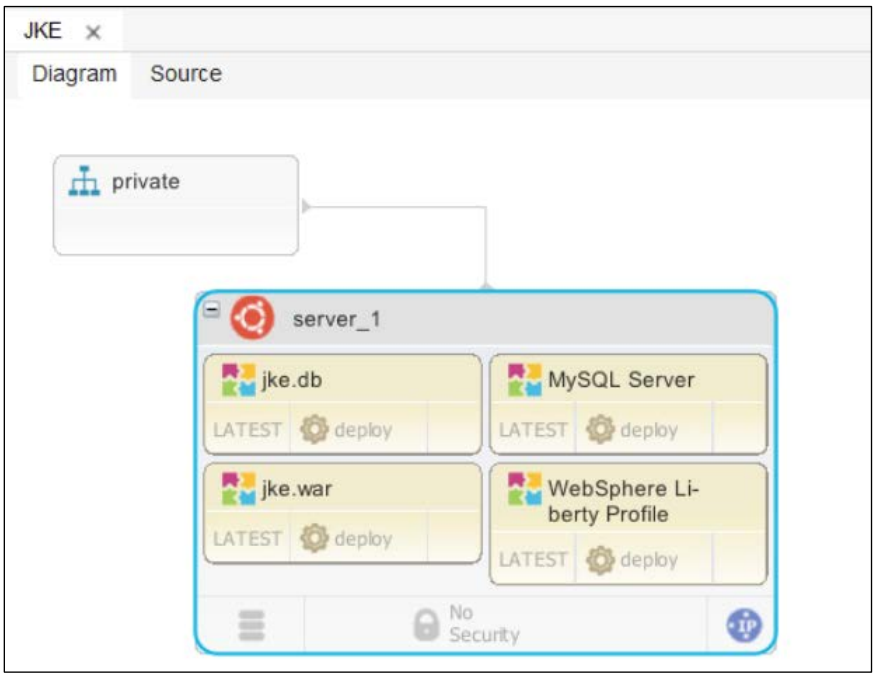
   The MySQL Server component is added to the image:



3. In the MySQL Server component, click **CHOOSE**, select **deploy** as the component process, and click **OK**.
4. 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:



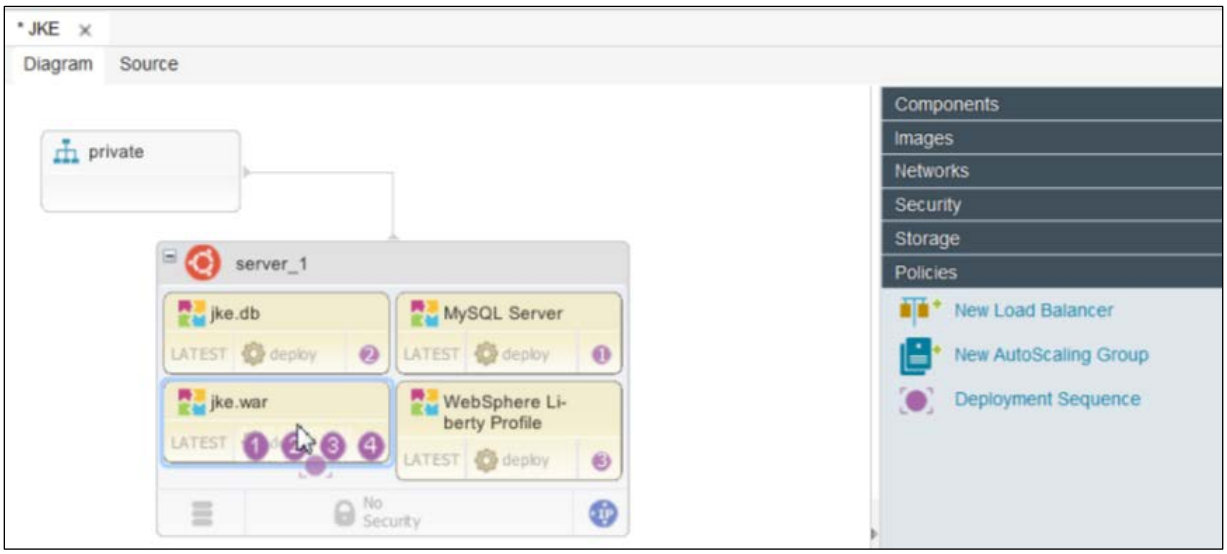
7. Save the blueprint.

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.

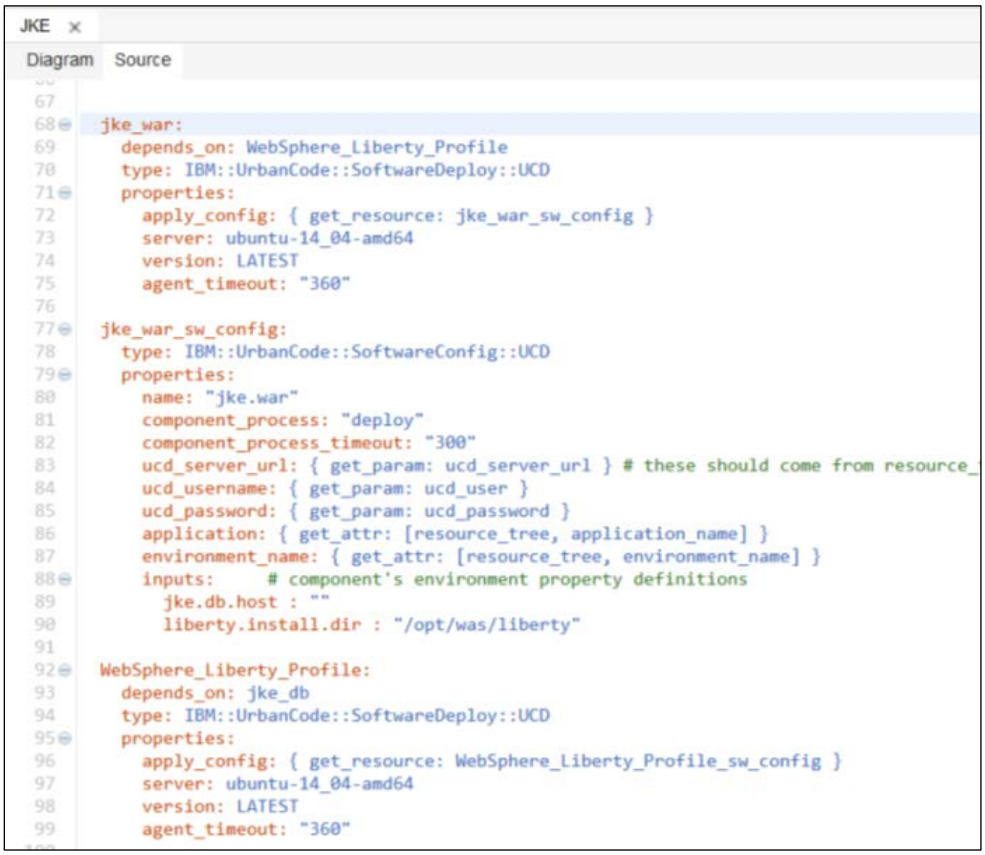
1. Open the **Policies** drawer, and drag the **Deployment Sequence** resource on to each of the components, on at a time, in the following order:
   a. MySQL_Server
   b. jke.db
   c. WebSphere Liberty Profile
   d. jke.war

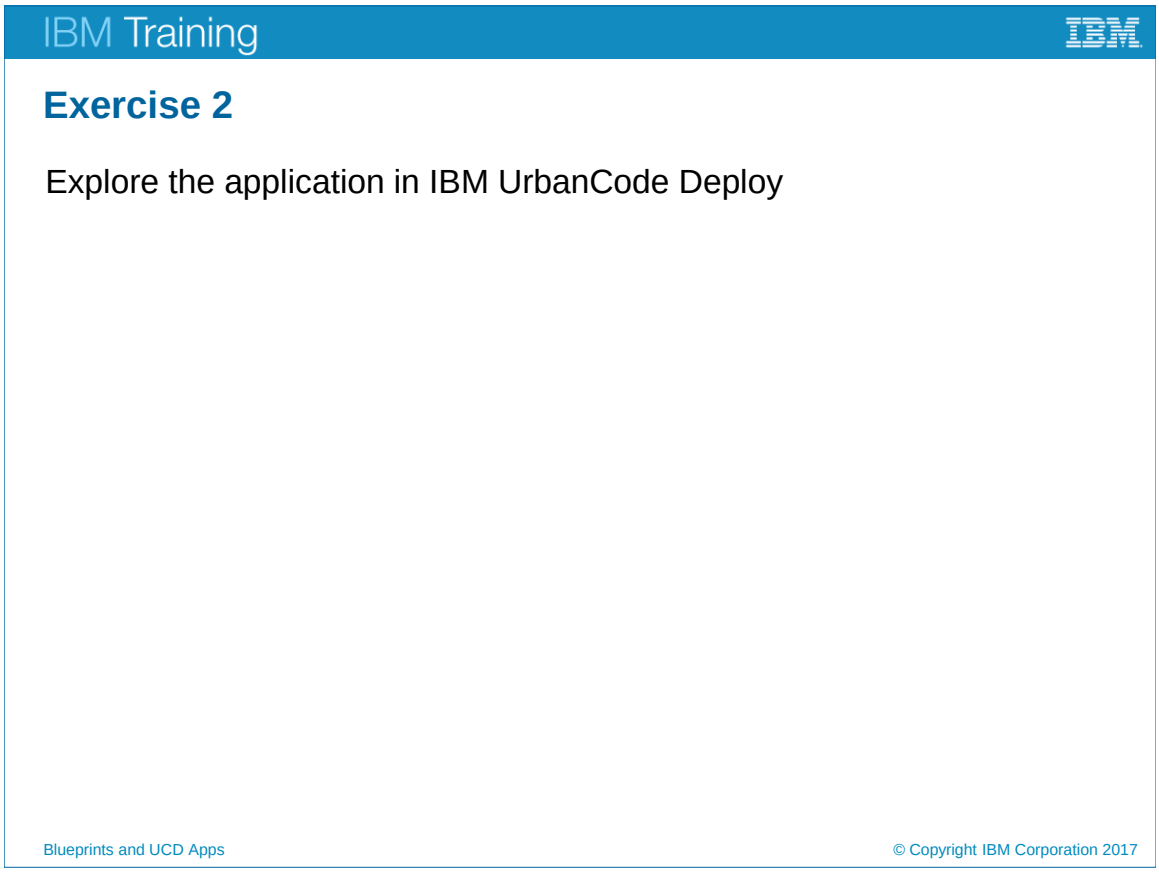


Note: If you set the deployment sequence incorrectly when dragging, you can click the **Deployment Sequence** icon of any of the components and change the sequence.

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.





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.

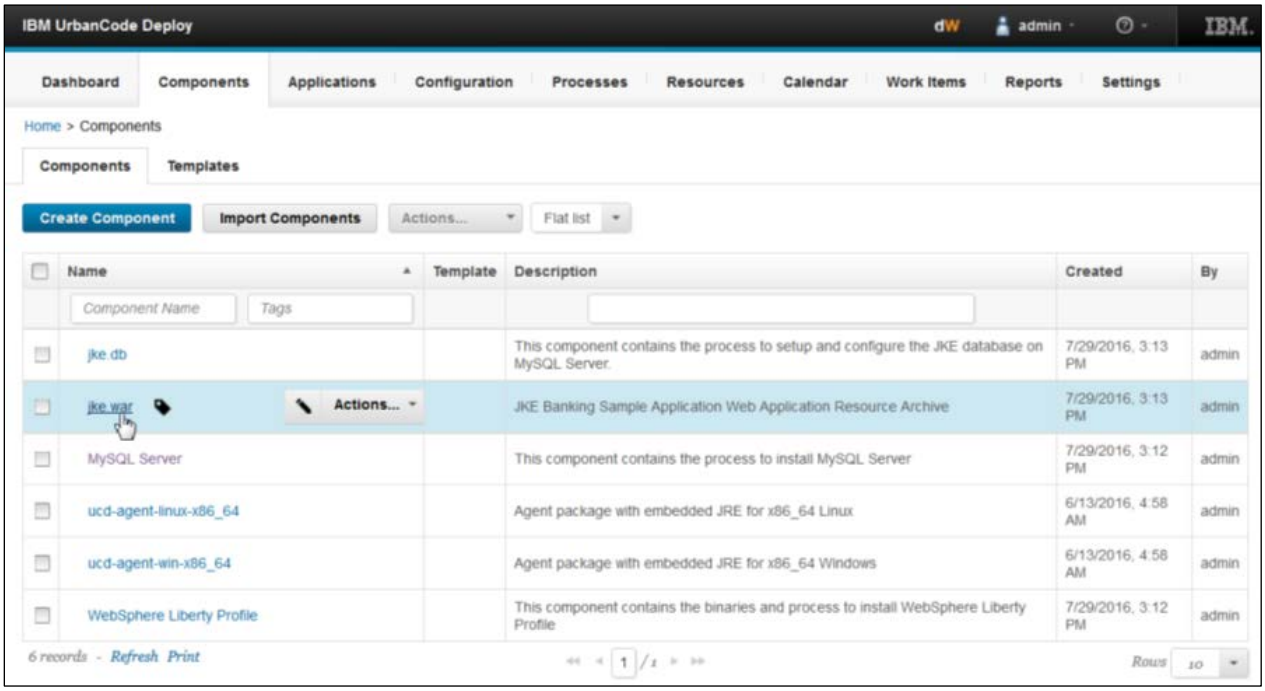
Exercise 2: Tasks and results

Task 1. Explore the component processes.

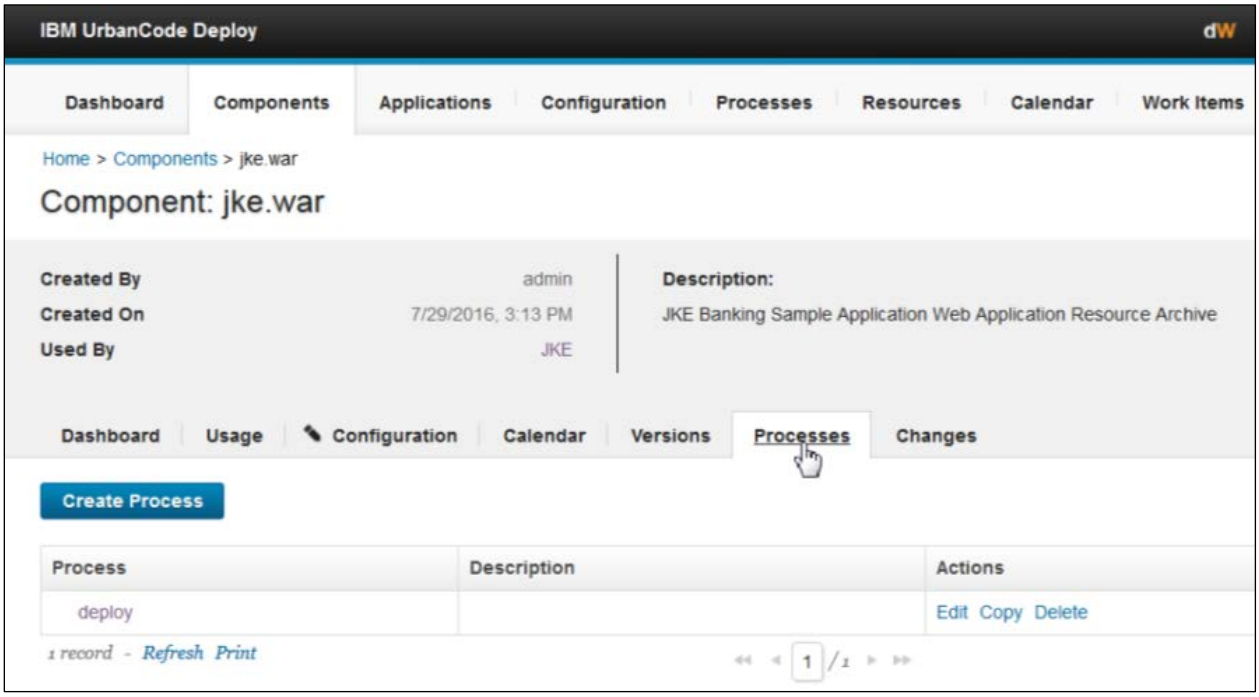
1. From the browser, switch to the **UrbanCode Deploy** tab.
2. From the navigation at the top of the page, click **Components**.

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



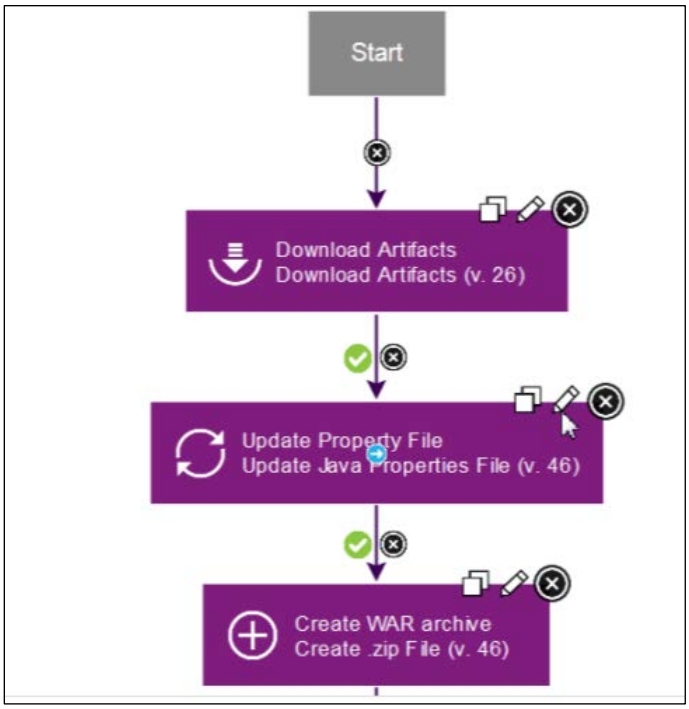
4. From the navigation for this component, click **Processes**.



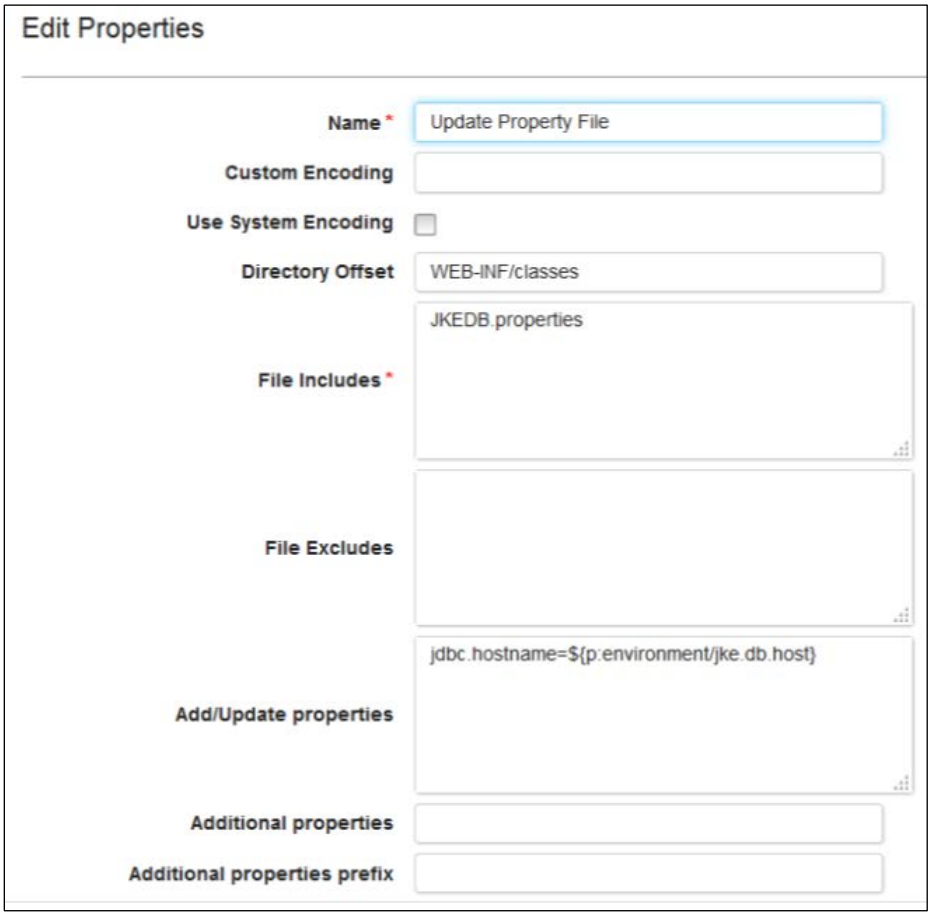
5. In the Process column, click **deploy** to open the process.

   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.

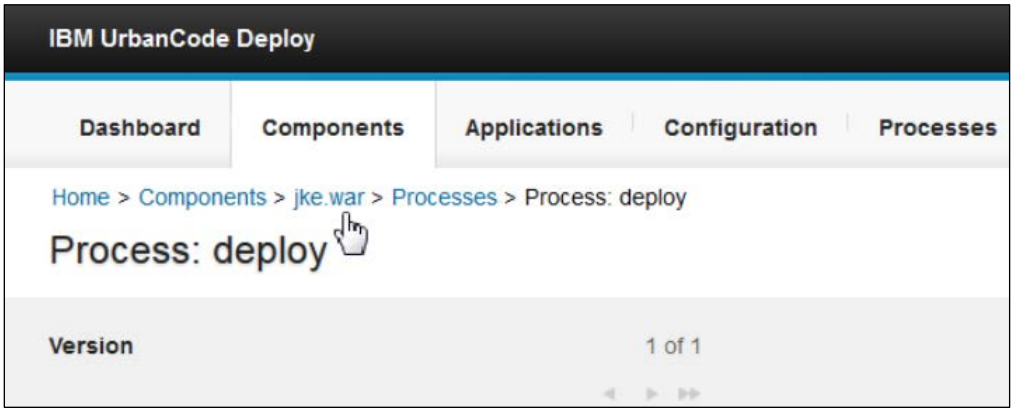


Observe that the field for **File includes** specifies one file, **JKEDB.properties**. In that file, this step searches for the string `jdbc.hostname=` and replaces the trailing value with the value of the IBM UrbanCode Deploy environment property called `jke.db.host`. In IBM UrbanCode Deploy, each environment 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.



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

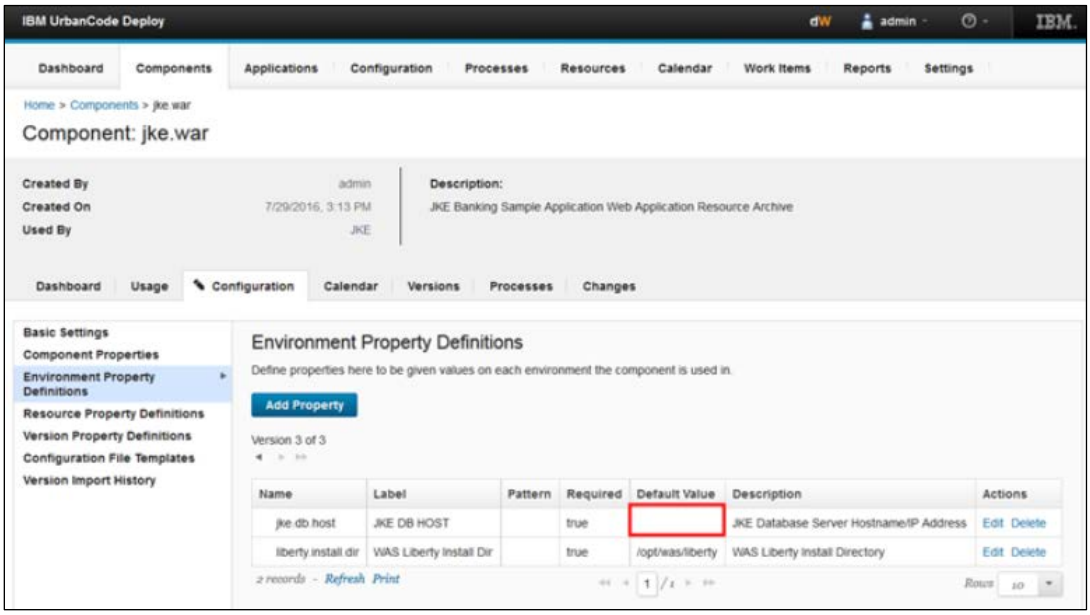
8. At the top of the page, click the **jke.war** link.



9. 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.

10. 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.



IBM Training

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

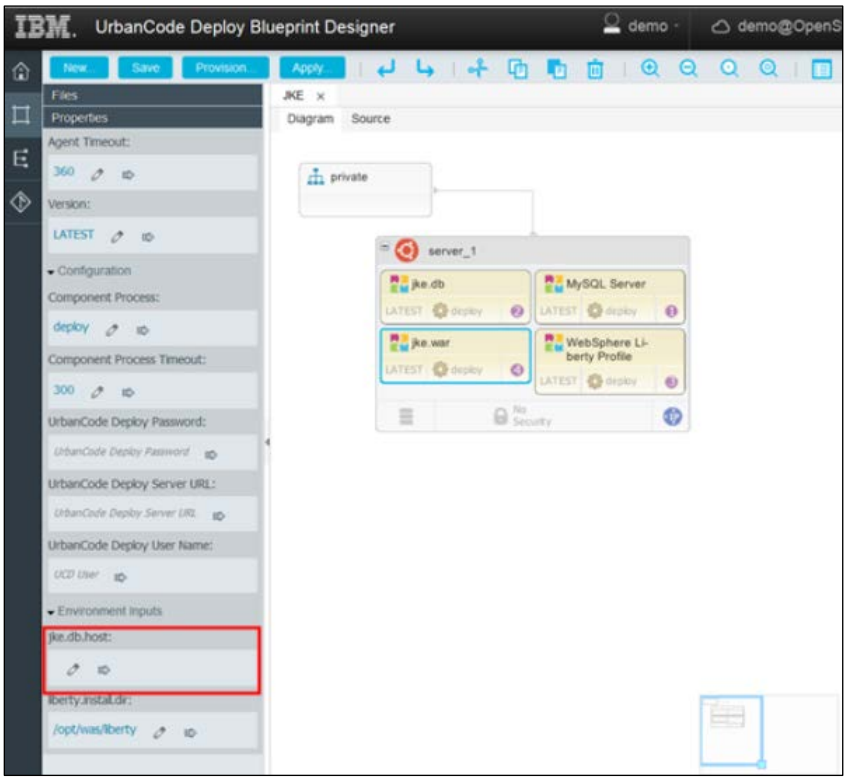
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 3: Tasks and results

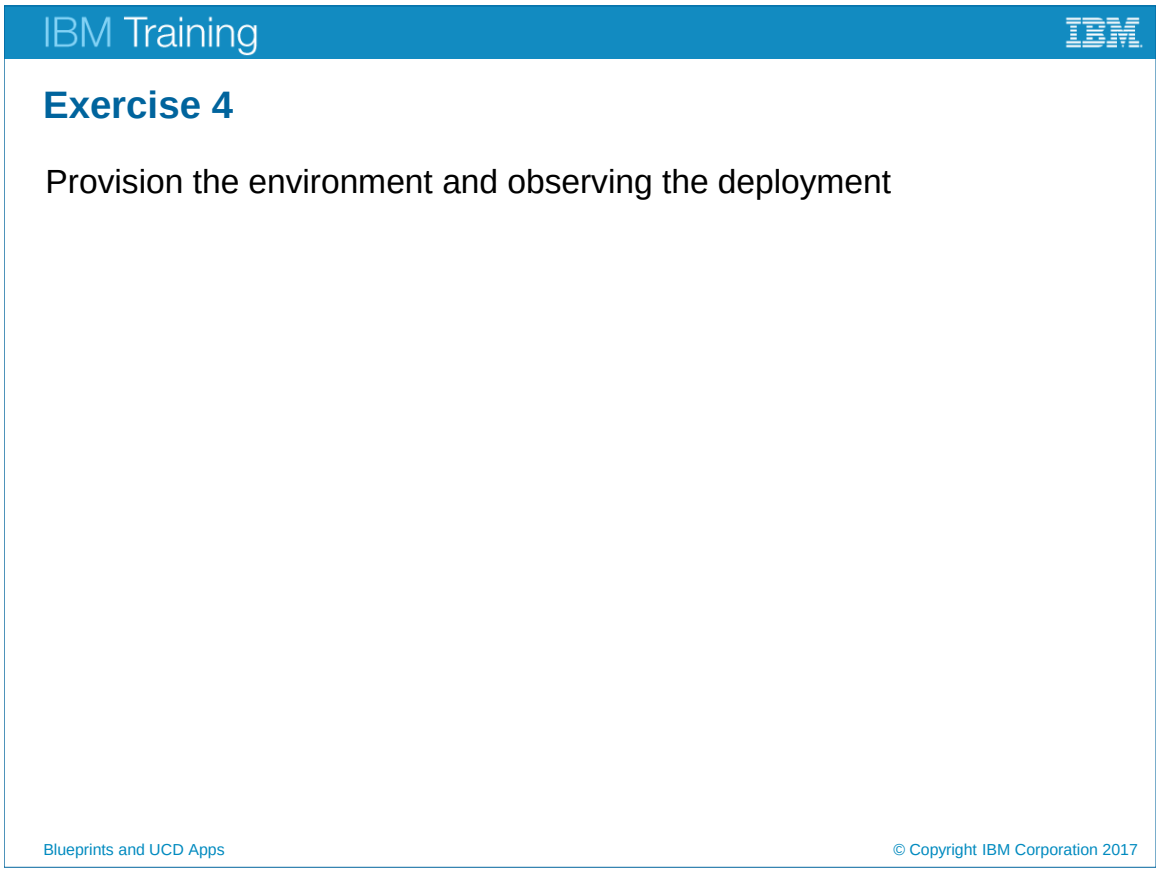
Task 1. Update the environment property.

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

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:

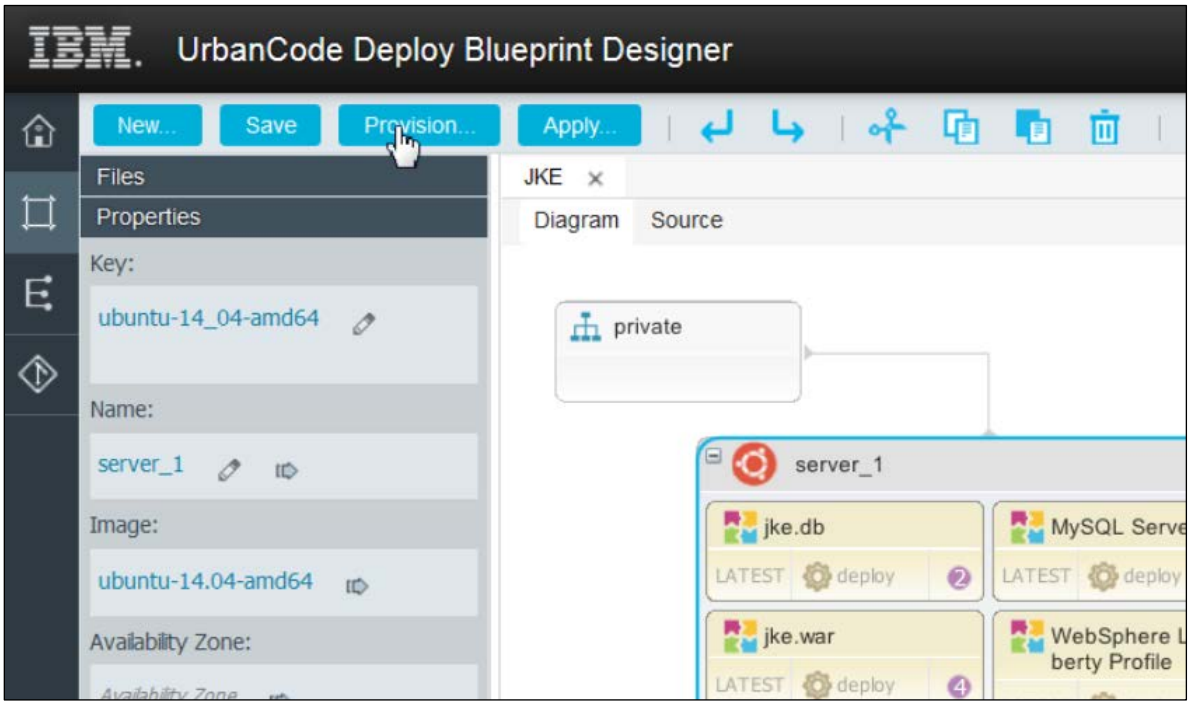
- Provision the blueprint.
- Observe the OpenStack provision.
- Observe the provision and deployment in IBM UrbanCode Deploy.

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

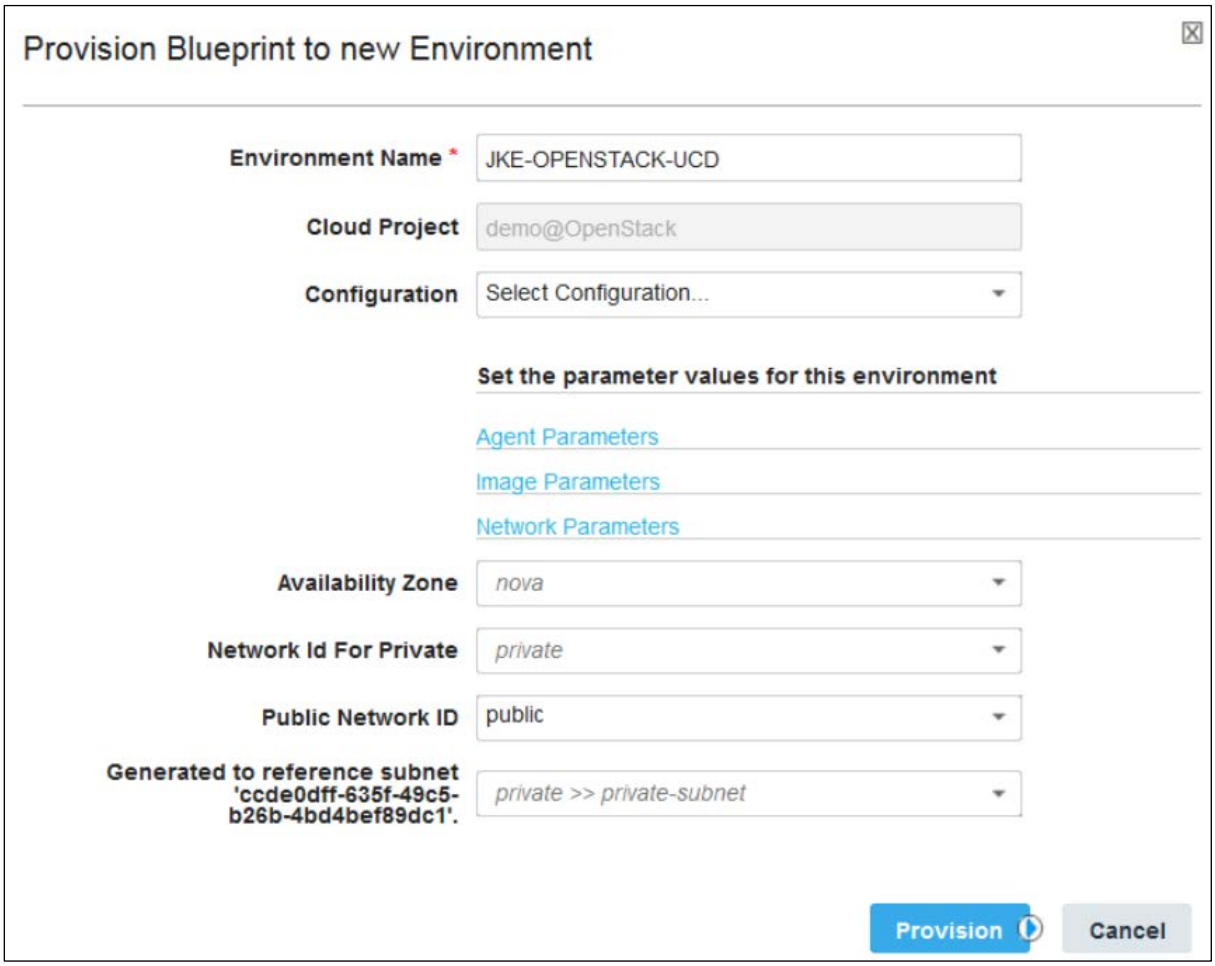
Exercise 4:
Tasks and results

Task 1. Provision the blueprint.

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



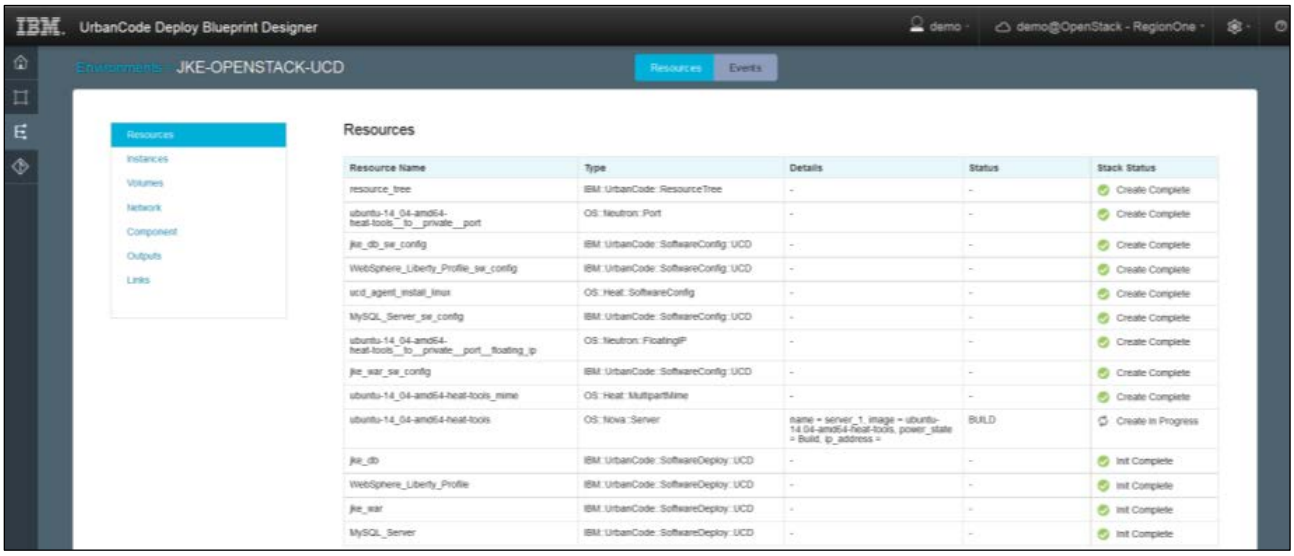
2. In the “Provision Blueprint to new Environment” window, do the following steps:
   - In the **Environment Name** field, type `JKE-OPENSTACK-UCD`.
   - Click **Image Parameters**.
     - In the **Flavor** field, select **m1.medium**.
     - In the **Key name** field, select **demo-key**.
   - Click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



3. Click **Provision**.

4. Click the **Environments** tab.

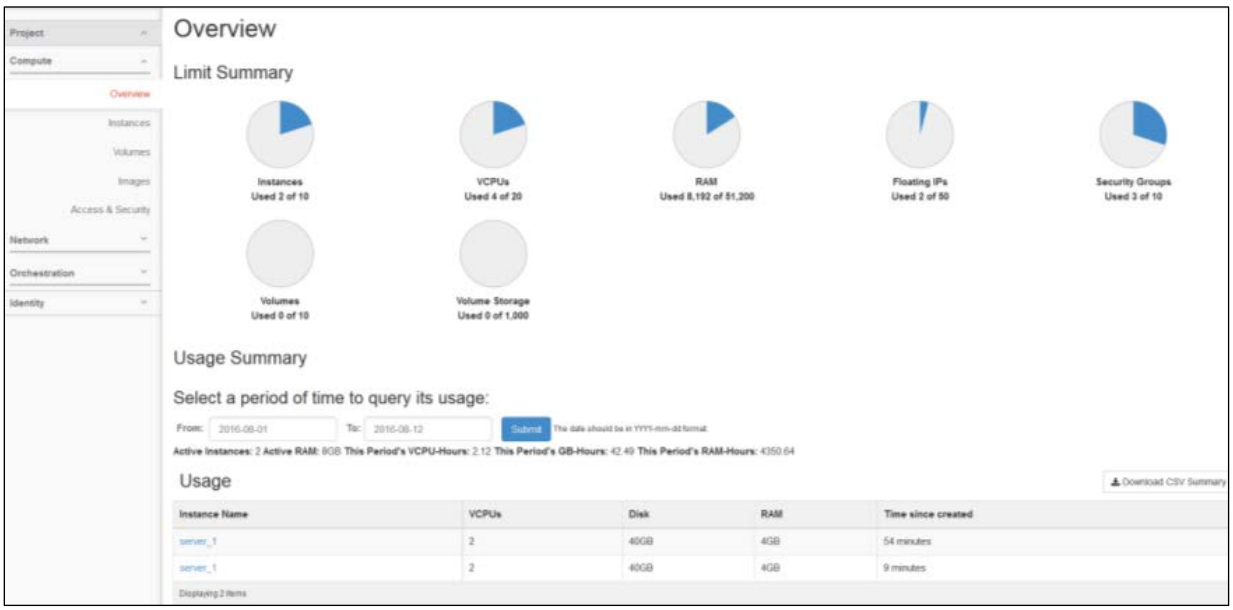
5. Expand the list of resources by clicking the twistie next to **JKE-OPENSTACK_UCD**. Observe the progress of the deployment.
6. In the Environment column, click **JKE-OPENSTACK-UCD** to see the resource details.



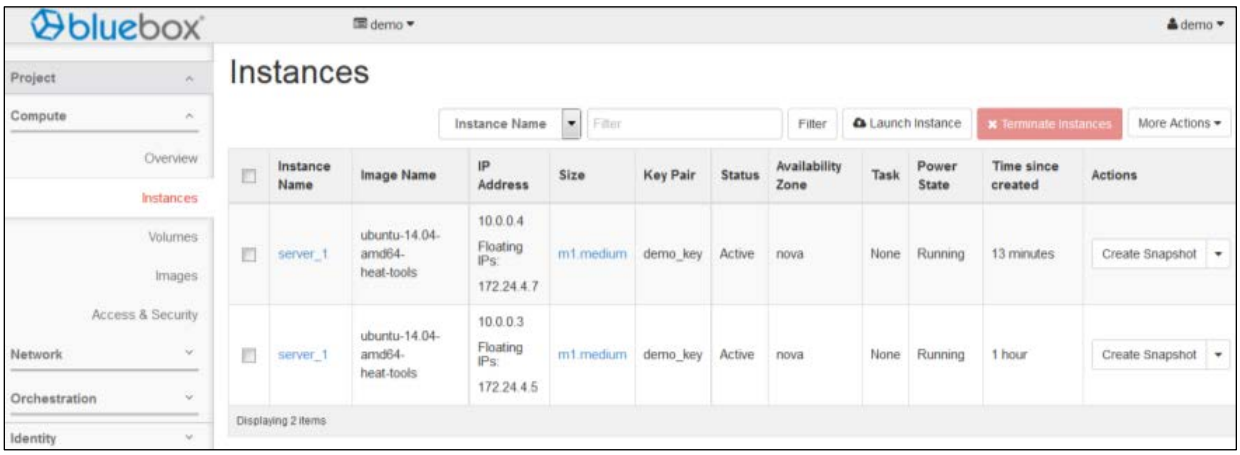
7. Click **Instances** to view the floating IP address that is assigned to the image.

Task 2. Observe the OpenStack provision.

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

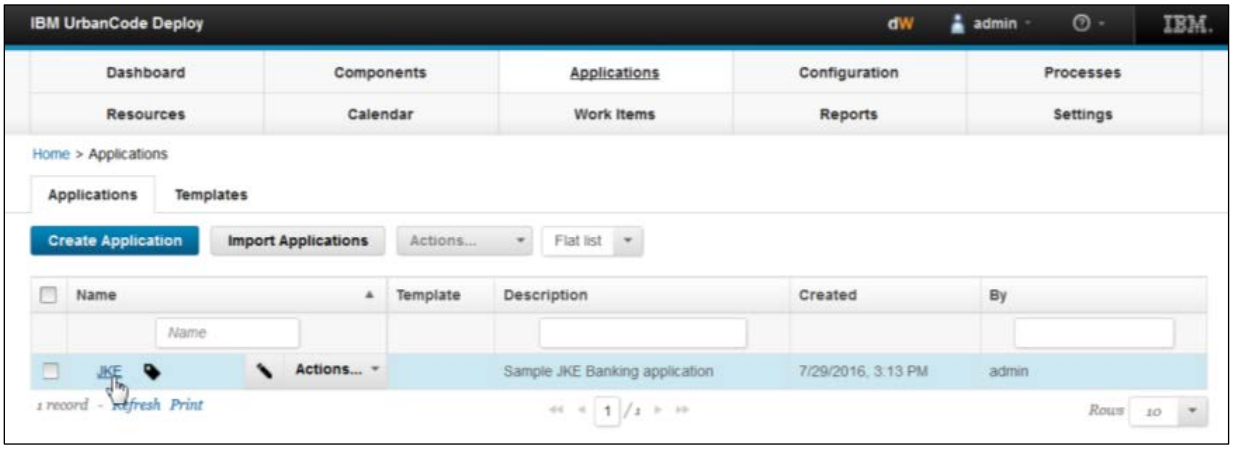


3. Click **Instances** to see the image coming up and to see the IP addresses assigned to the system.



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

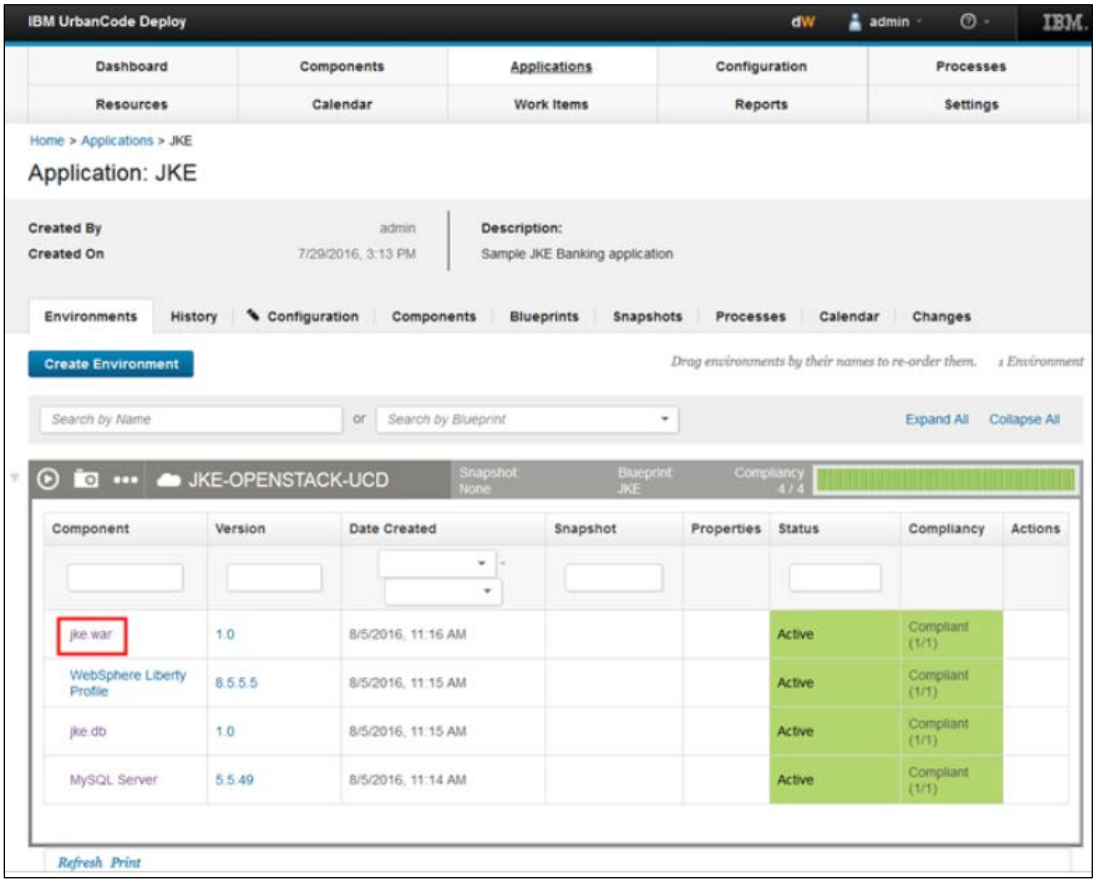
1. In the browser, switch to the **IBM UrbanCode Deploy** tab to observe the environment.
2. Click **Applications**, and click **JKE**.



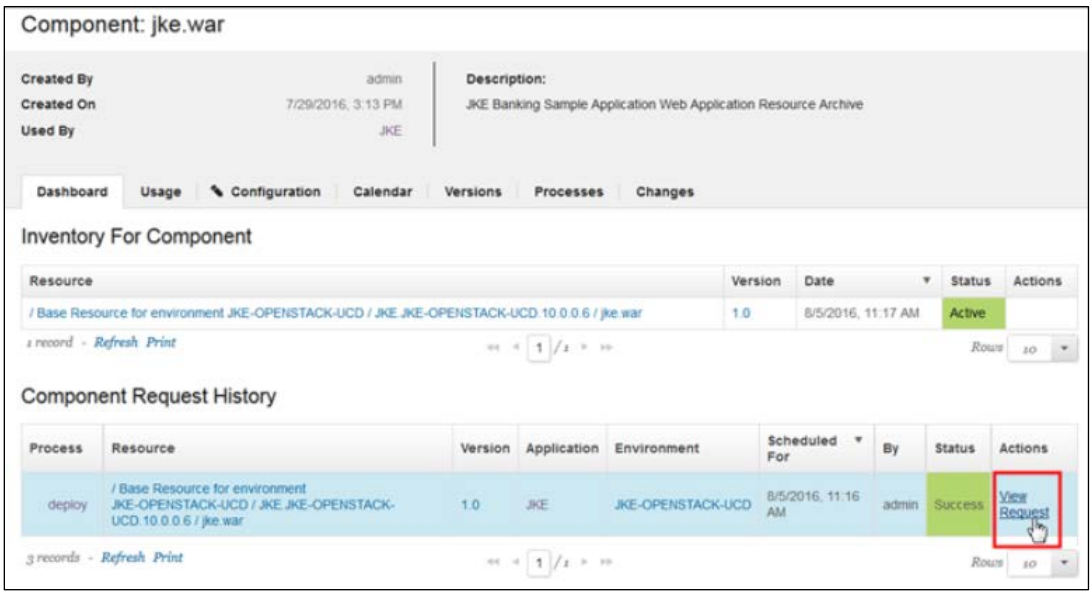
3. Click the twistie to the left of the created environment, **JKE-OPENSTACK-UCD**. Occasionally refresh the page until the **jke.war** component is displayed.
4. Observe the components as they progress through the deployment steps.

   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.

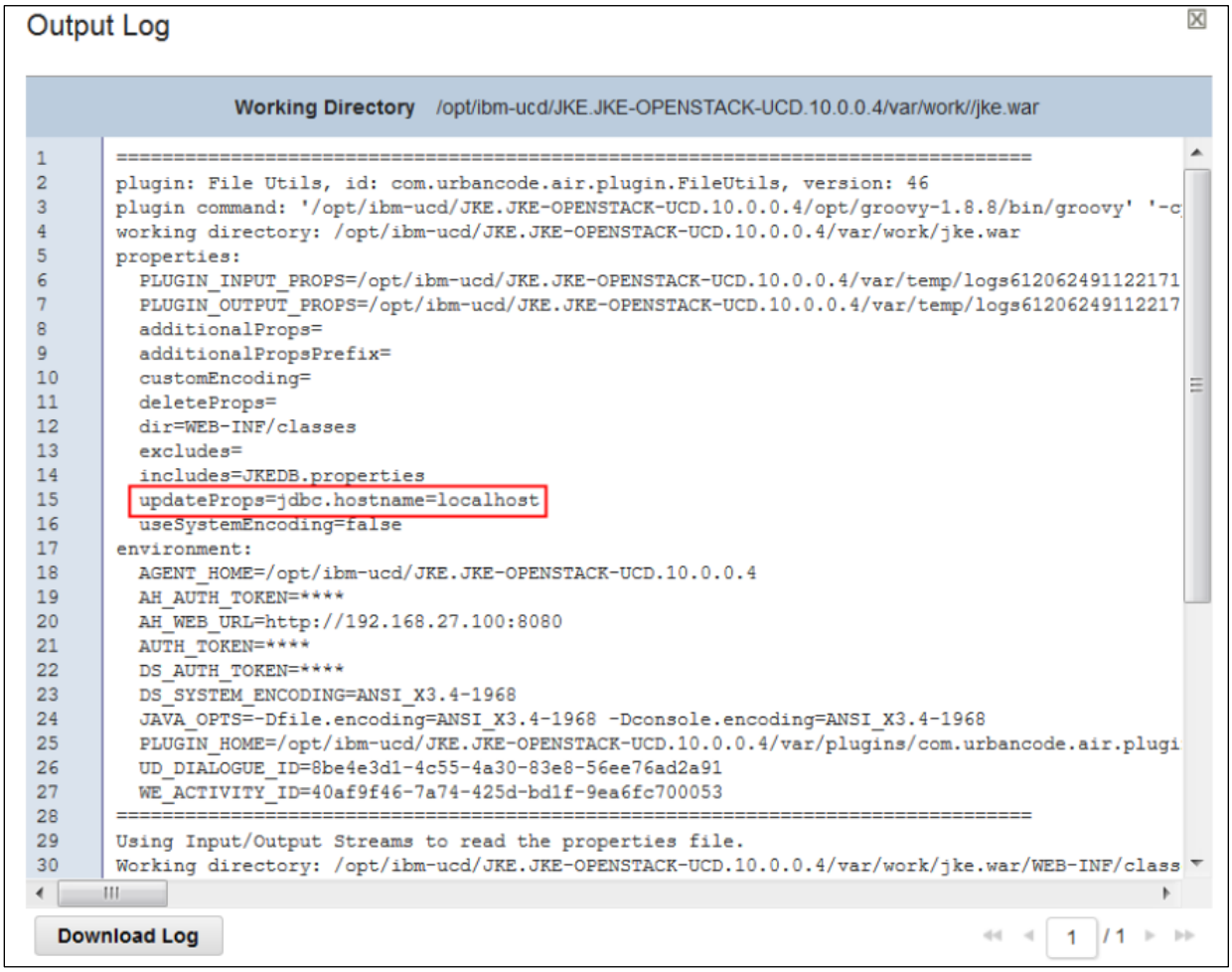
5. In the Component column, click **jke.war** after it is displayed.

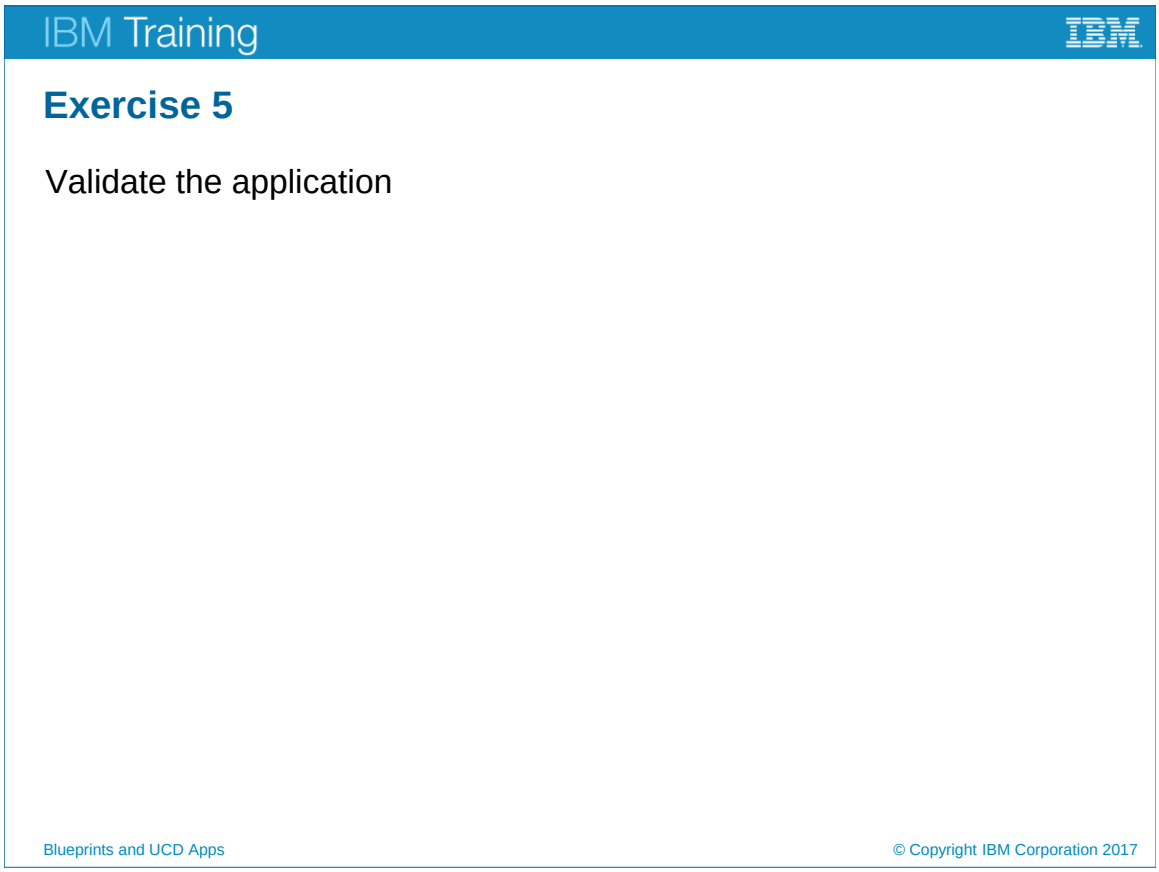


6. In the **Actions** column, click **View Request** and observer the steps as they complete.



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.





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.

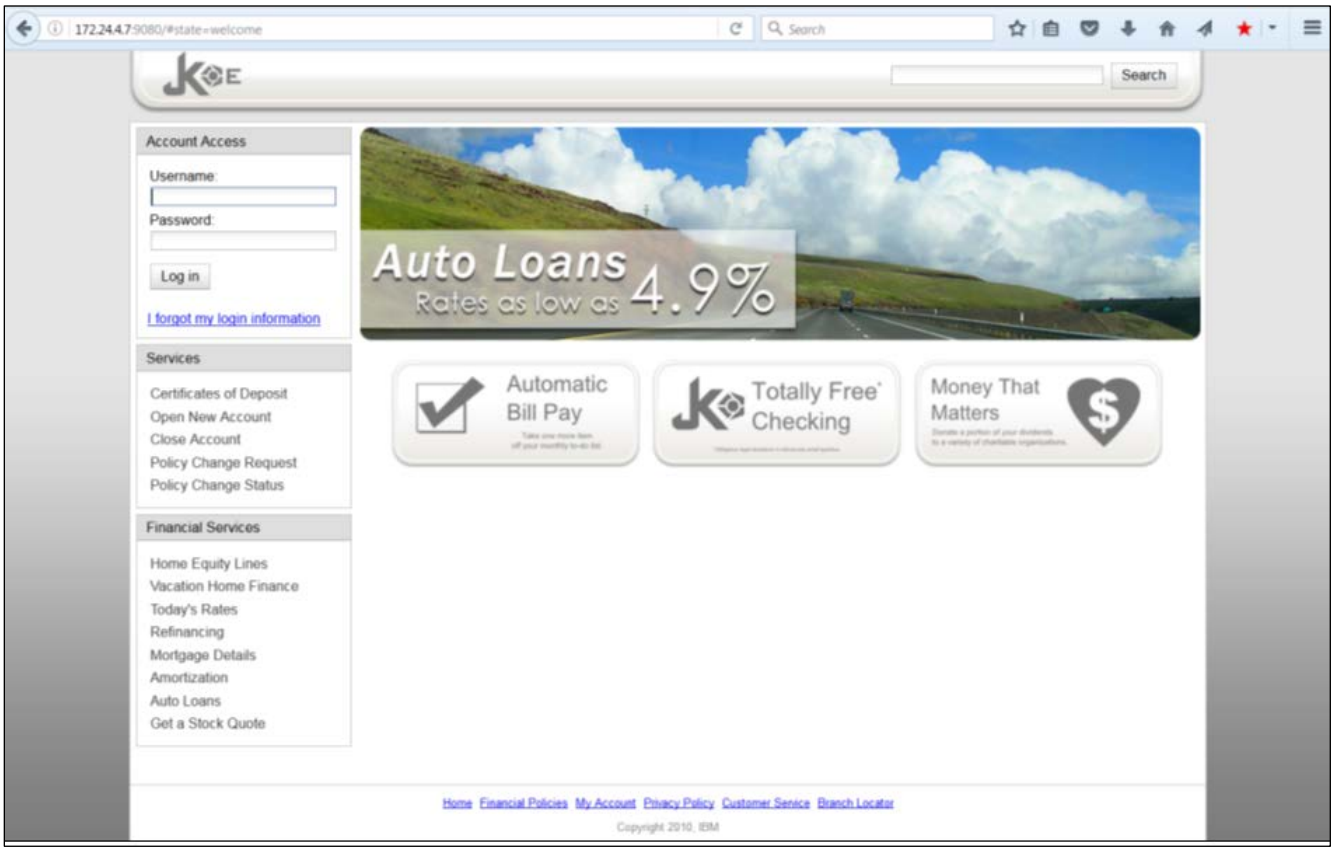
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 5: Tasks and results

Task 1. Validate the application.

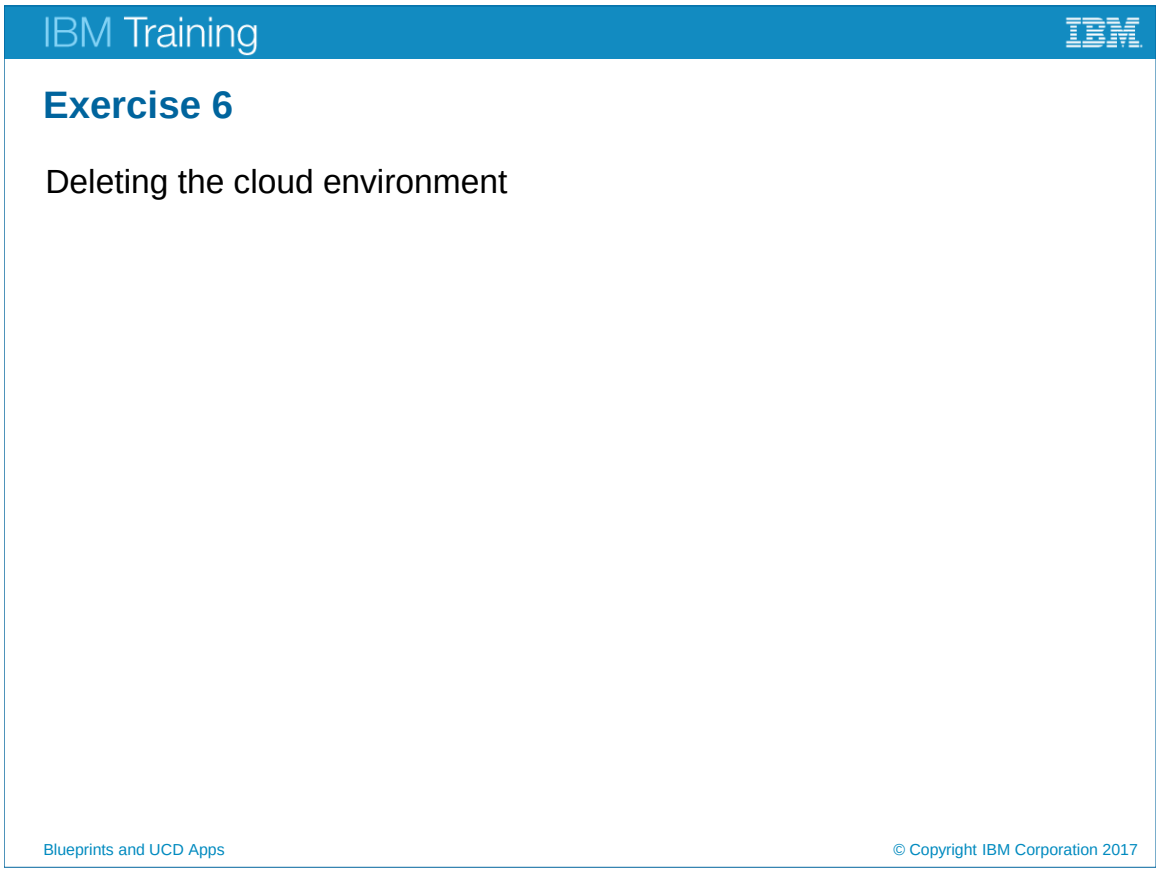
1. Open a new tab in the browser.
2. Type the floating IP address that OpenStack assigned to your environment (172.24.4.x) and include port 9080. In this lab, the URL is http://172.24.4.7:9080.

   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: 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:

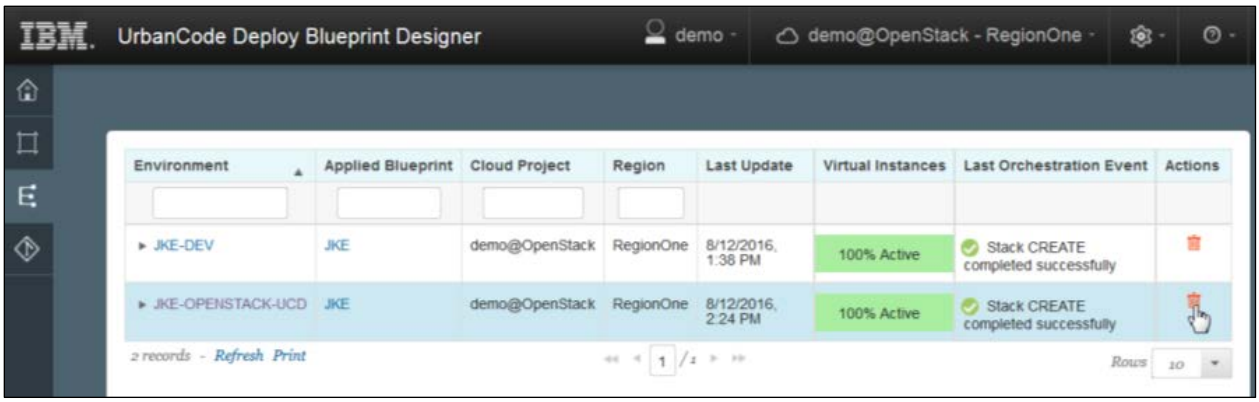
- Delete the cloud environment.

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

Exercise 6: Tasks and results

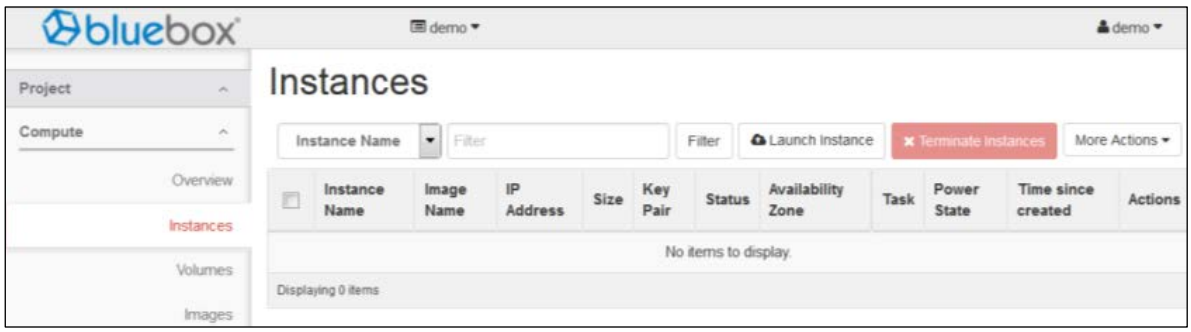
Task 1. Delete the cloud environment.

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.



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.



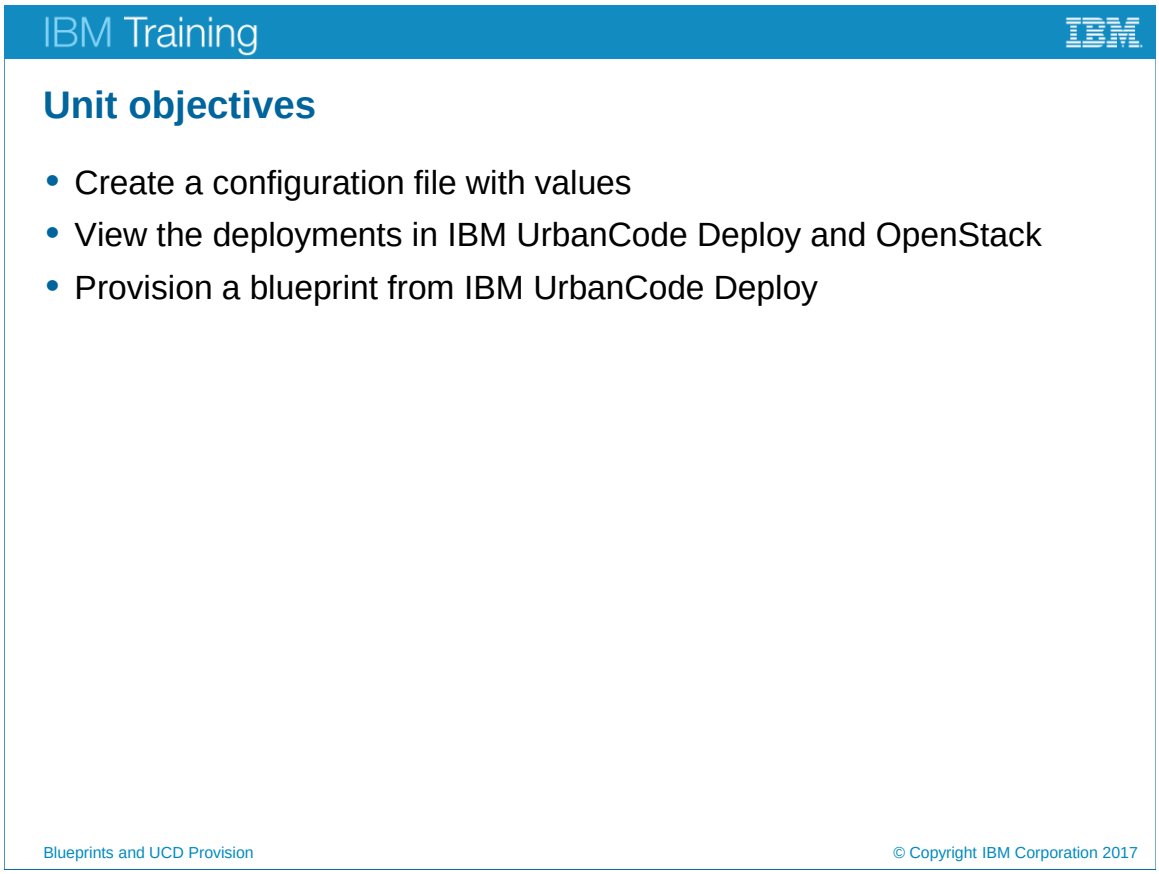
5. From the browser, click the tab with the JKE application, and refresh the browser to verify that the environment is removed.

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.

IBM Training

Blueprints and UCD Provision

IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1

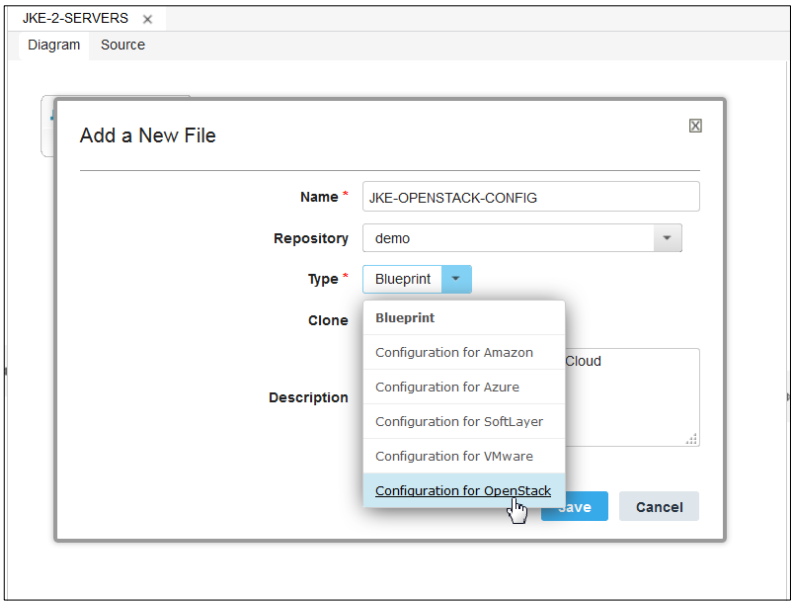


Unit objectives

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

Provisioning configuration files on different cloud systems



Blueprints and UCD Provision

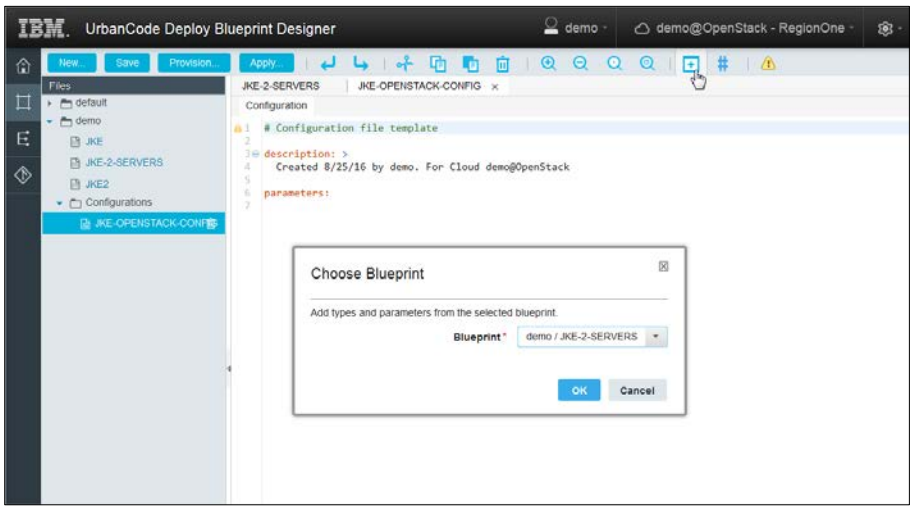
Provisioning configuration files on different cloud systems

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.

IBM Training

Importing properties from existing blueprints



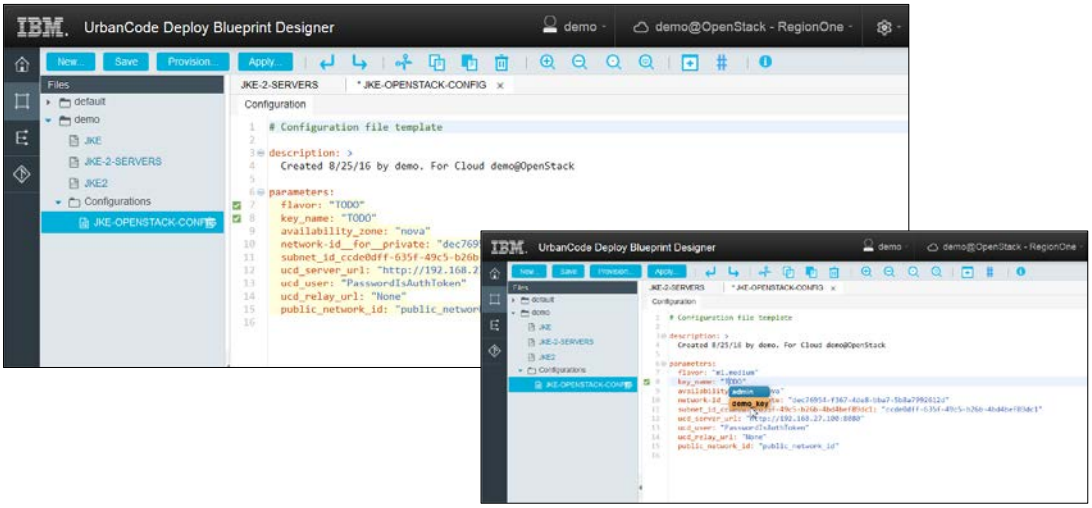
Blueprints and UCD Provision

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

Configuration files list parameters to provision a blueprint



Blueprints and UCD Provision

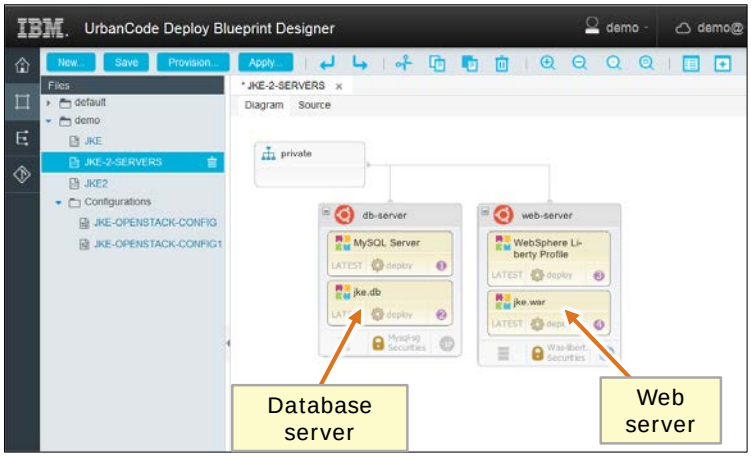
Configuration files list parameters to provision blueprint

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.

IBM Training

Separate functional areas for the web and database servers

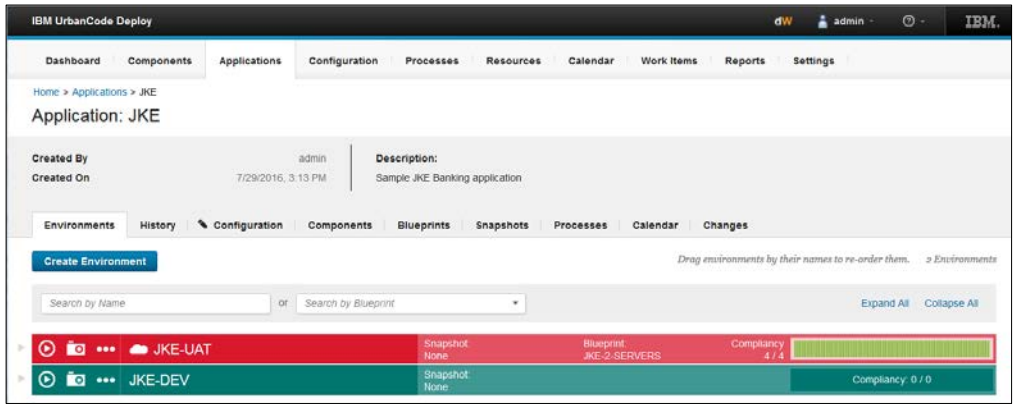


Separate functional areas for the web and database servers

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

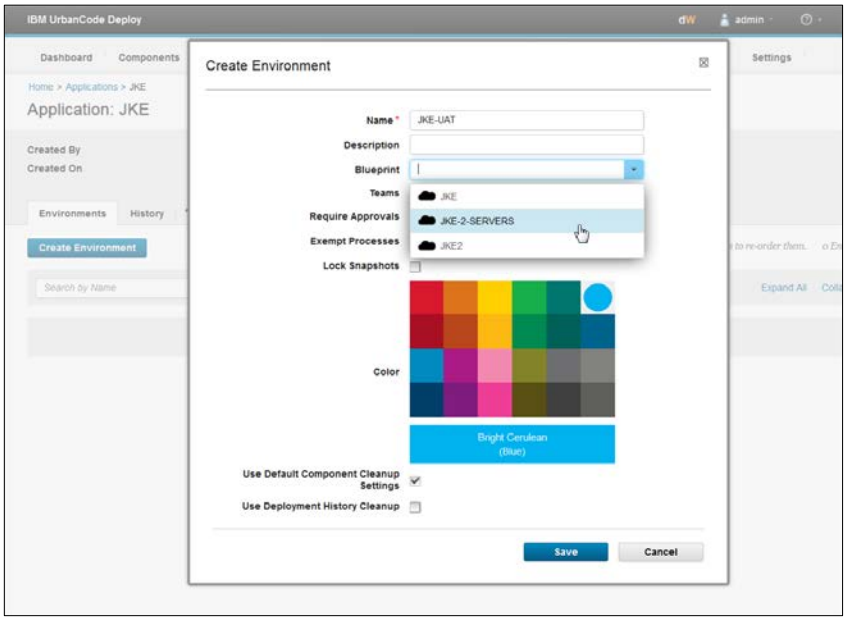


Environments created from blueprints

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

IBM Training

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

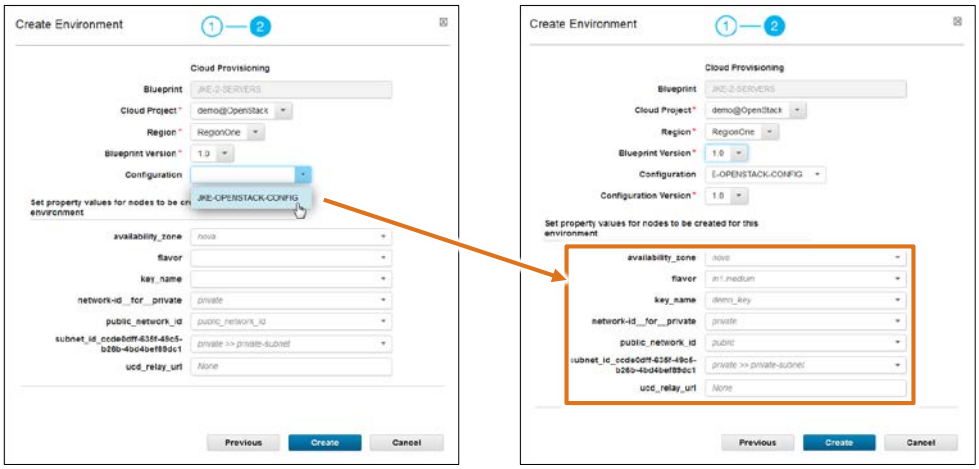


Blueprints created in the Blueprint Designer are available in IBM UrbanCode Deploy

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.

IBM Training

Predefined properties and values to use



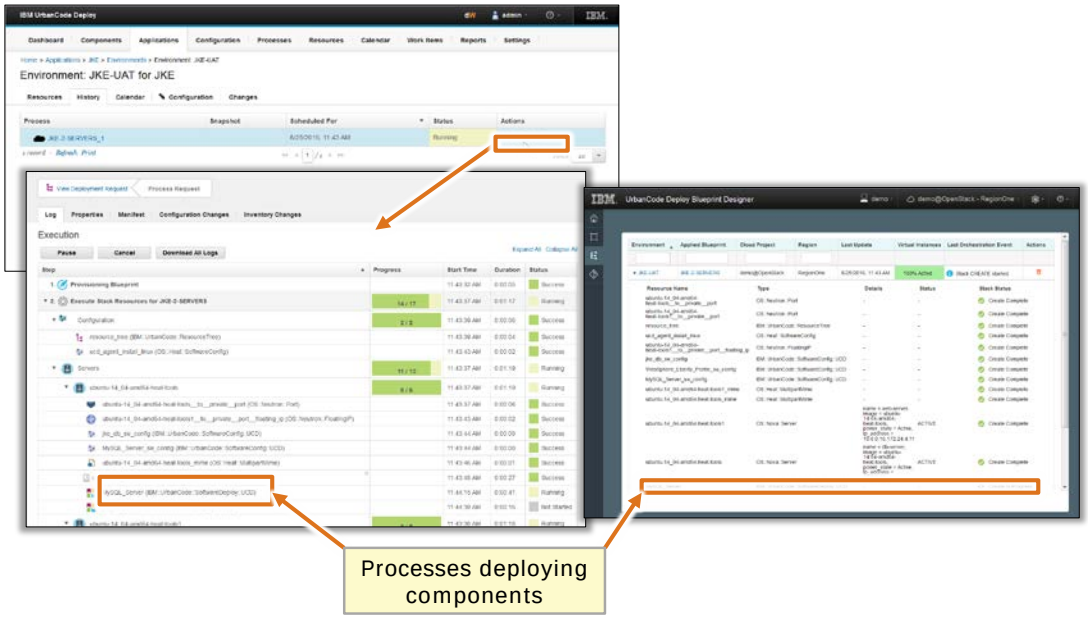
Blueprints and UCD Provision

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.

IBM Training

Creating an environment

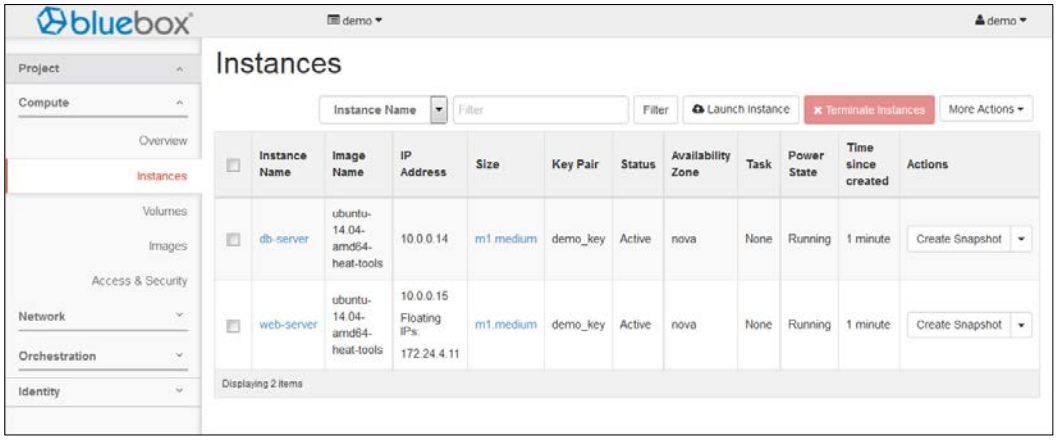


Creating an environment

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.

IBM Training

Instances of servers on OpenStack



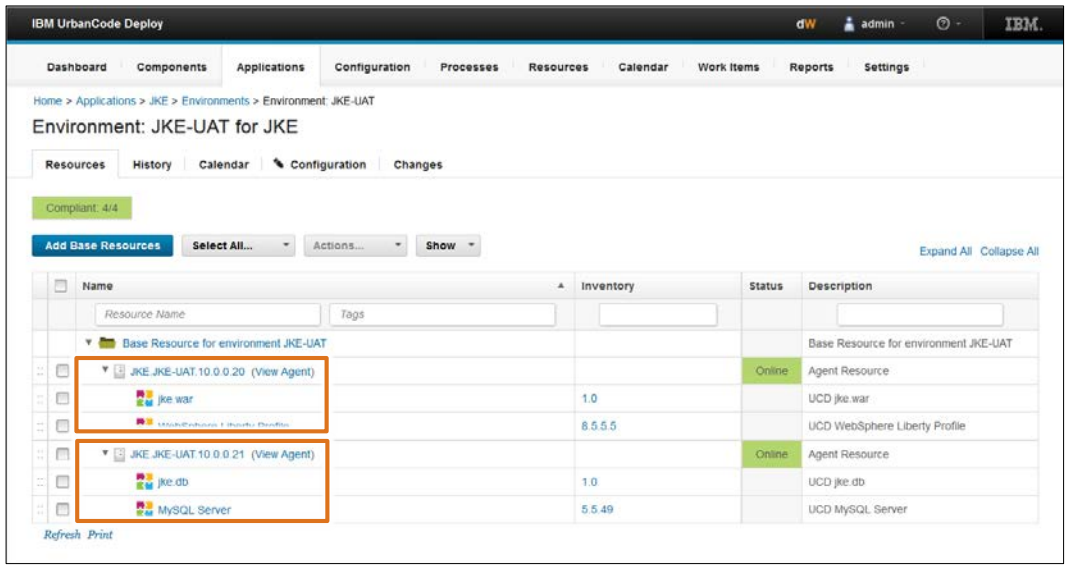
Blueprints and UCD Provision

Instances of servers on OpenStack

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

IBM Training

Agent processes are started separately for each resource



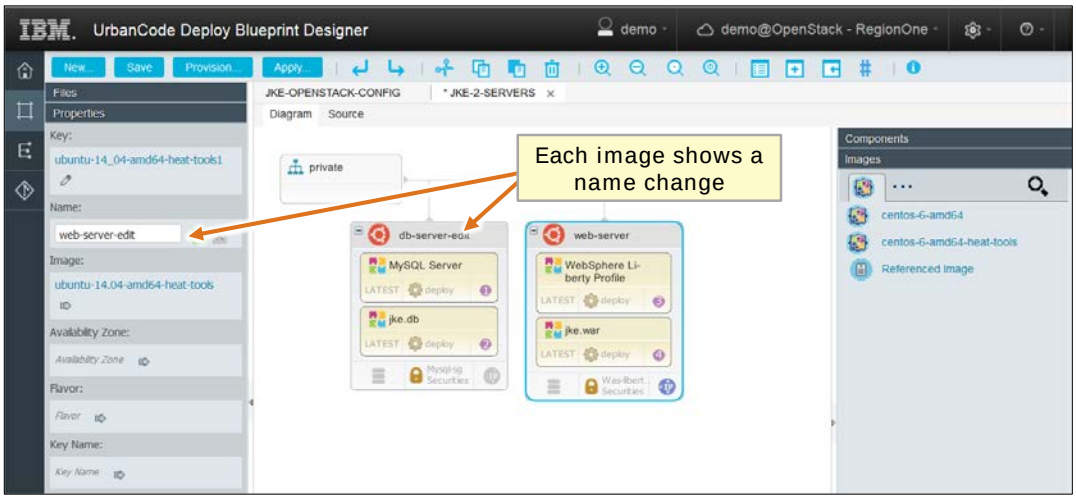
Blueprints and UCD Provision

Agent processes are started separately for each resource

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

IBM Training

Updating environment in Blueprint Designer



Blueprints and UCD Provision

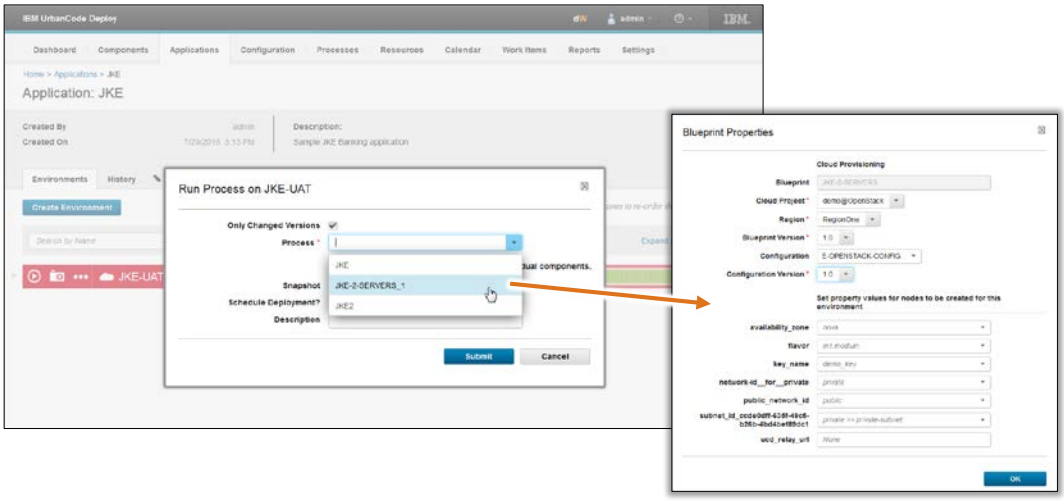
Updating environment in Blueprint Designer

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.

IBM Training

Blueprints can update a running environment



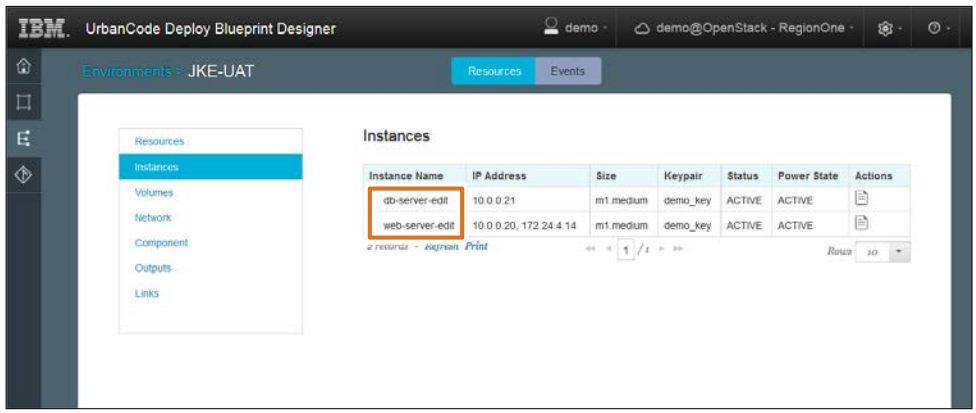
Blueprints and UCD Provision

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.

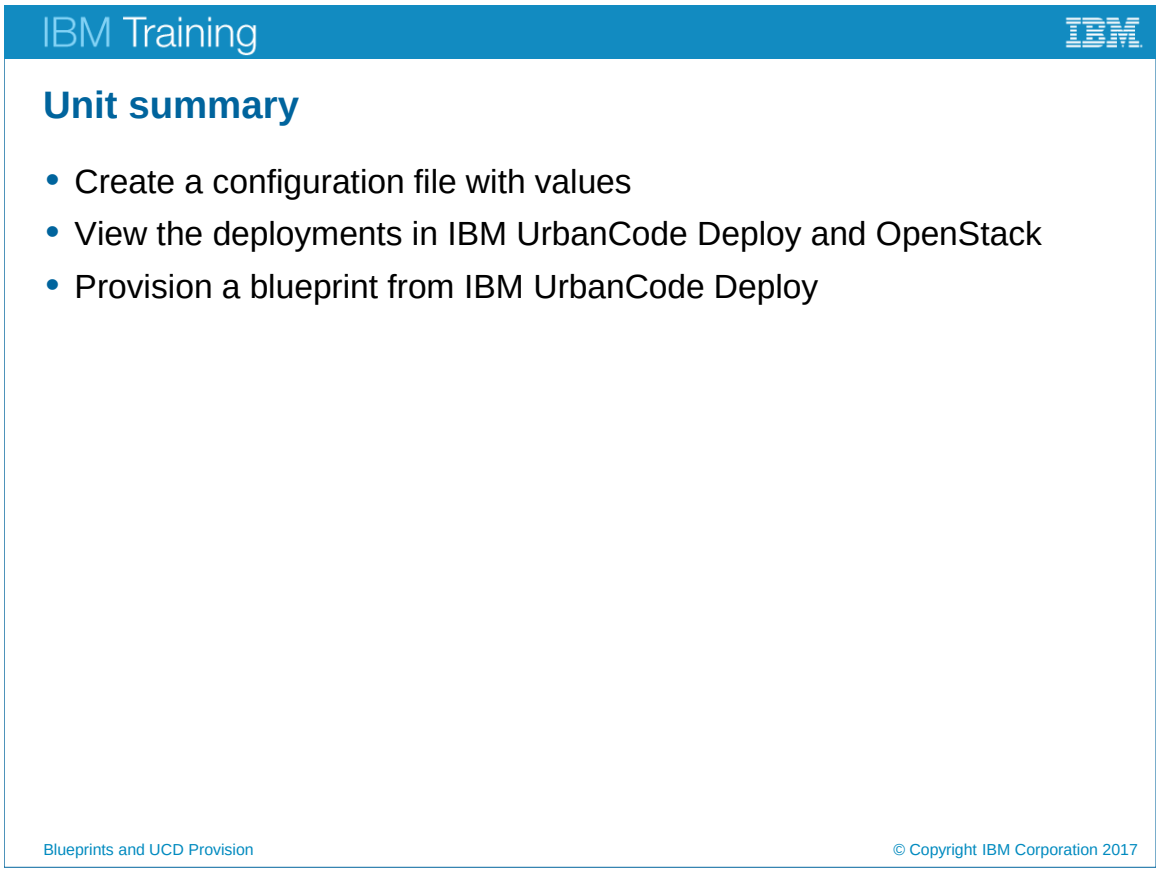
IBM Training

Instance names are updated in the Blueprint Designer



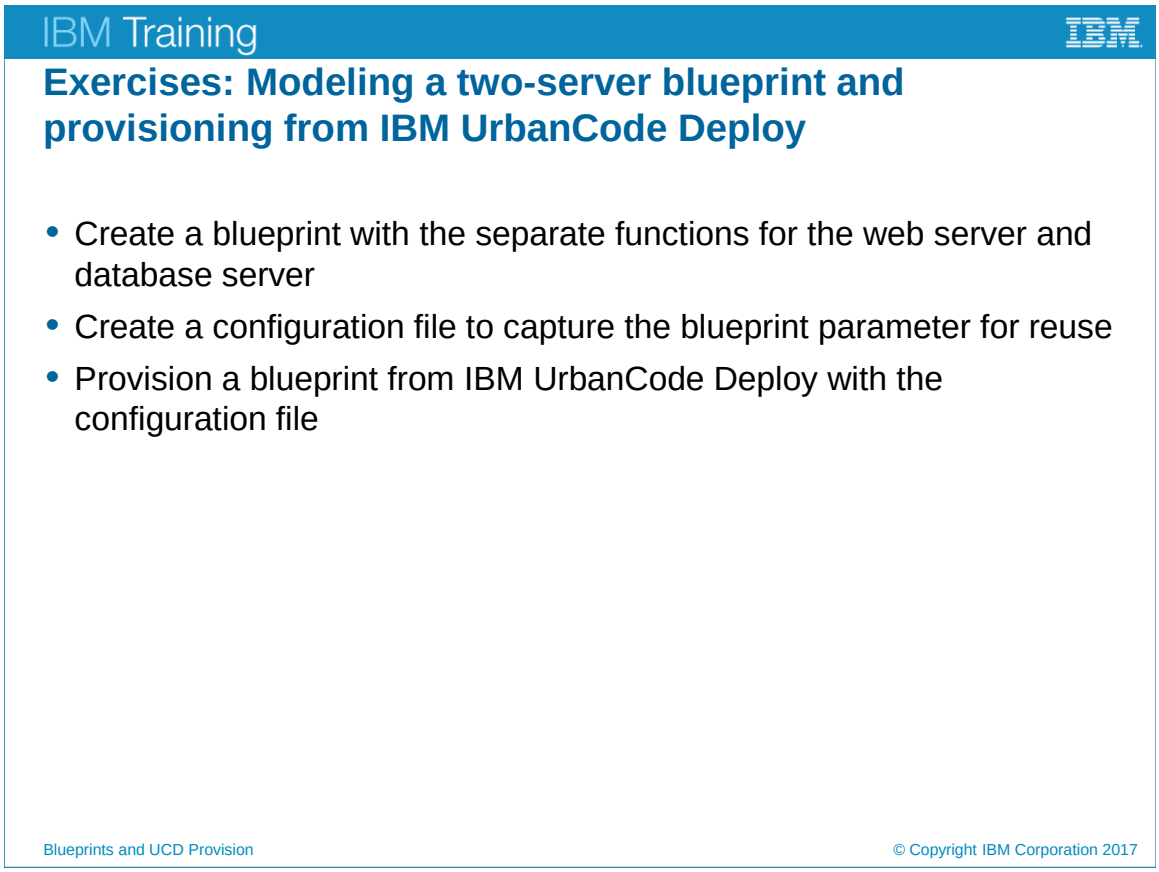
Instances name are updated in the Blueprint Designer

The Blueprint Designer shows the updates to the names.



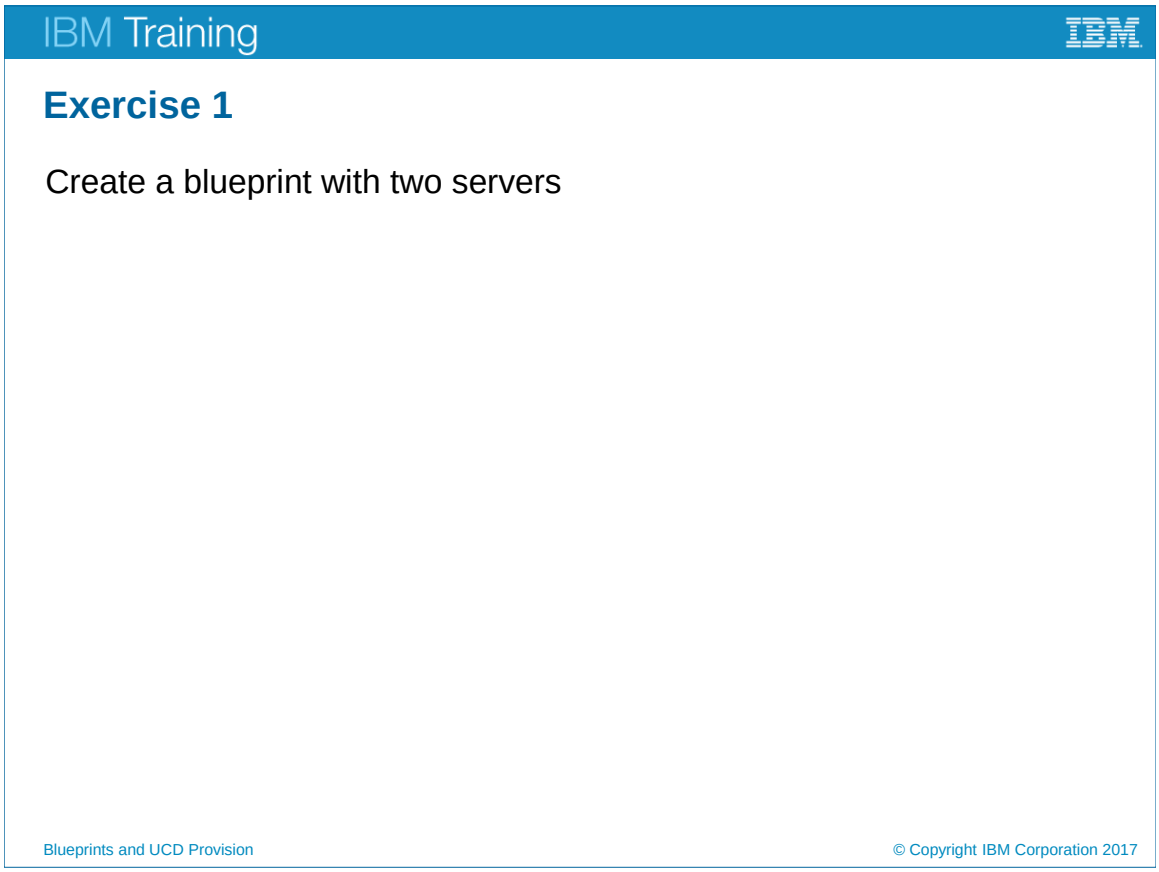
Unit summary

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 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

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.

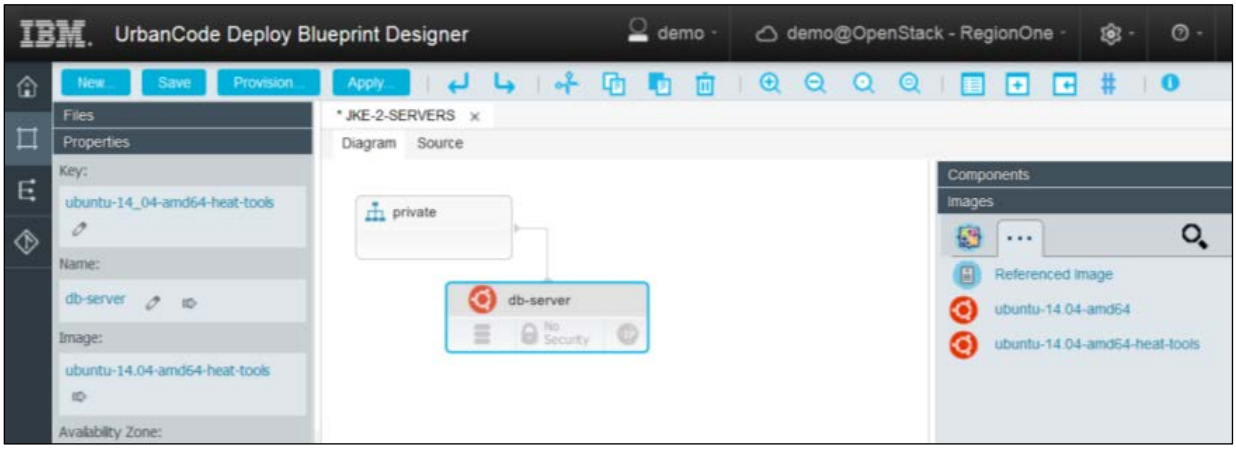
Exercise 1: Tasks and results

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**.

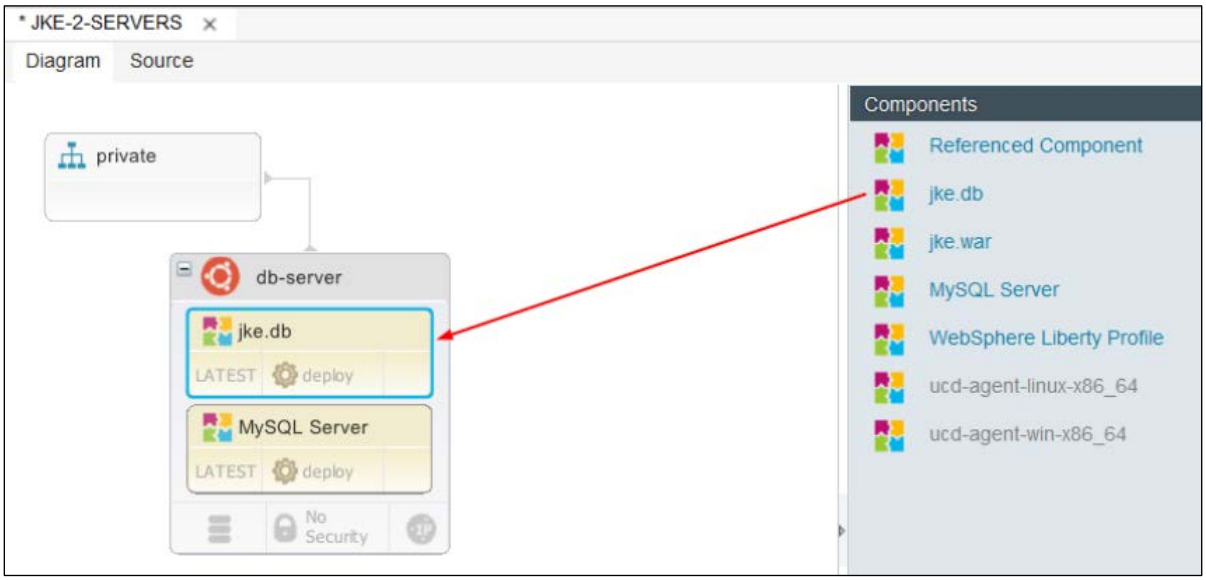
Task 2. Create the database server.

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.

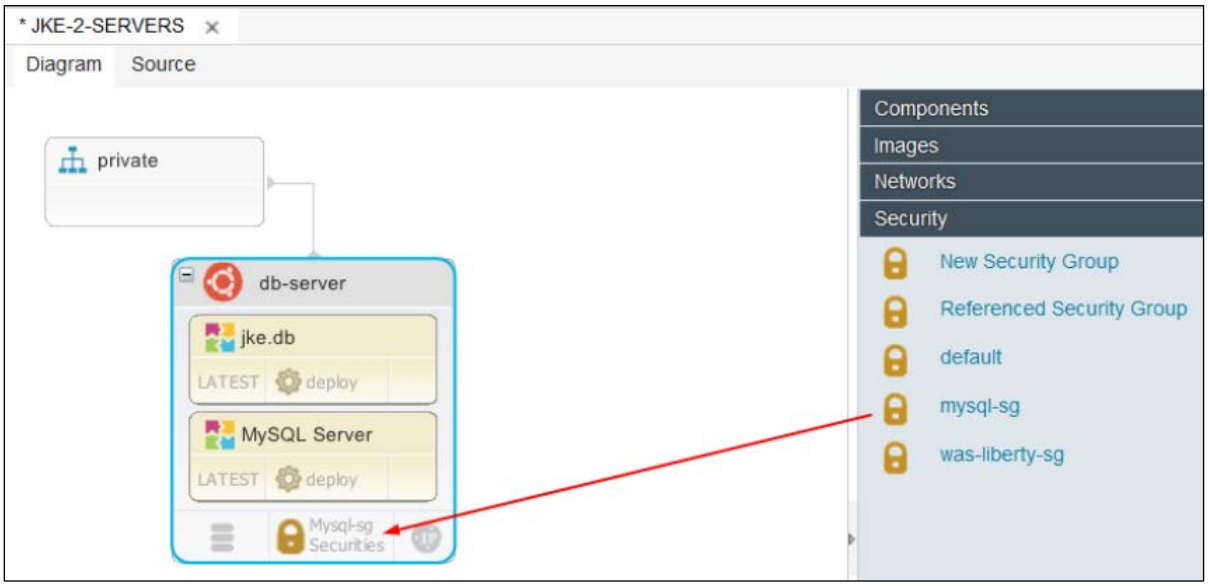


4. From the Components drawer, drag the **MySQL Server** component on to the db-server image, select **JKE** as the application, and click **OK**. Click **CHOOSE**, select **deploy** as the IBM UrbanCode Deploy component process, and click **OK**.

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.

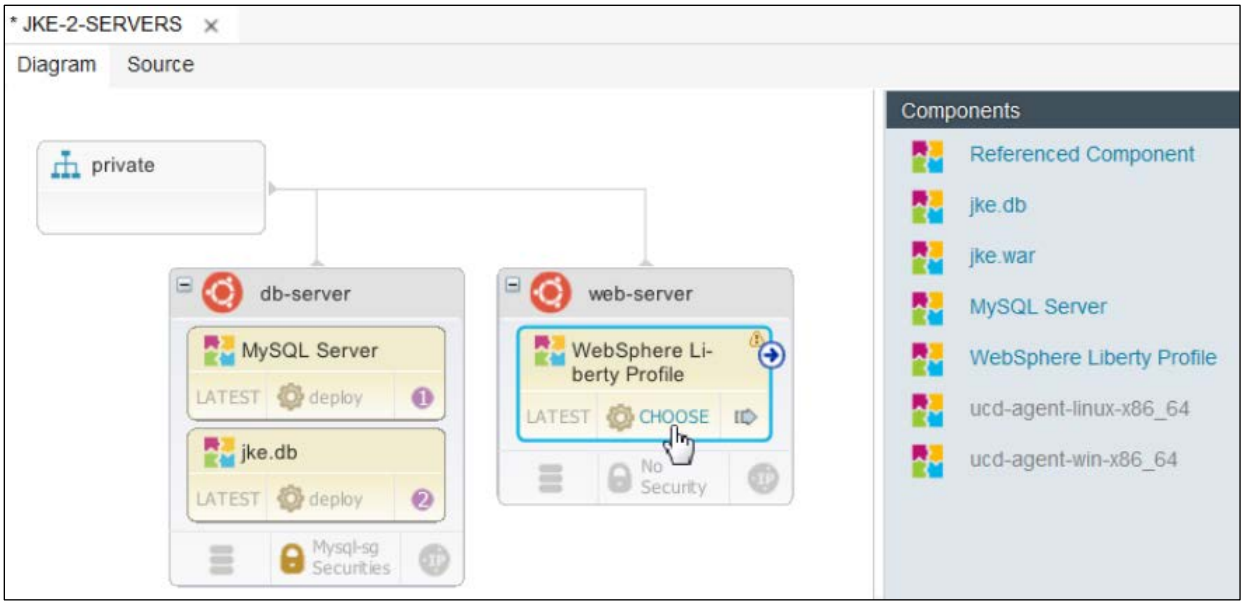


The integration with the cloud provider enables the blueprint to specify the security policies that are set for the provisioned server.

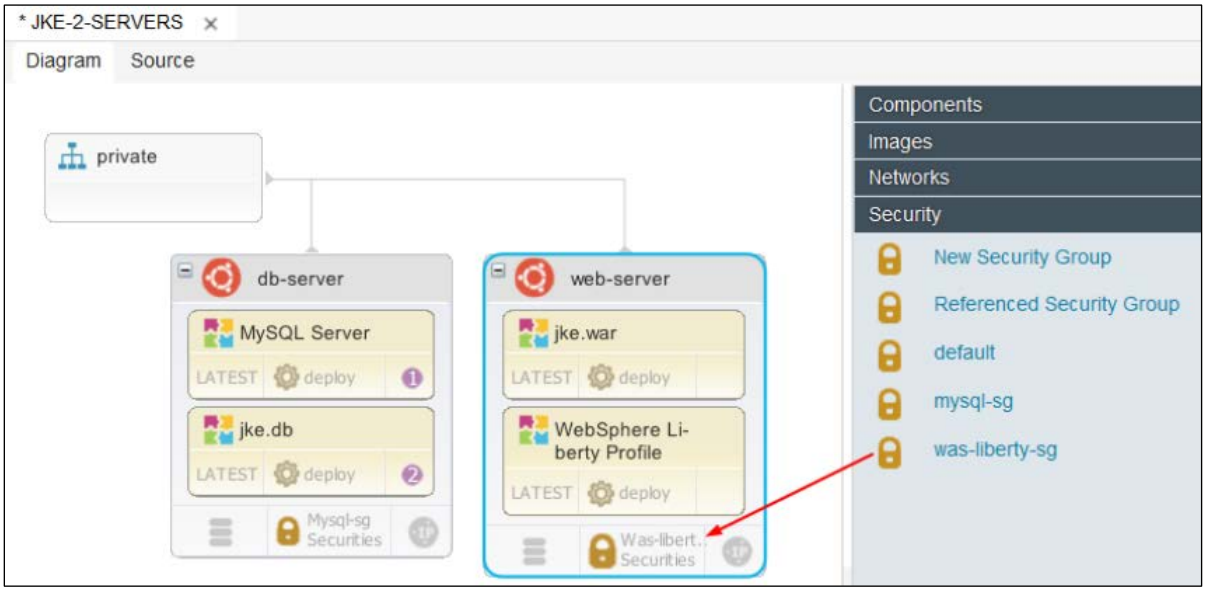
Note: The security policy for a database server opens port 3306 for MySQL access and port 22 for SSH access.

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.



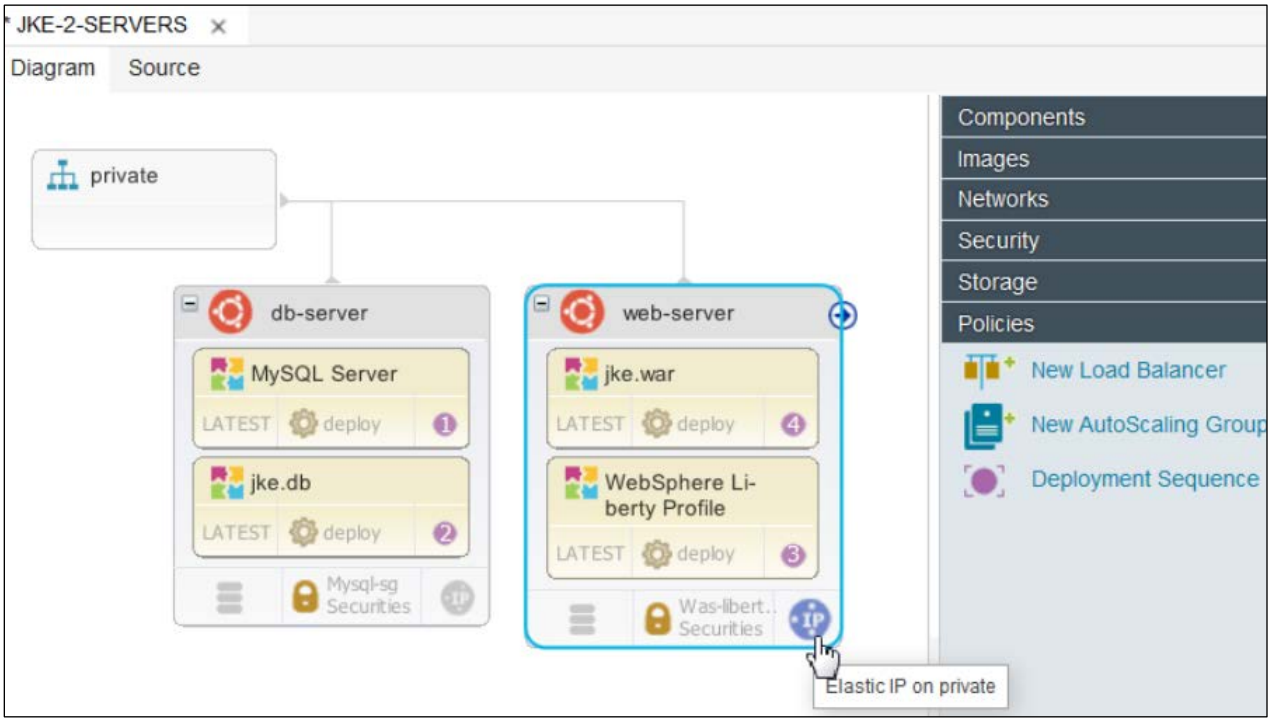
The security policy for web servers opens ports `80`, `443`, `9080`, and `9443` for http(s) traffic and port 22 for SSH traffic.

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.

IBM Training

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:

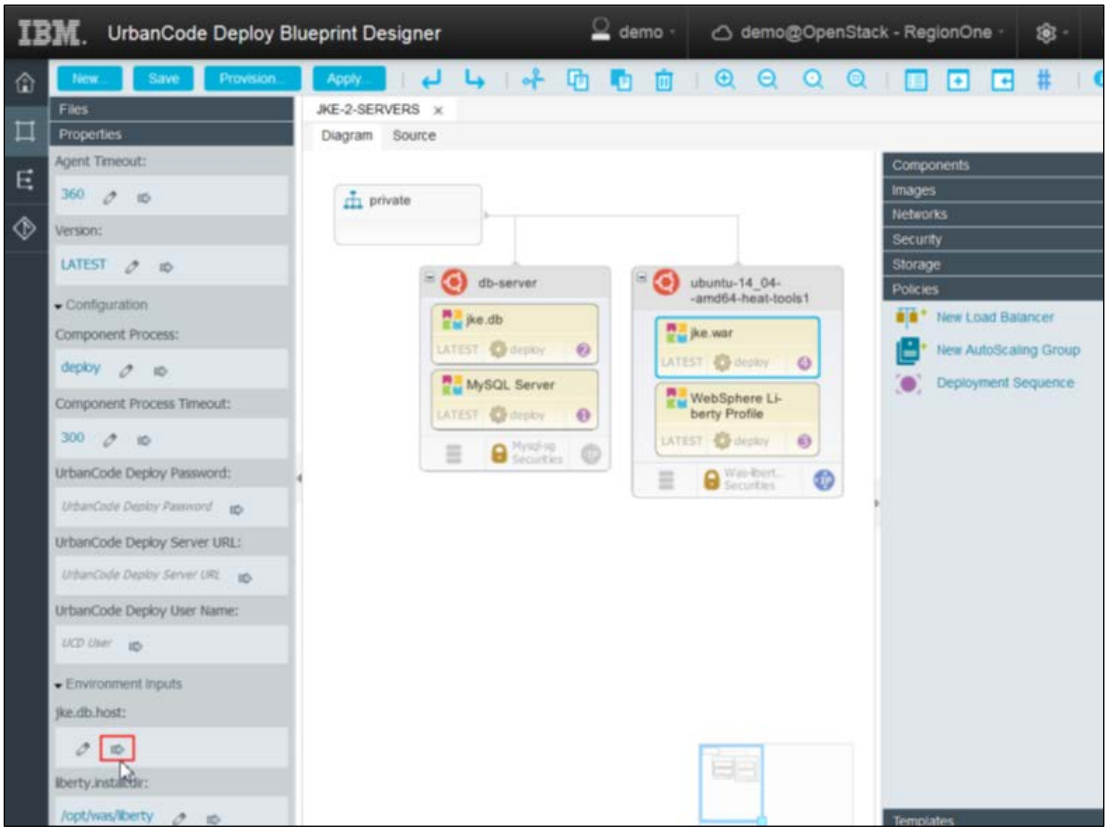
- Assign the IP address to the database server.

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

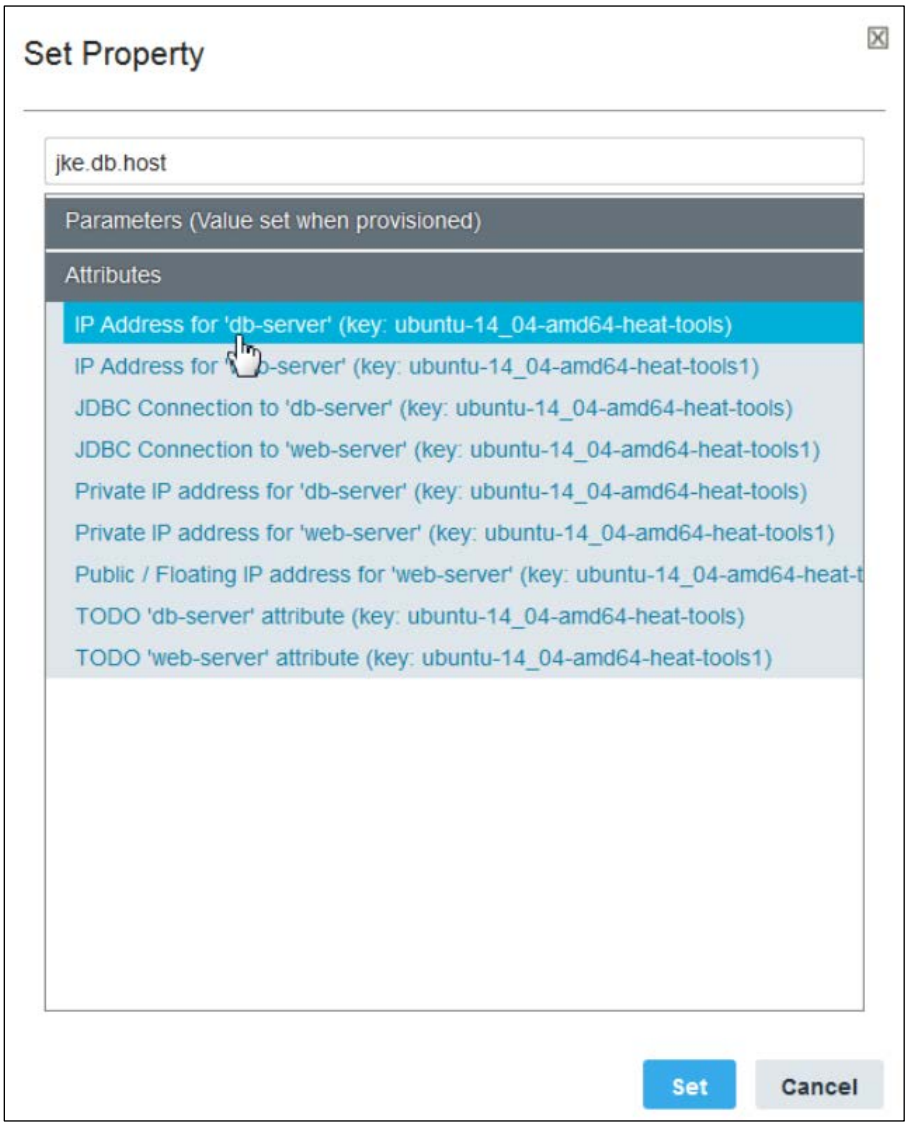
Exercise 2:
Tasks and results

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  .



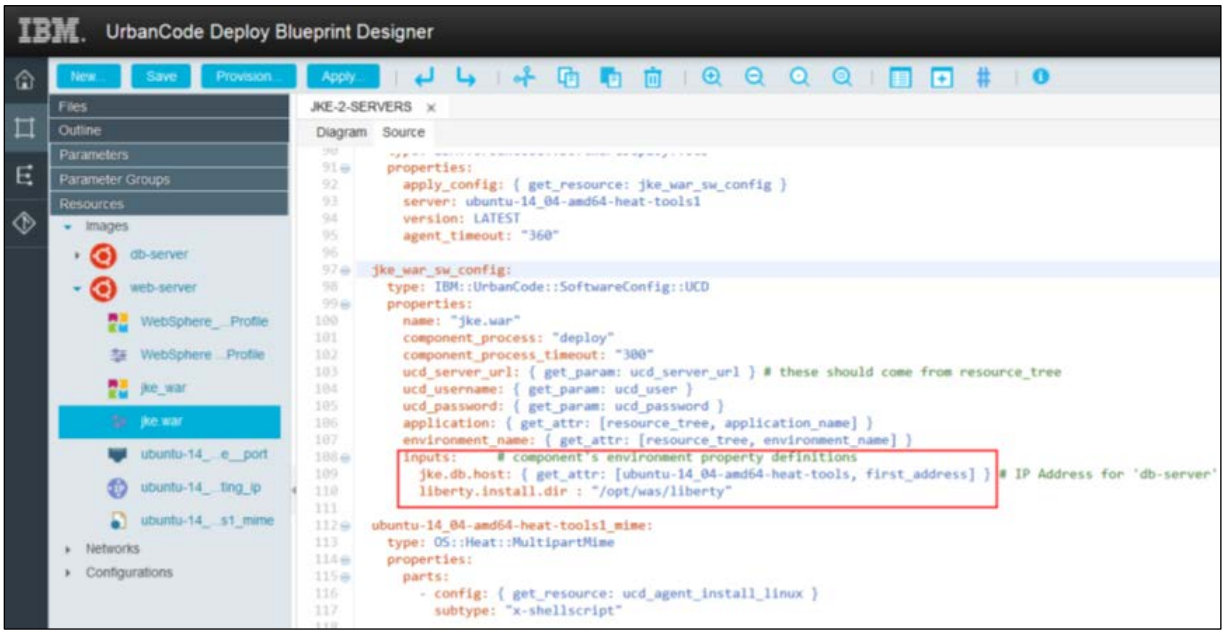
4. Click **Attributes** to expand the section, click **IP Address for db-server**, and click **Set**.



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 Training

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:

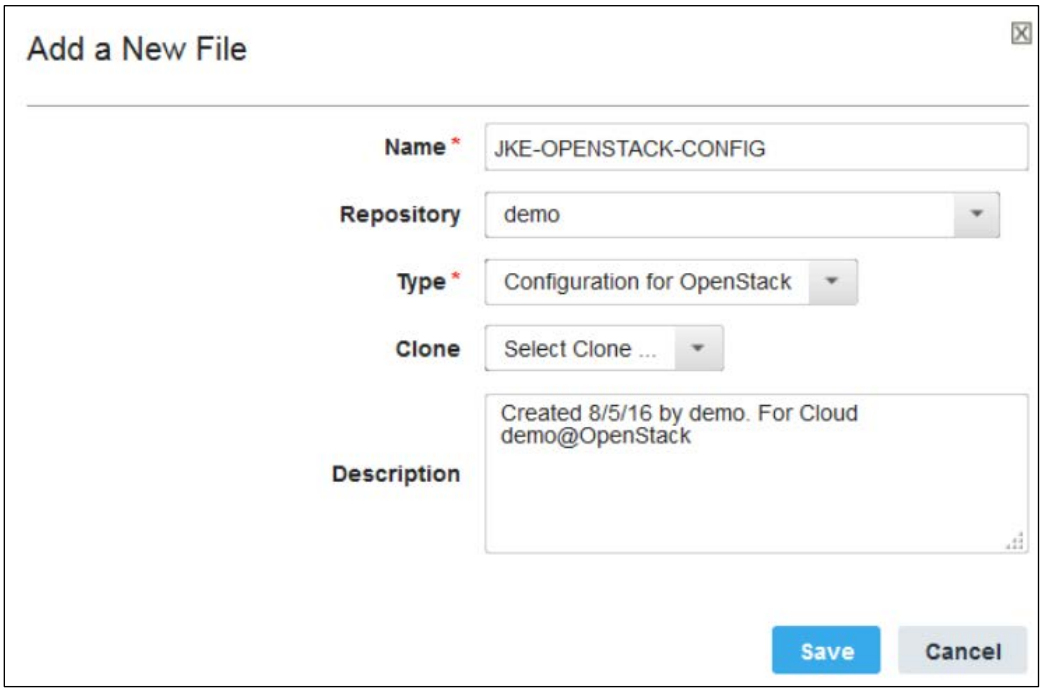
- Create the configuration file.

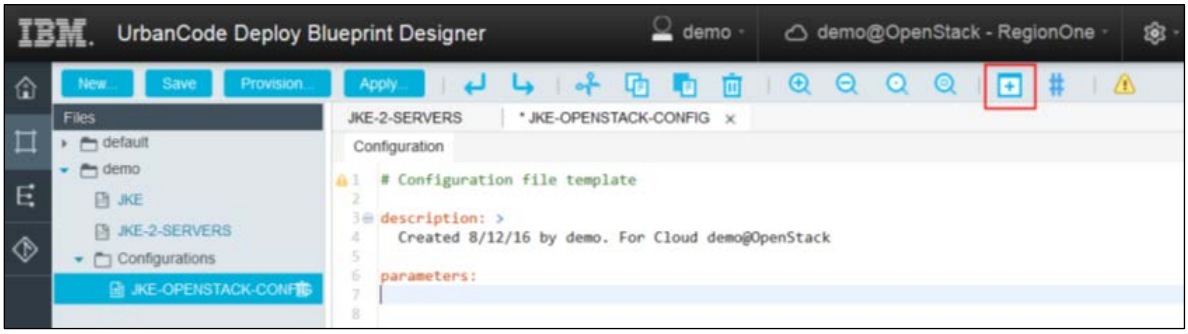
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 3:
Tasks and results

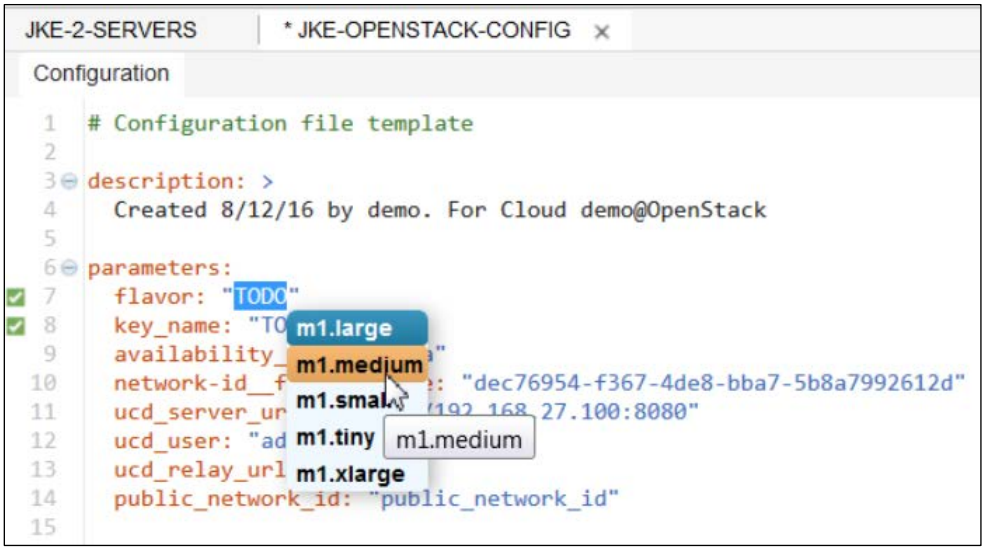
Task 1. Create the configuration file.

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**.





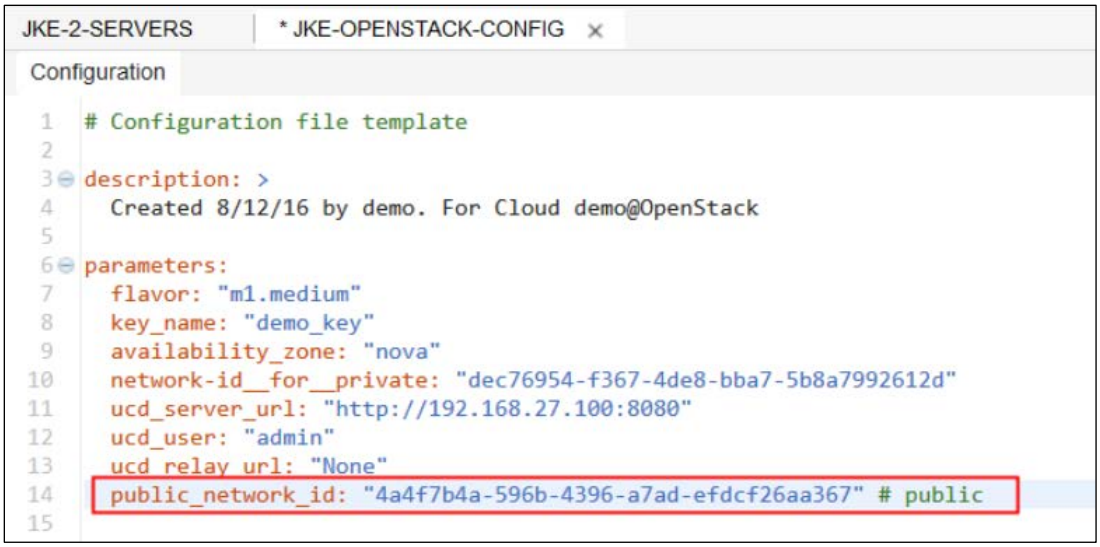
6. To assign a value to the flavor parameter, click the word **TODO** to the right of flavor parameter, press Ctrl+Space bar at the same time, and select **m1.medium** from the options.



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

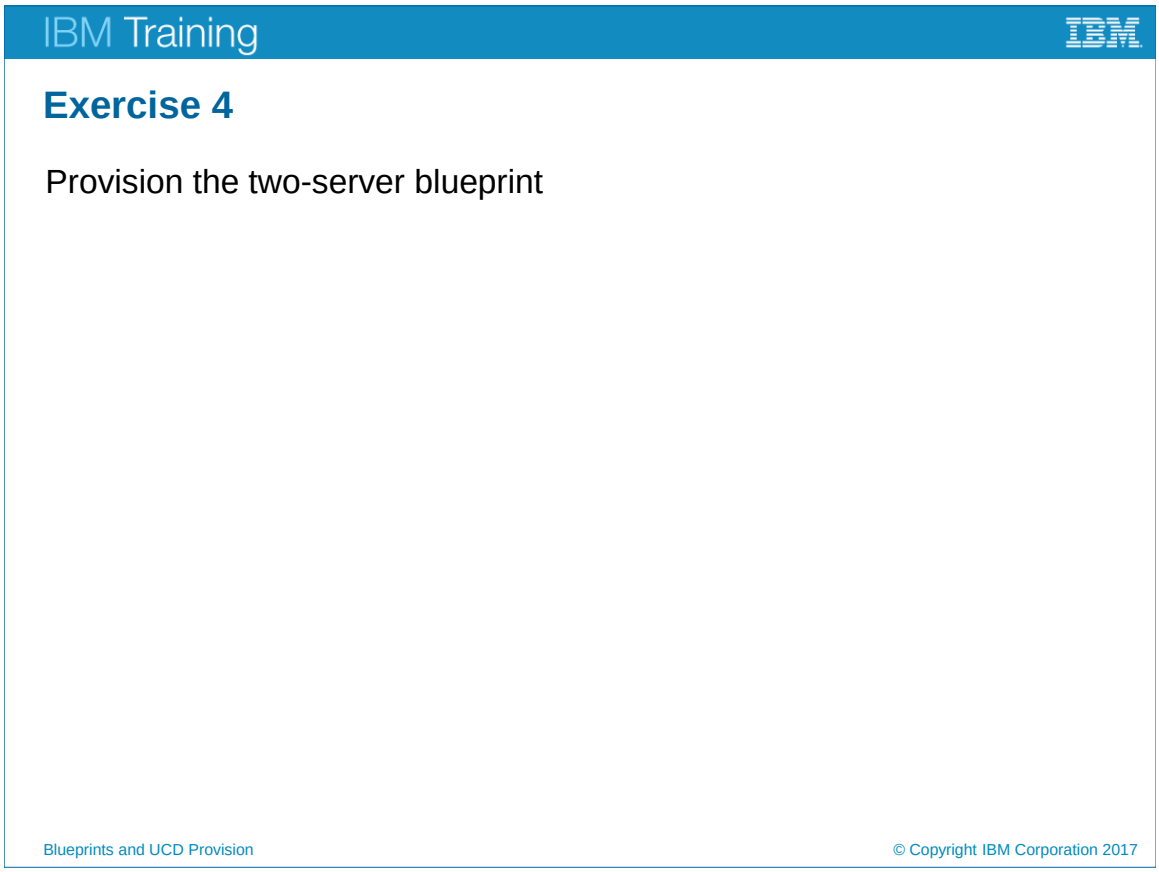
   - key_name: **demo_key**
   - public_network_id: **public**

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



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

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

To do this, you will:

- Provision the two server blueprint.

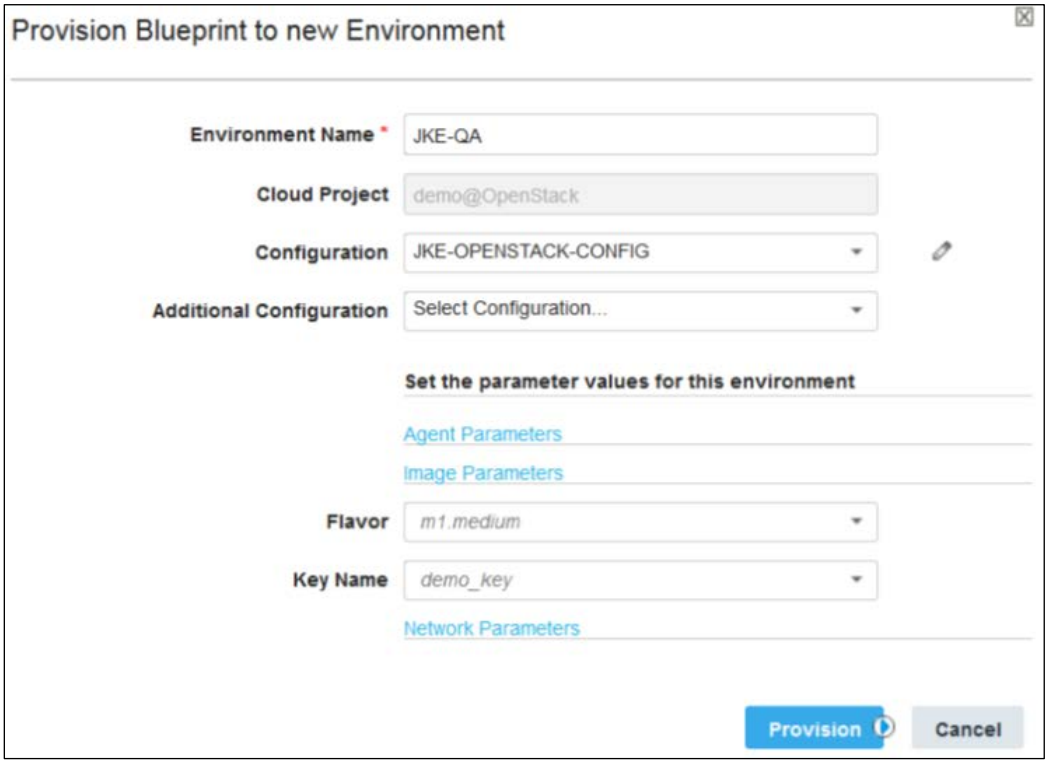
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

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**.

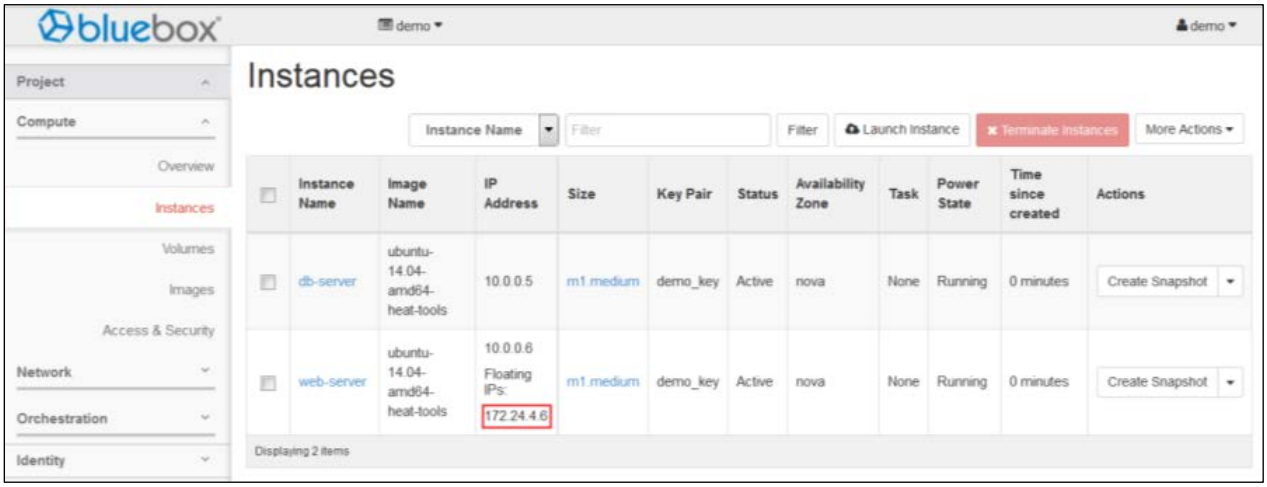
   Note that the parameter fields now have values that are prepopulated.



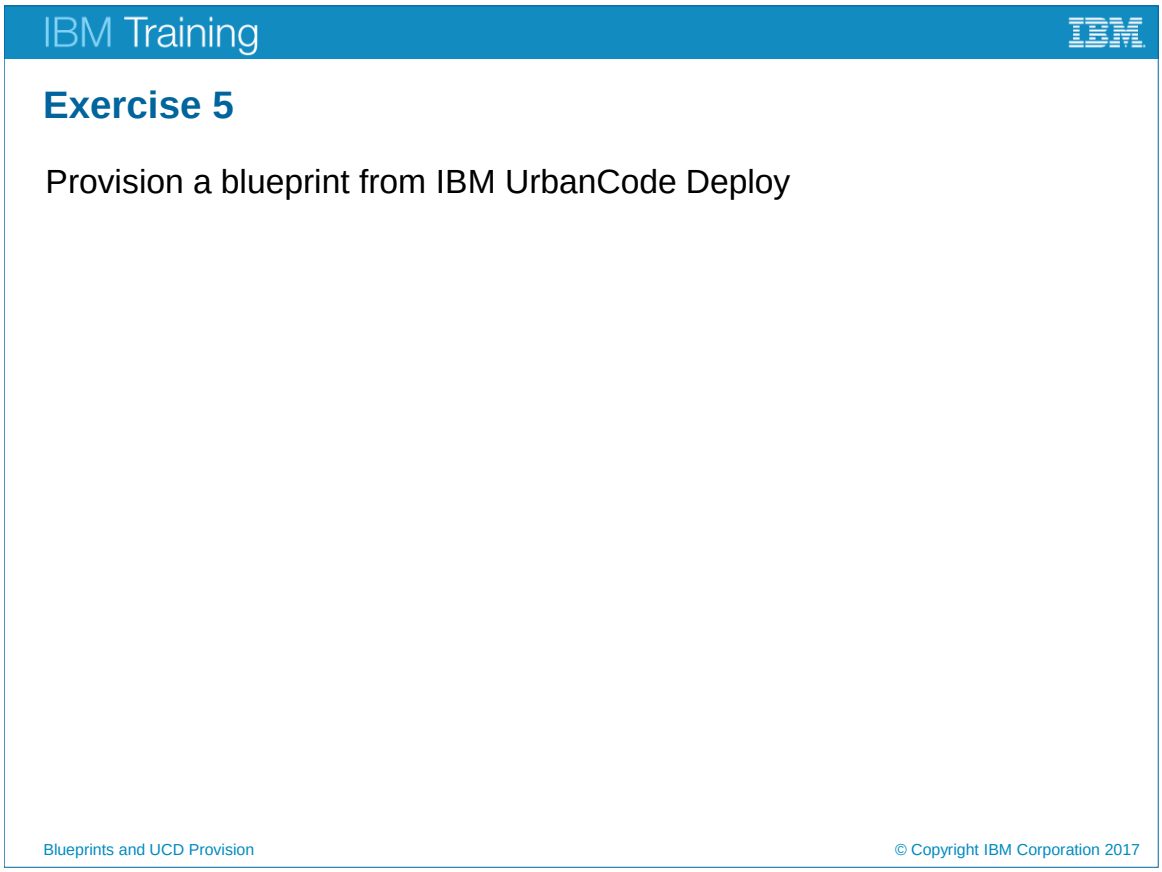
- Click **Provision**.

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.



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

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:

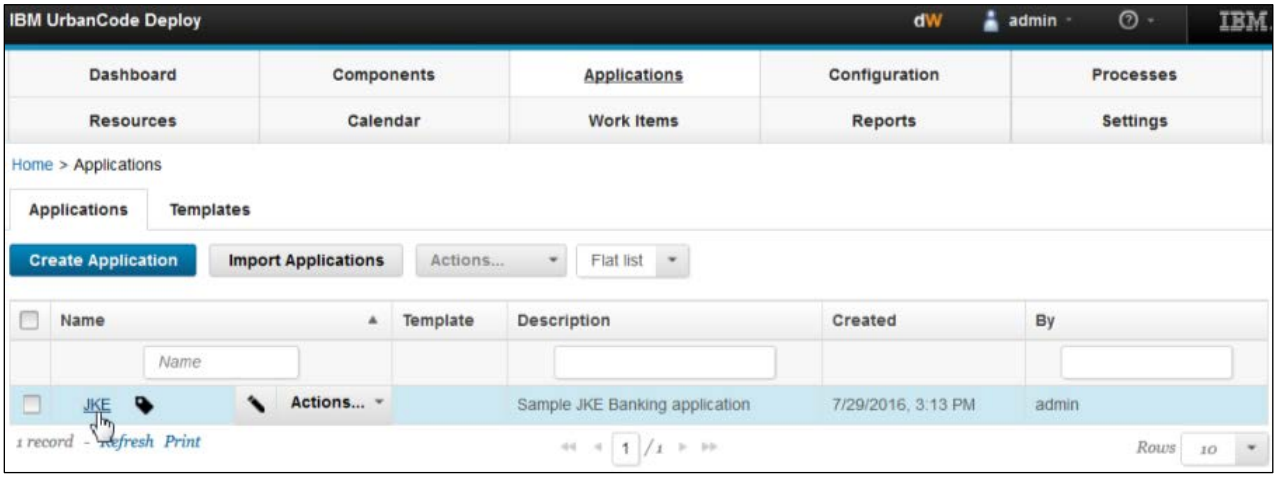
- Create the environment.
- Observe the provisioned environment.
- Verify the application.

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

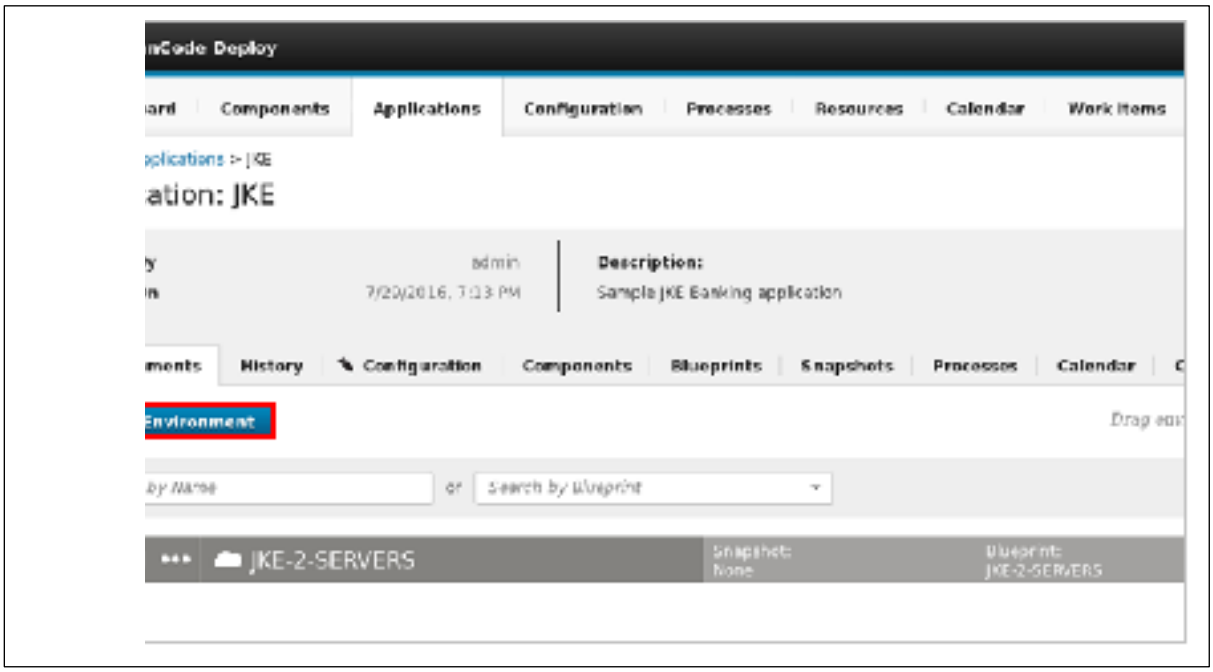
Exercise 5:
Tasks and results

Task 1. Create an environment.

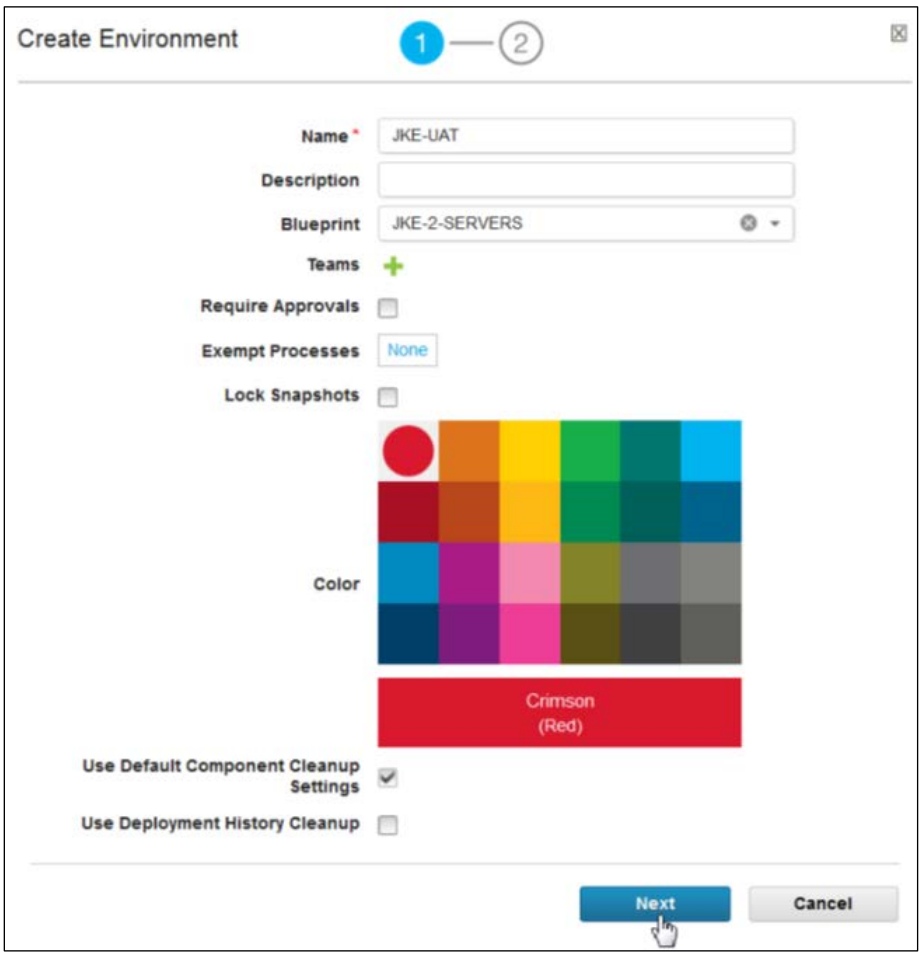
1. From the browser, click the **UrbanCode Deploy** tab. If necessary, type `admin` for the user ID and `admin` for the password.
2. From the top navigation, click **Applications**, and in the Name column, click **JKE**.



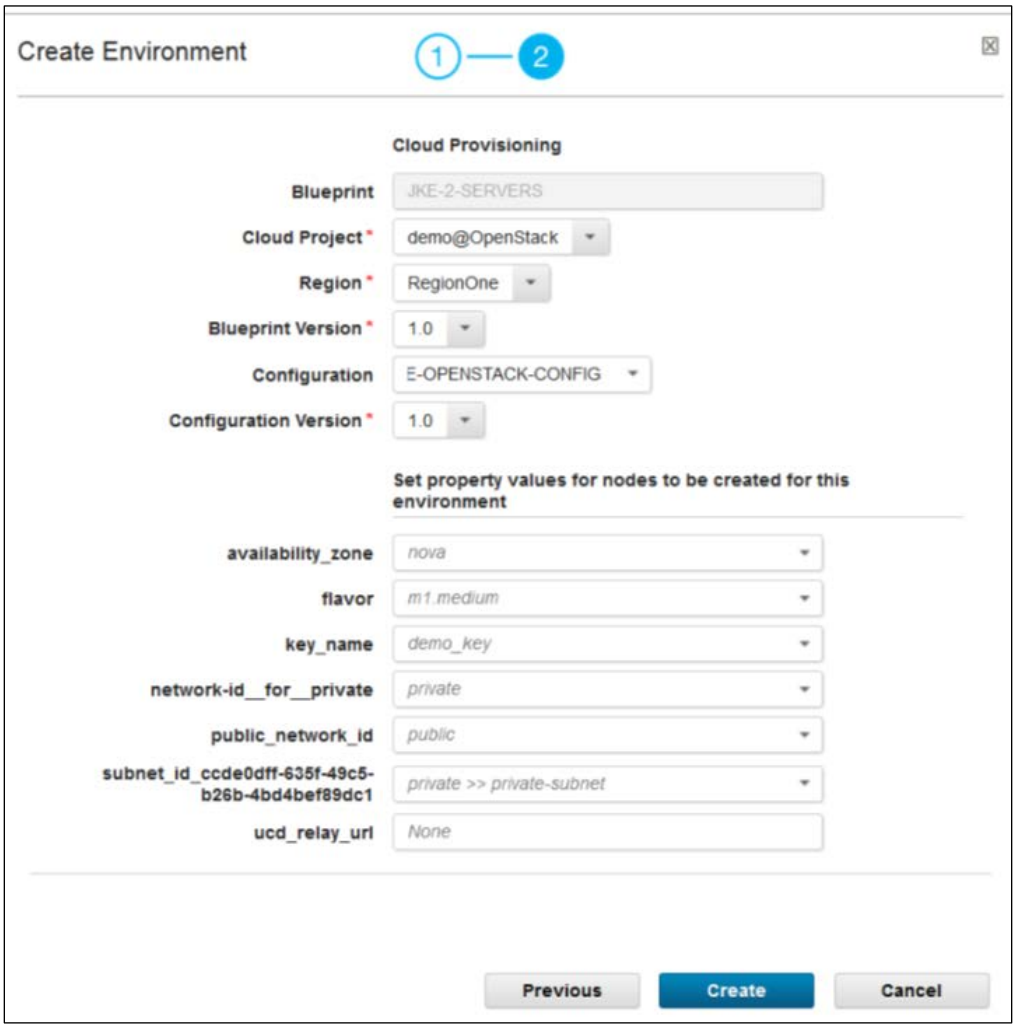
3. On the **Environments** tab, click **Create Environment**.



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**.

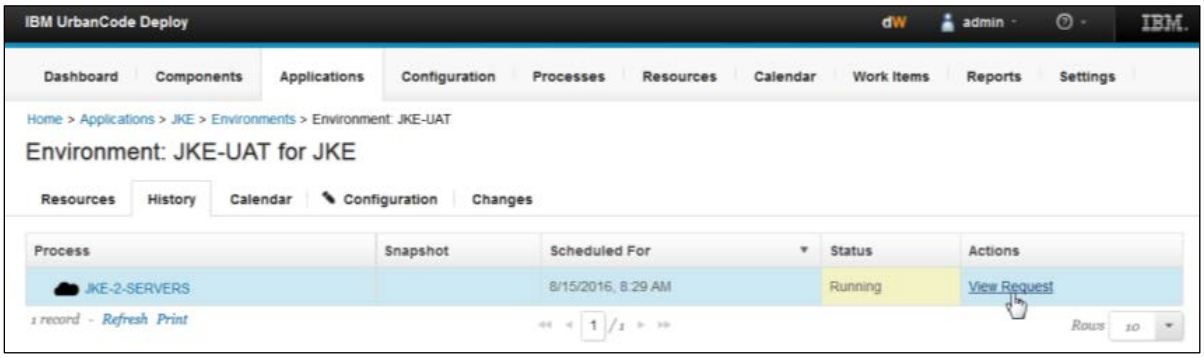


5. On the second page of the Create Environment window, do the following steps:
   - Accept the default values the for **Cloud Project** and **Region** fields.
   - In the **Blueprint Version** field, select **1.0**. Note that the property values are populated from the configuration file.
   - In the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

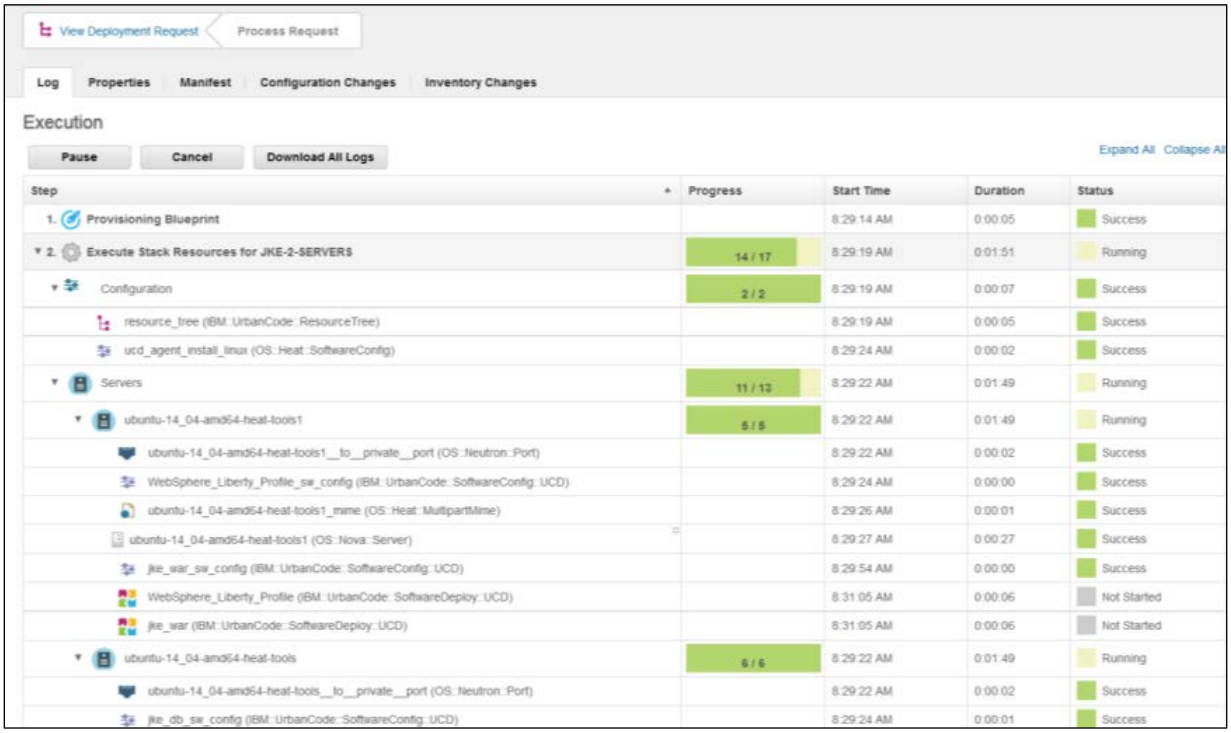


Task 2. Observe the provisioned environment.

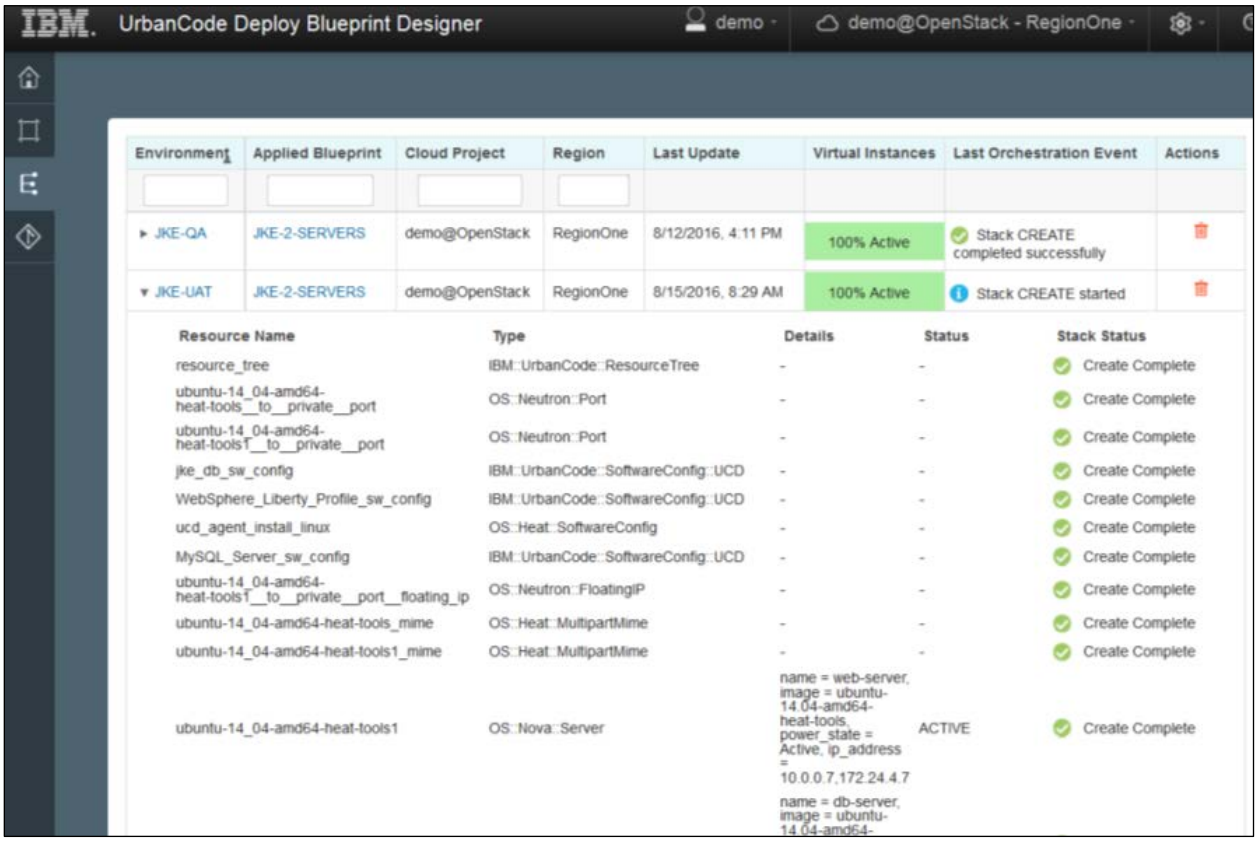
1. To see the status of the provisioned environment, in the Actions column, click **View Request**.



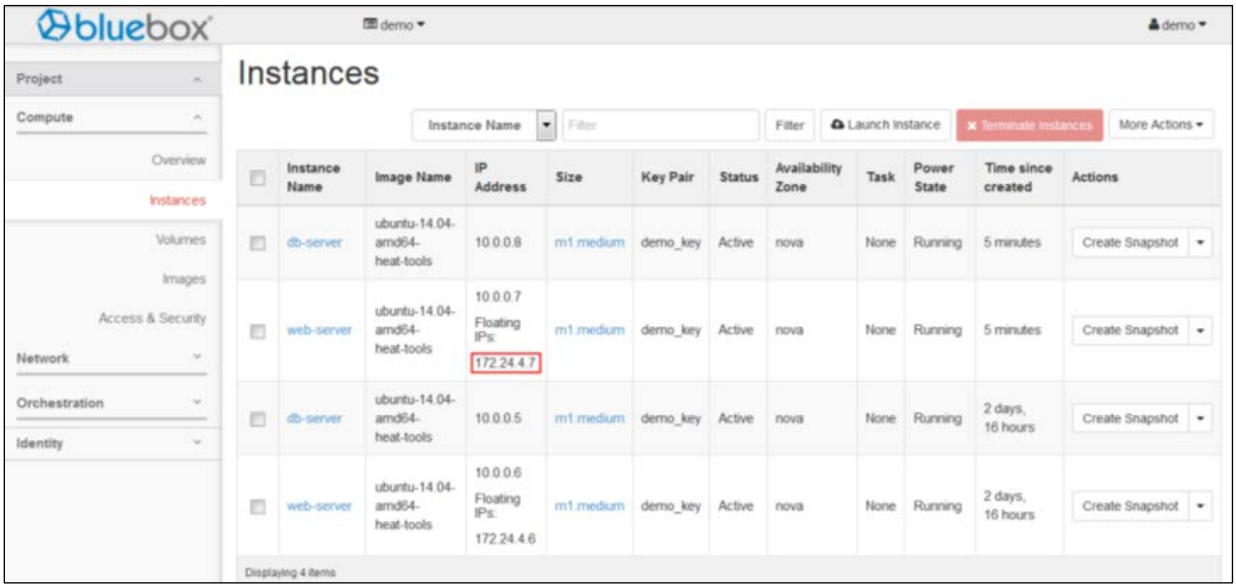
2. Expand the steps to see the progress of the deployment.



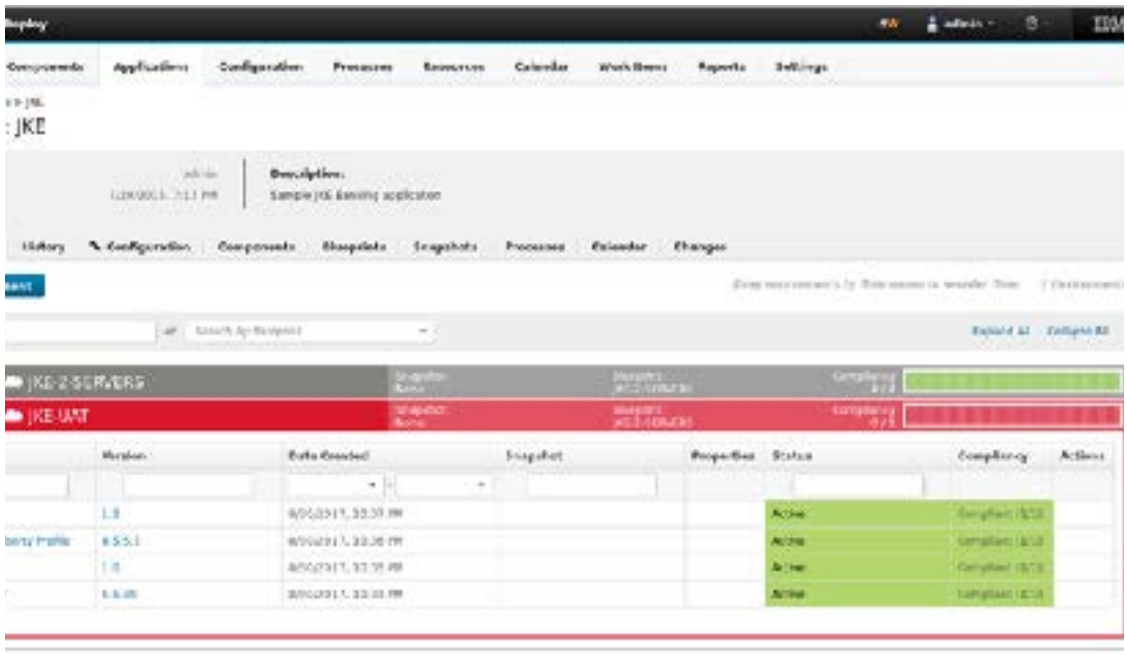
3. In the Blueprint Designer, click the **Environments** tab, and expand **JKE-UAT** to see the progress.



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.



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

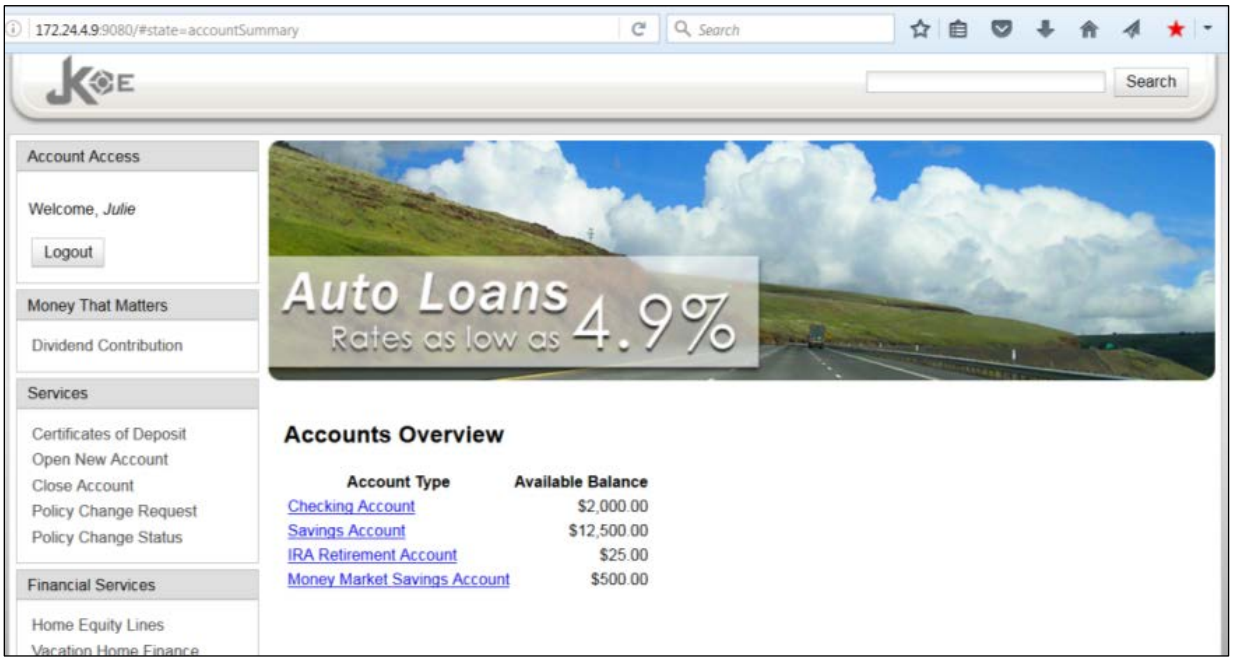


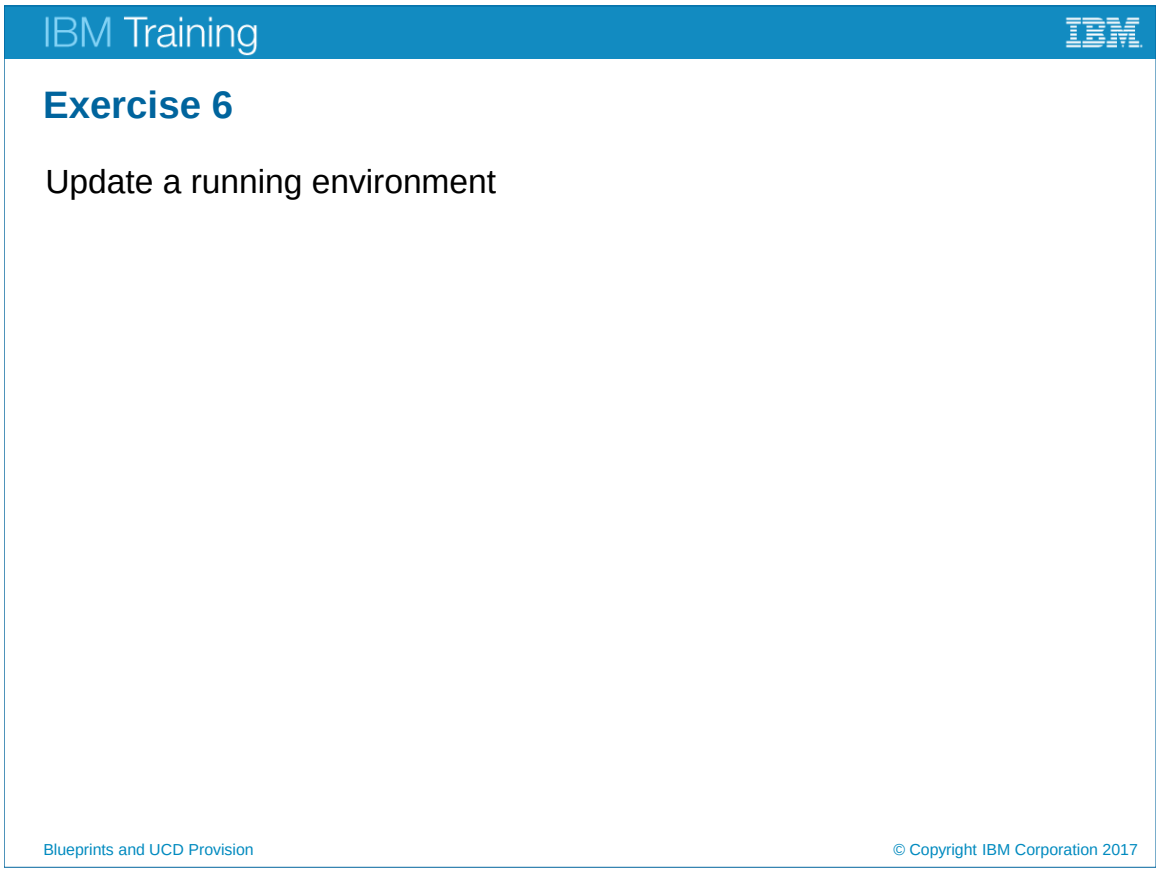
Task 3. Verify the application.

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.





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.

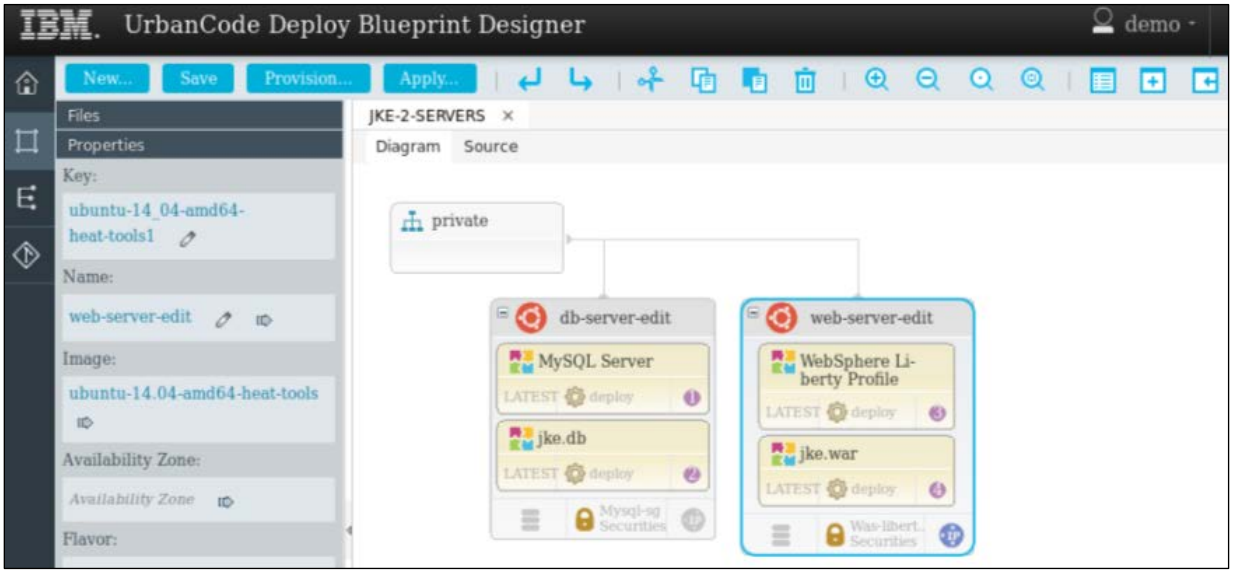
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

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`.

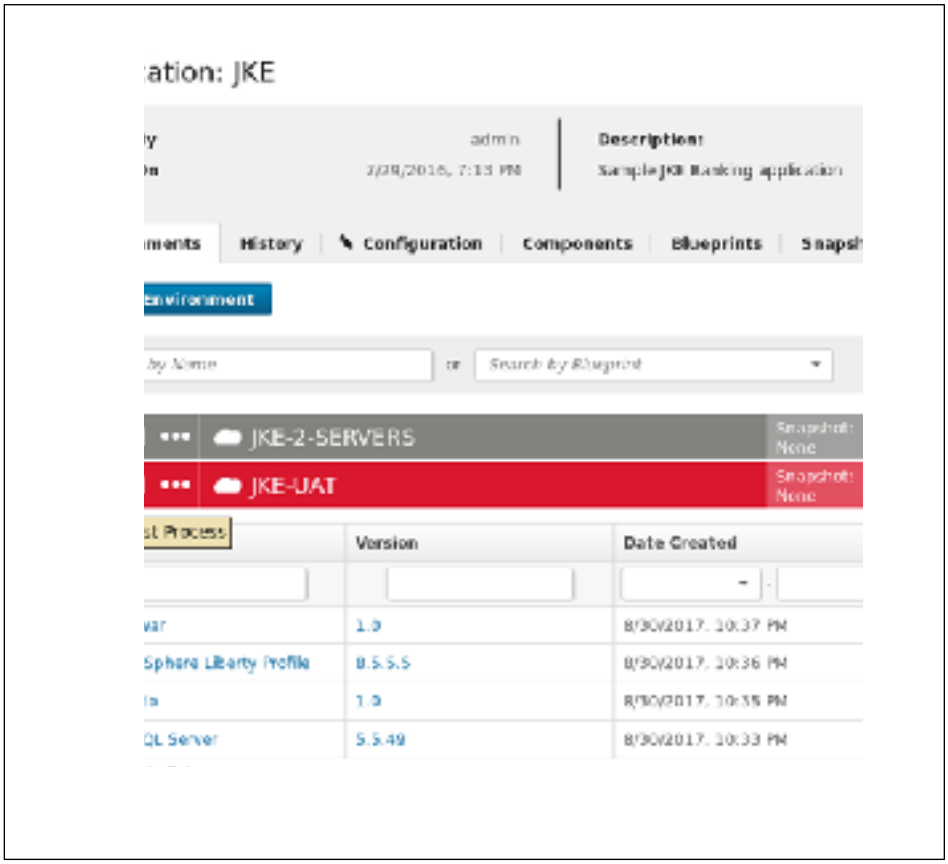


4. Save the blueprint.

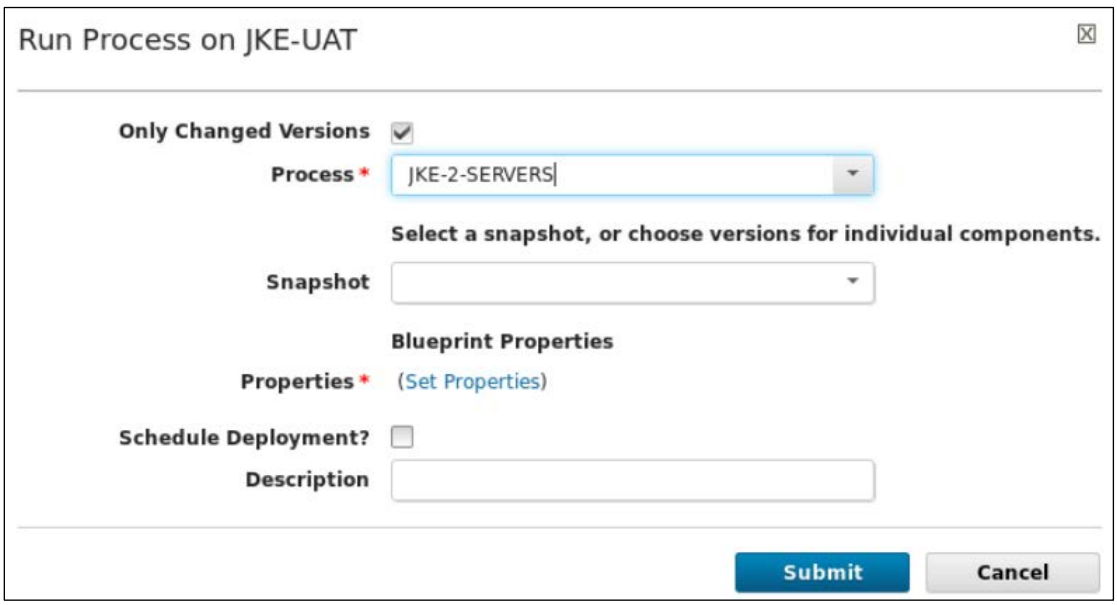
Task 2. Run the modified blueprint.

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.

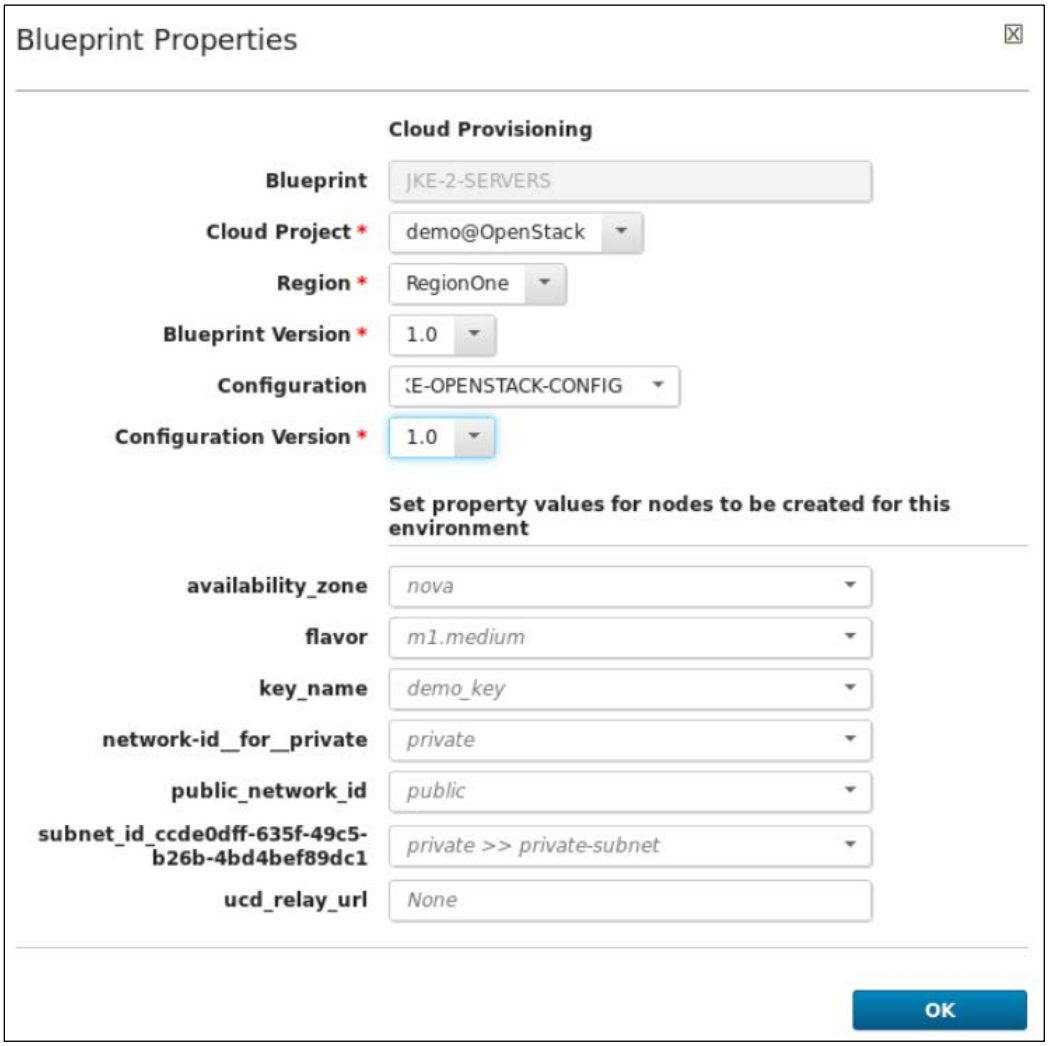
4. To run the same process request for the JKE-UAT environment, click the **Request Process** icon  .



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

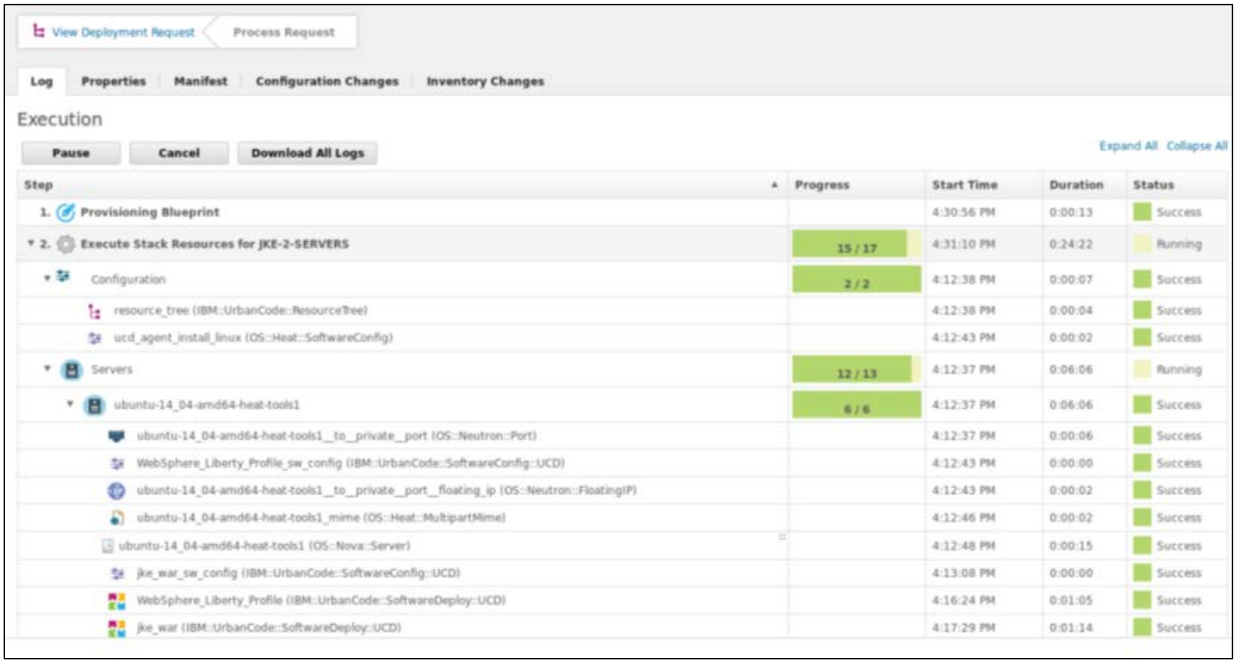


6. In the Blueprint Properties window, do the following steps:
   - In the **Blueprint Version** field, select **1.0**.
   - In the **Configuration** field, select **JKE-OPENSTACK-CONFIG**.
   - In the **Configuration Version** field, select **1.0**.

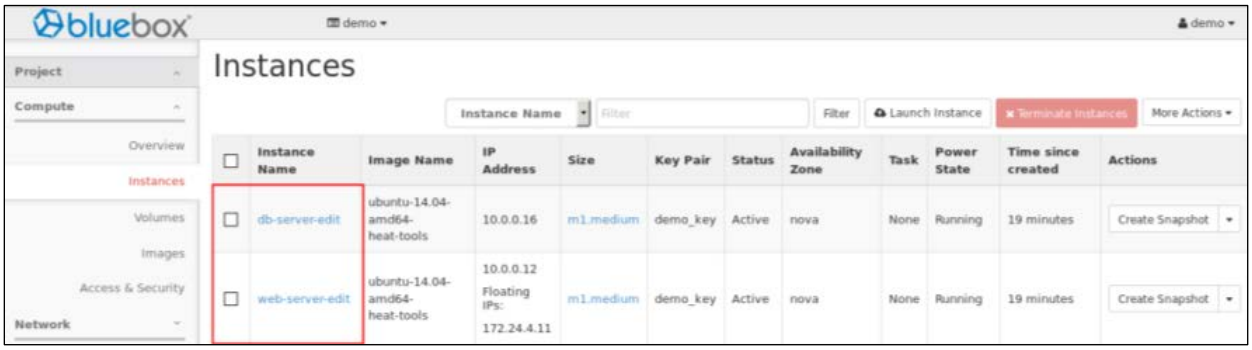


7. Click **OK**, and click **Submit**.

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.



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



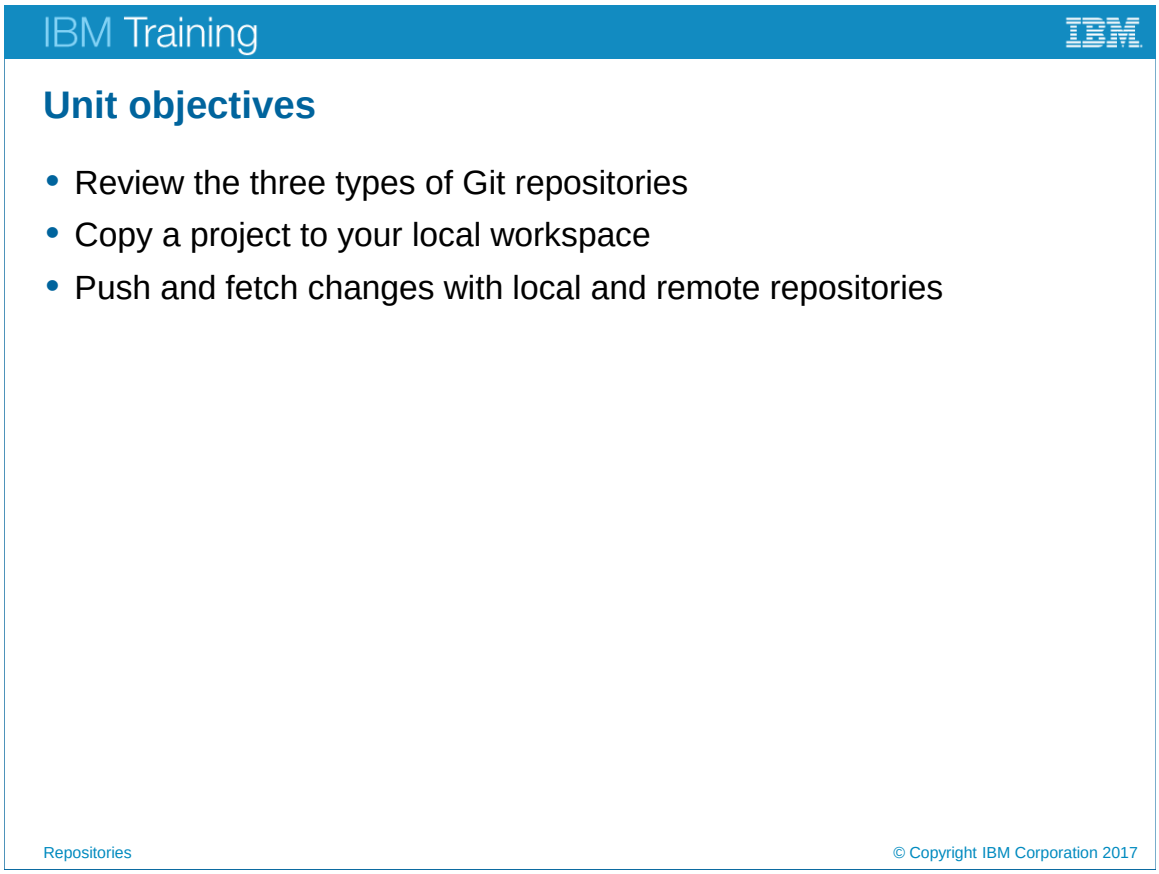
Although changing the name of a server is a minor request, the ability to update a running environment can have more complex changes. For example, you can change the component versions that are deployed to the environment and then run the application process to deploy the new versions. You can also add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

IBM Training

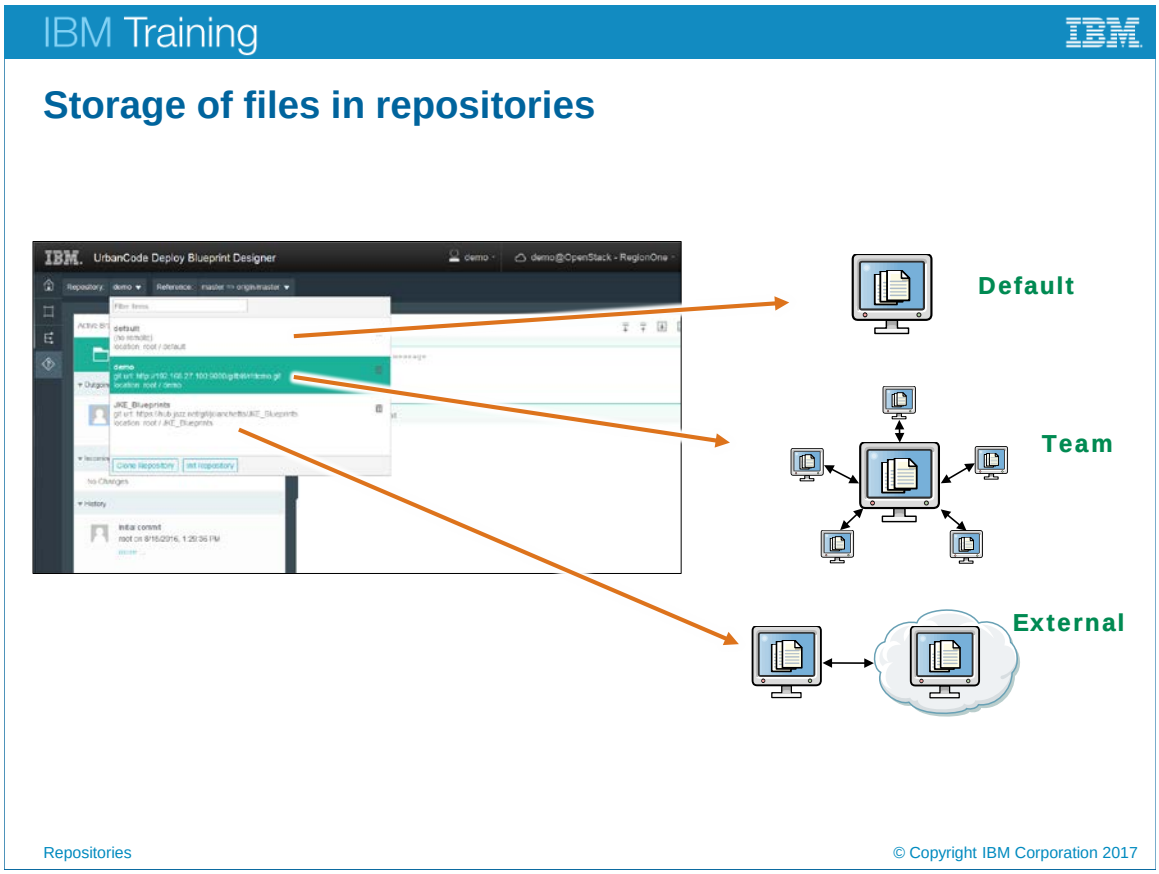
Repositories

IBM UrbanCode Deploy Blueprint Designer - v6.2.1.1



Unit objectives

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



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.

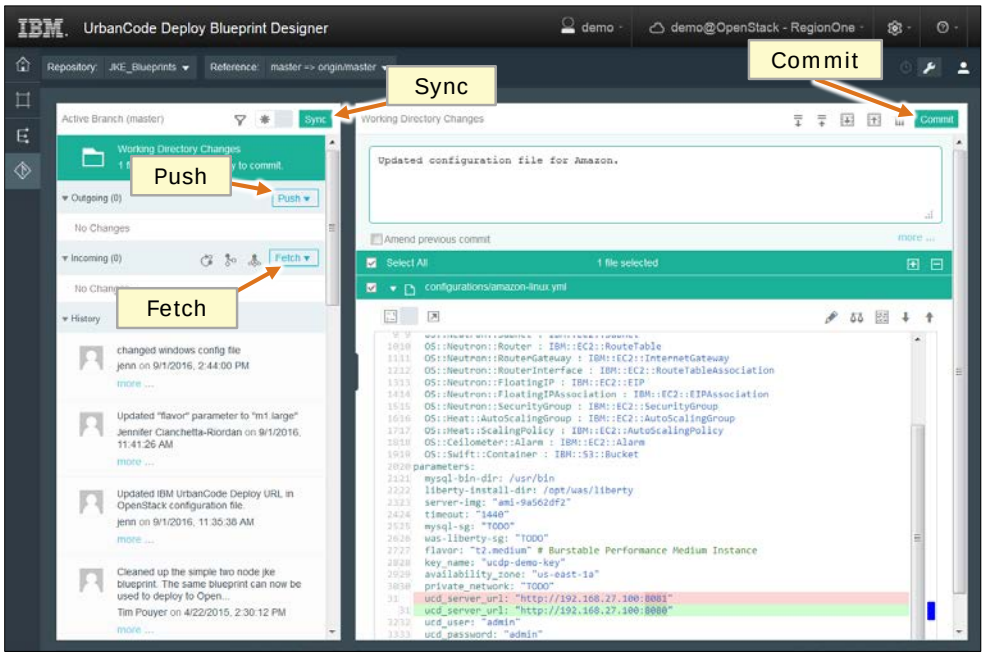
Team repository: The blueprint design server creates a repository on the local Git server for each team, where all members of the team can access those files. For example, throughout the lab exercises in this course, you saved blueprints and configuration files to the demo project and any members of 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.

IBM Training

Git functions on the Repositories page



Repositories

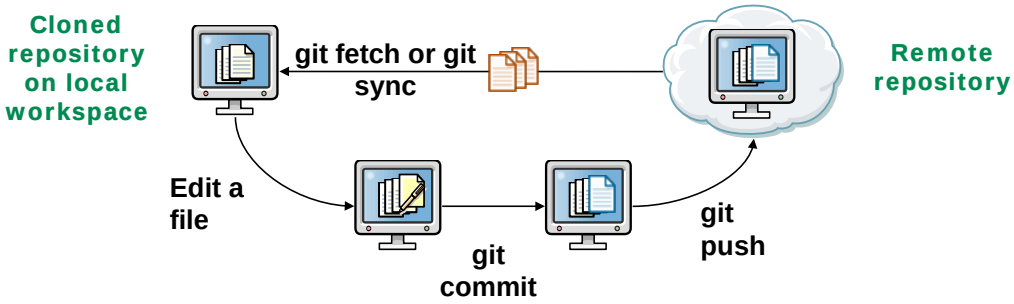
Git functions on the Repositories page

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.

IBM Training

Minimize conflicts

- Minimize conflicts by adopting a process for sharing repository files



Repositories

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.

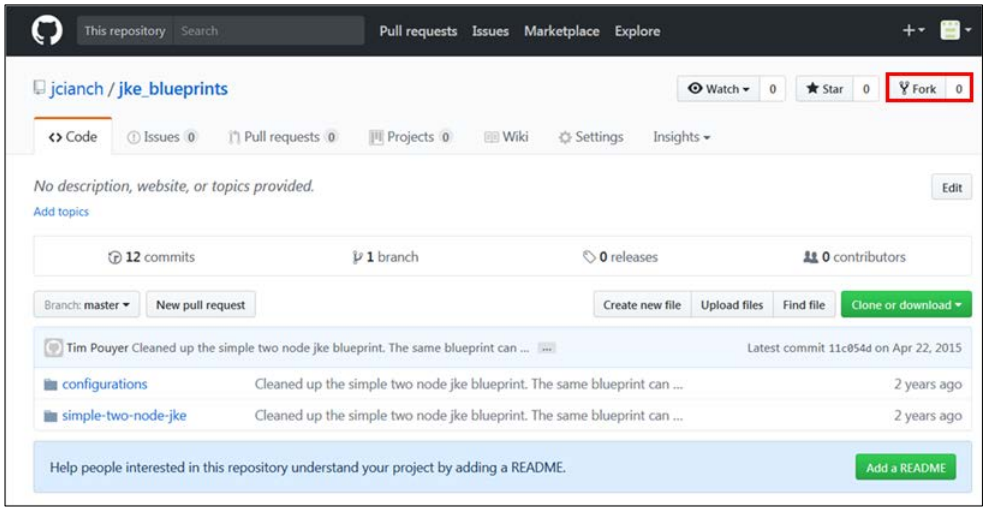
Make changes to blueprints in the Blueprint Designer.

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.

IBM Training

Forks

- Forks bridge the original repository and your personal copy



Repositories

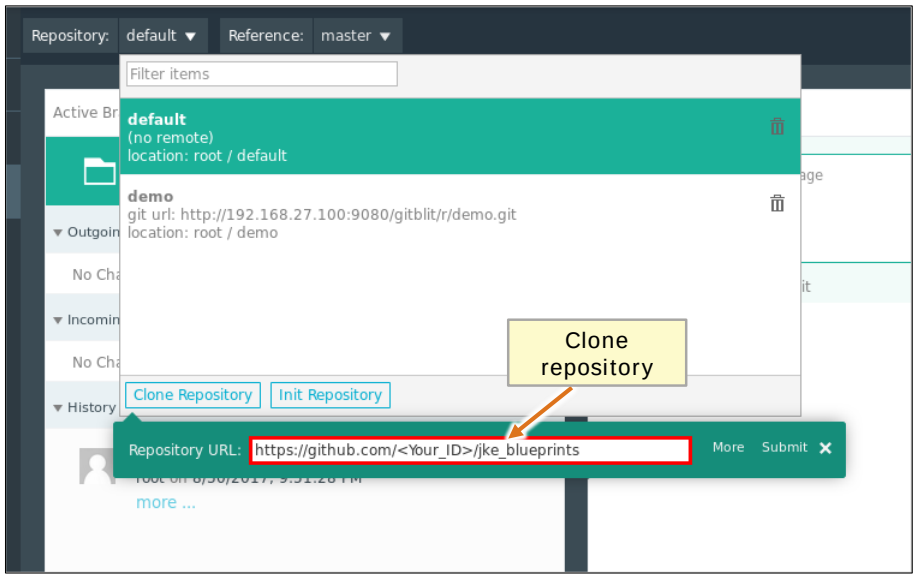
Forks

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

IBM Training

Cloned repository

- A cloned repository is a local copy on your computer



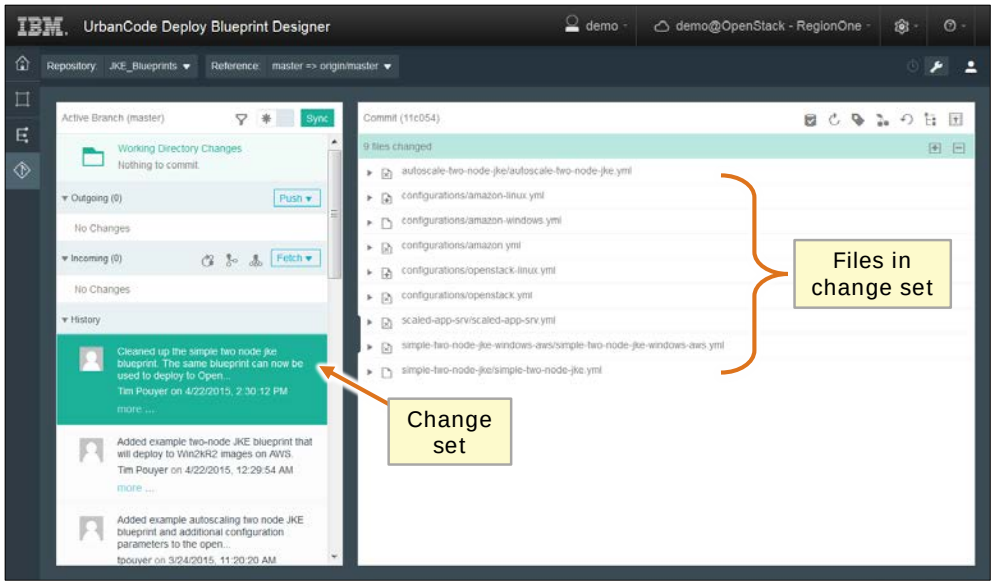
Repositories

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.

IBM Training

File history



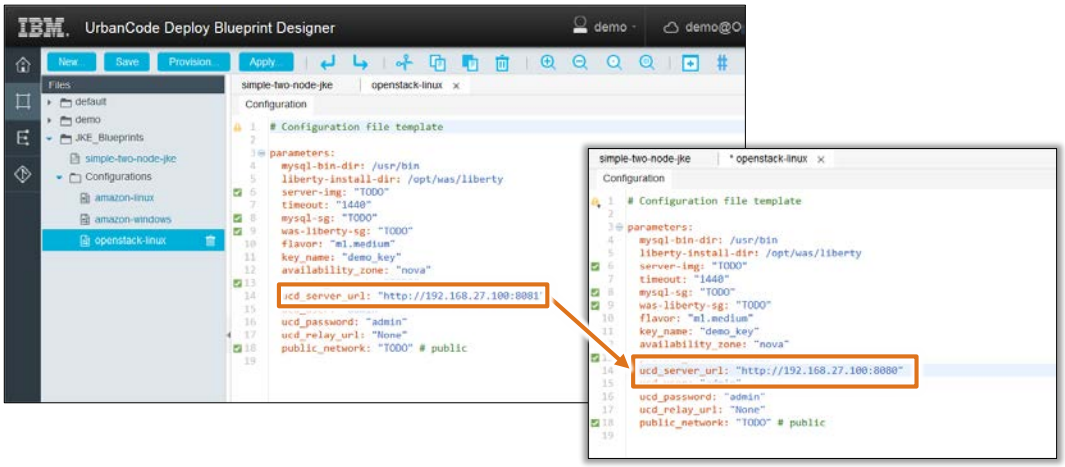
Repositories

File history

Files in your working directory can be in one of two states: tracked or untracked. When you first clone a project, all of the files are tracked, meaning that the files are unmodified because you just checked them out and haven't edited anything. You can view the history to see what activity occurred. When you begin editing files, you see untracked files. These are modified files in your working directory that were not in your last snapshot and are not in your staging area.

IBM Training

Transferring changes from local repository to remote repository



Repositories

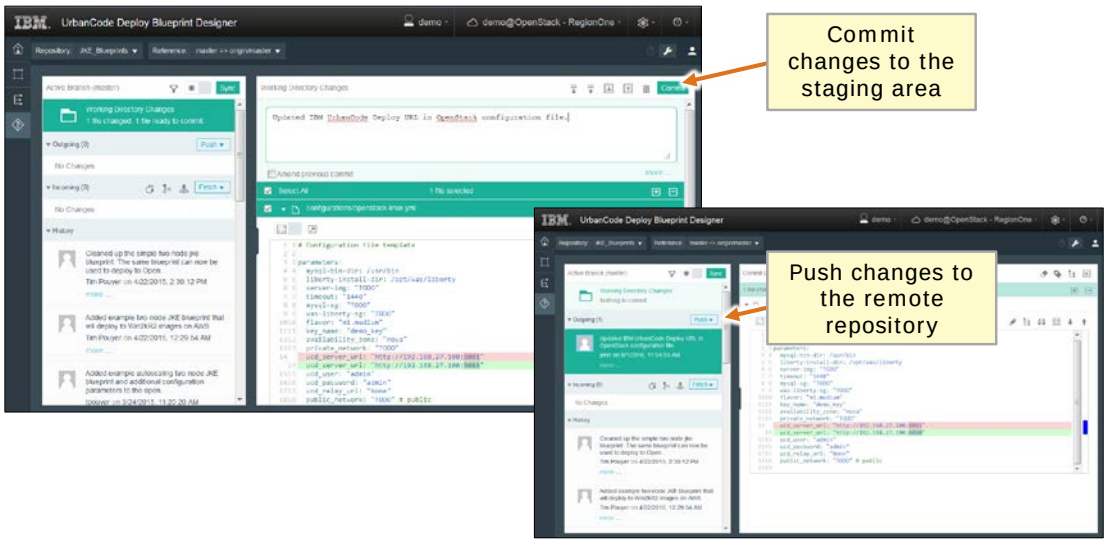
Transferring changes from local repository to remote repository

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.

IBM Training

Committing changes

- Changes are stored as change sets in the working directory



Repositories

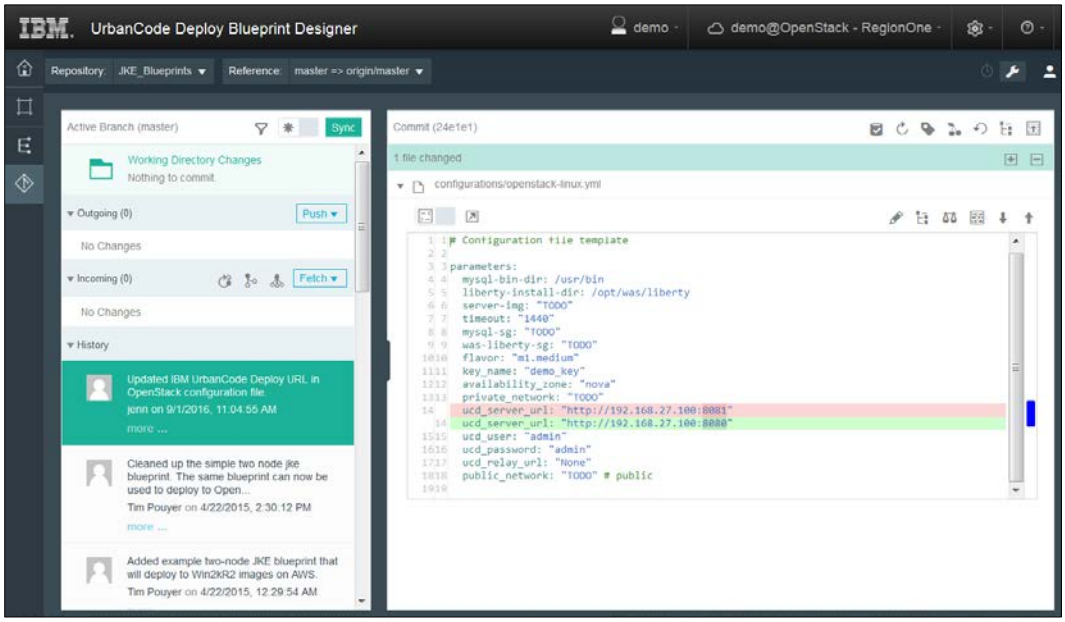
Committing changes

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.

IBM Training

Commit history

- The commit history shows the update



Repositories

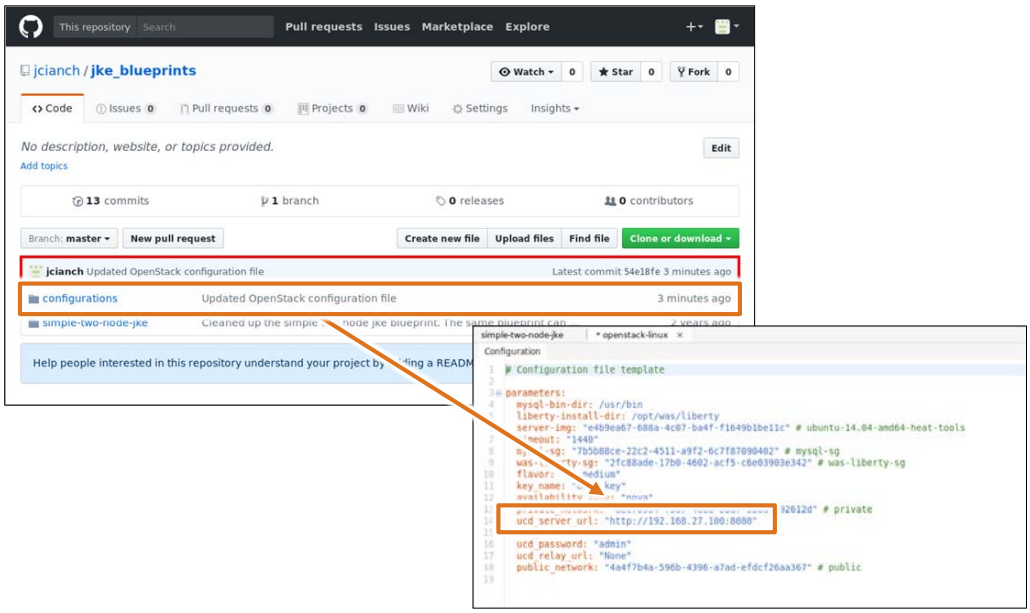
Commit history

When you click an item in the History section, you can see the updates to the file in the right pane.

IBM Training

Recent activity

- Recent activity displays in the remote project



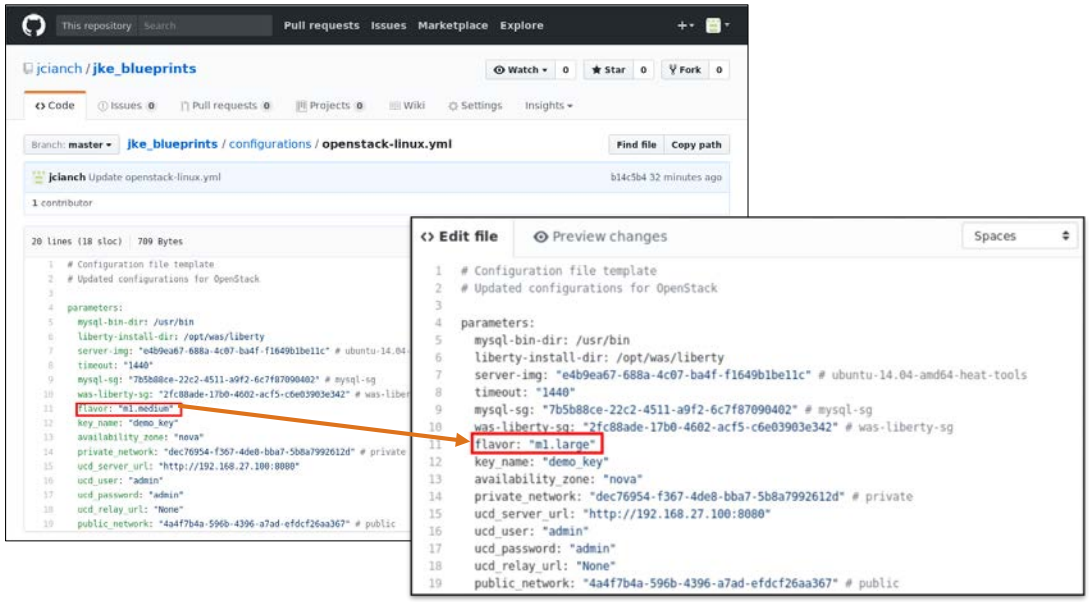
Recent activity

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

IBM Training

Editing the source code

- You can edit the code of blueprints and configuration files



Repositories

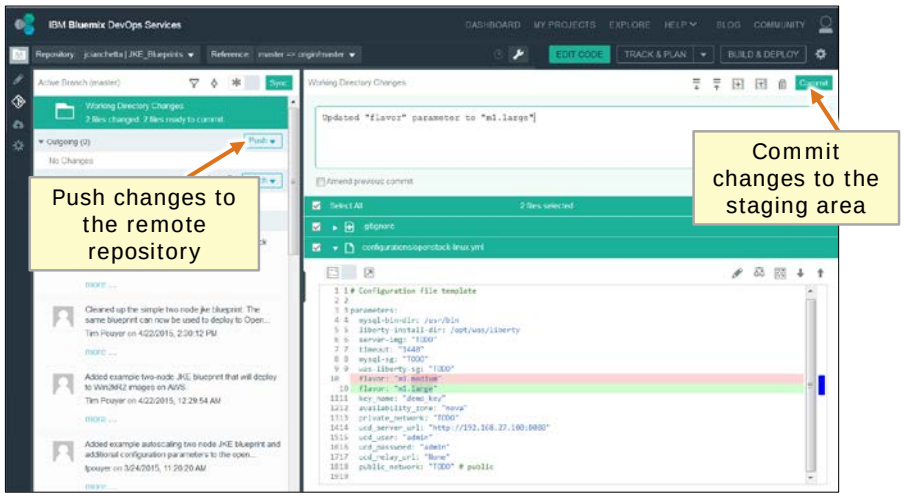
Editing the source code

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

IBM Training

Making changes available to local repositories

- Remote repositories make changes available to local repositories



Repositories

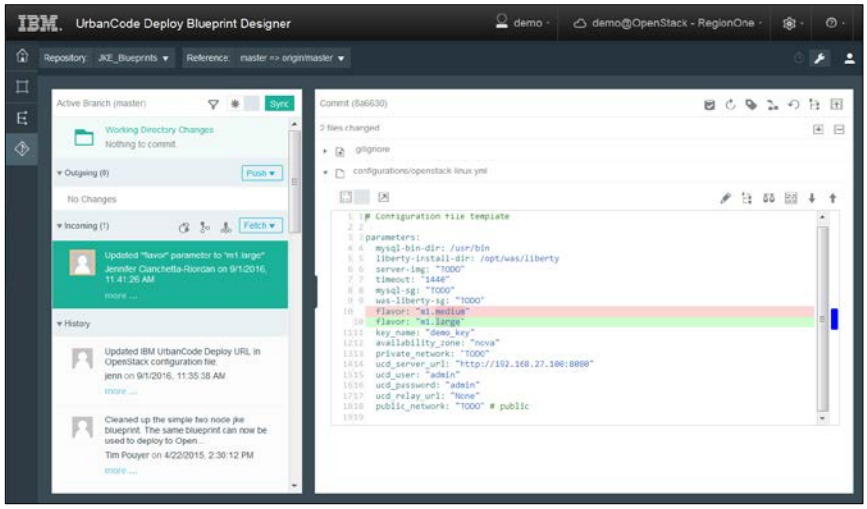
Making changes available to local repositories

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



Repositories

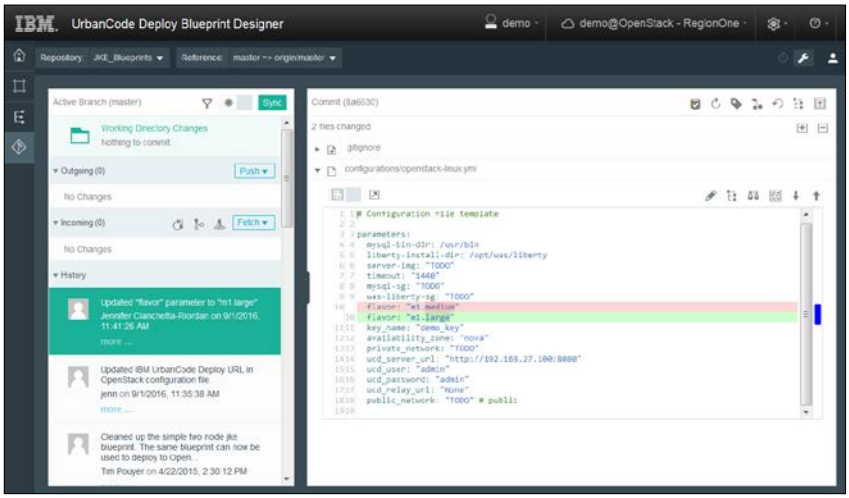
Fetching

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

Syncing

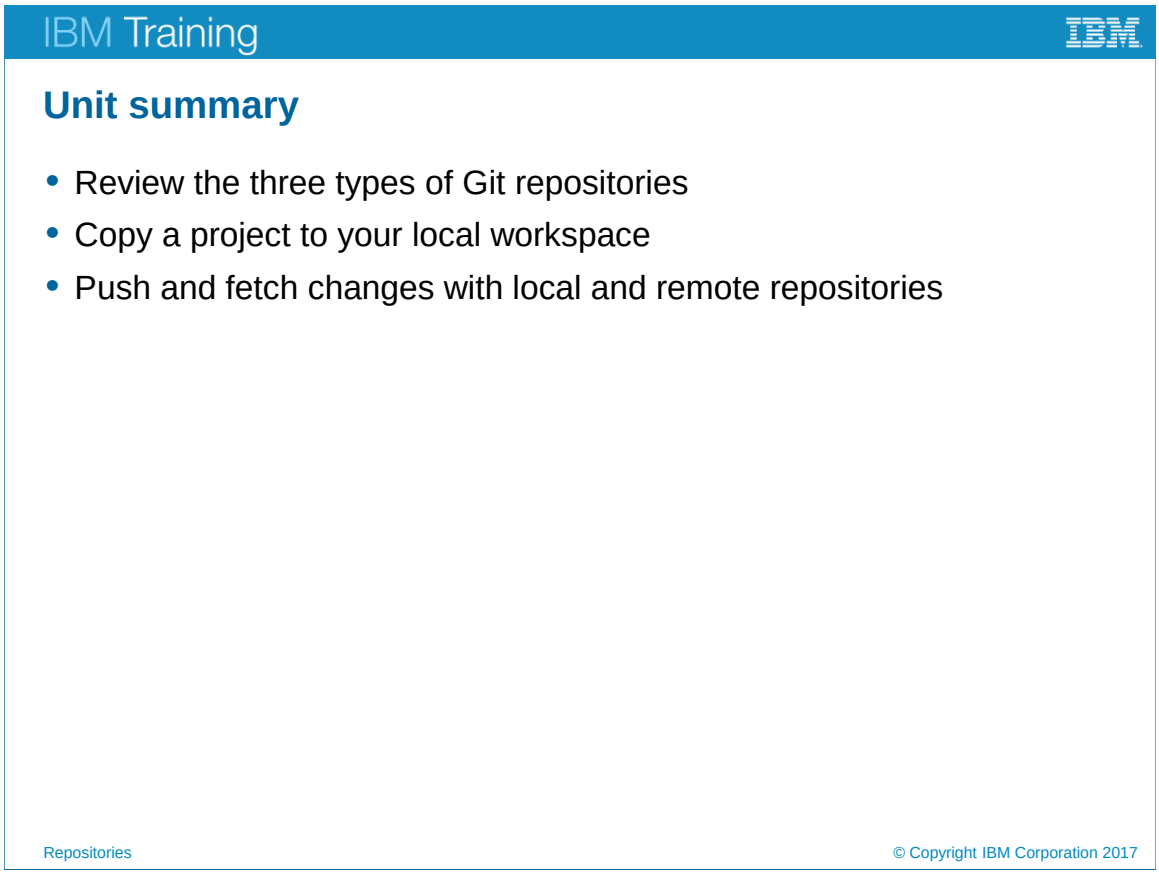
- Syncing updates your local copy of the repository



Repositories

Syncing

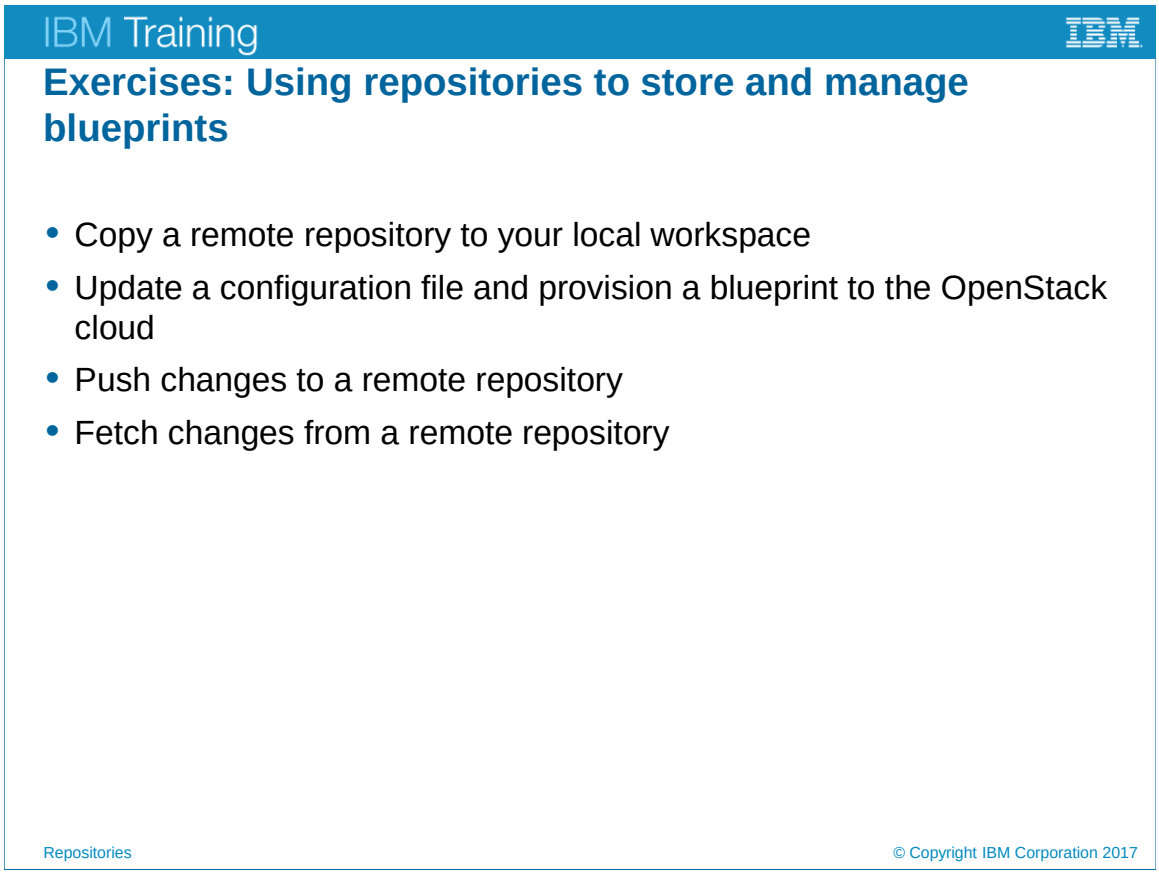
After you *sync*, the change is accepted and becomes part of the history.



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

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.

IBM Training

Exercise 1

Copy a project to your local workspace

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.

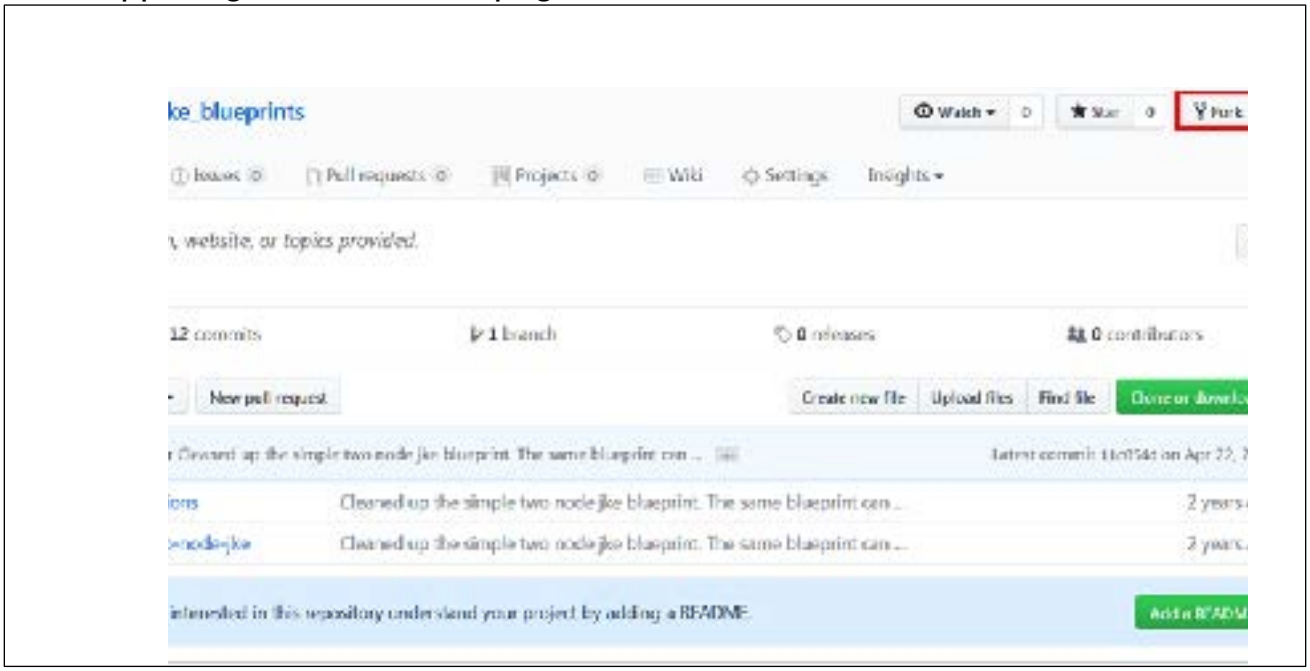
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

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**.

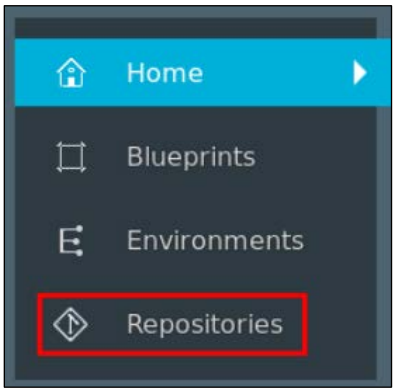


3. You now have a personal copy of the jke_blueprints project.
4. Note the URL for your repository. It is similar to https://github.com/<Your_ID/jke_blueprints.

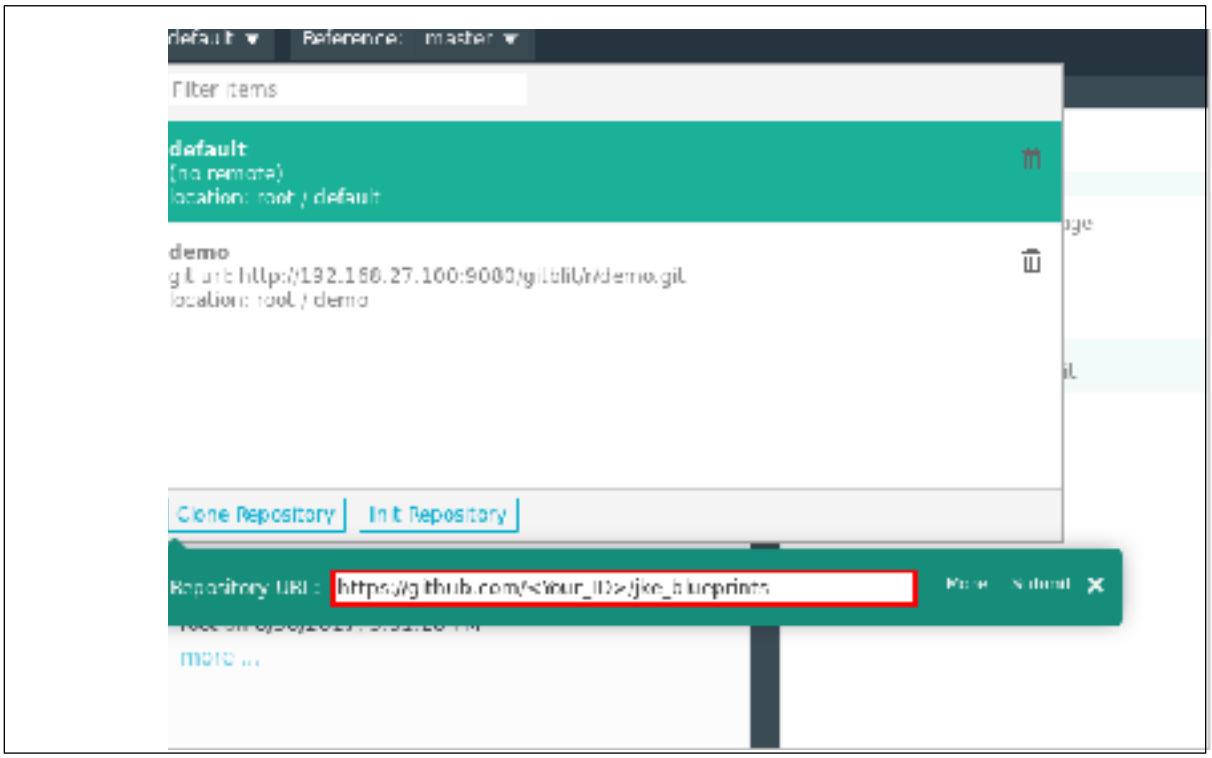
Task 2. Clone a Git repository in the Blueprint Designer.

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.

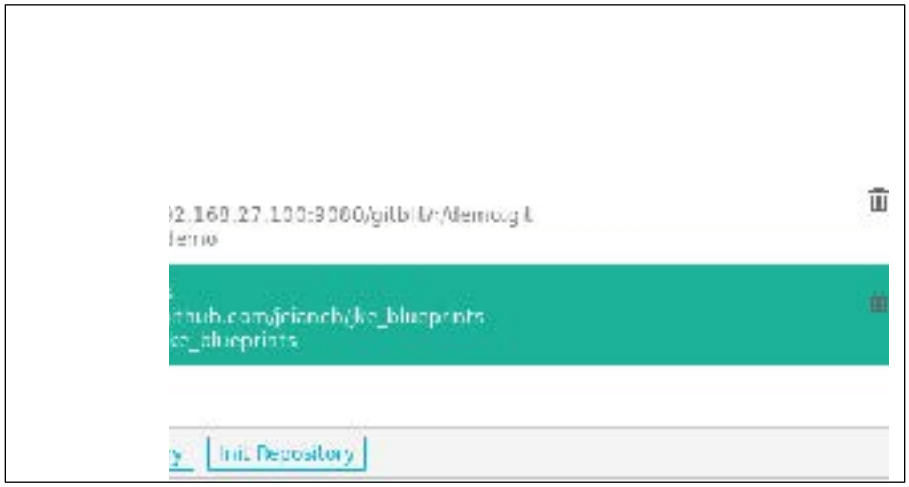
1. From the Blueprint Designer home page, click **Repositories**.



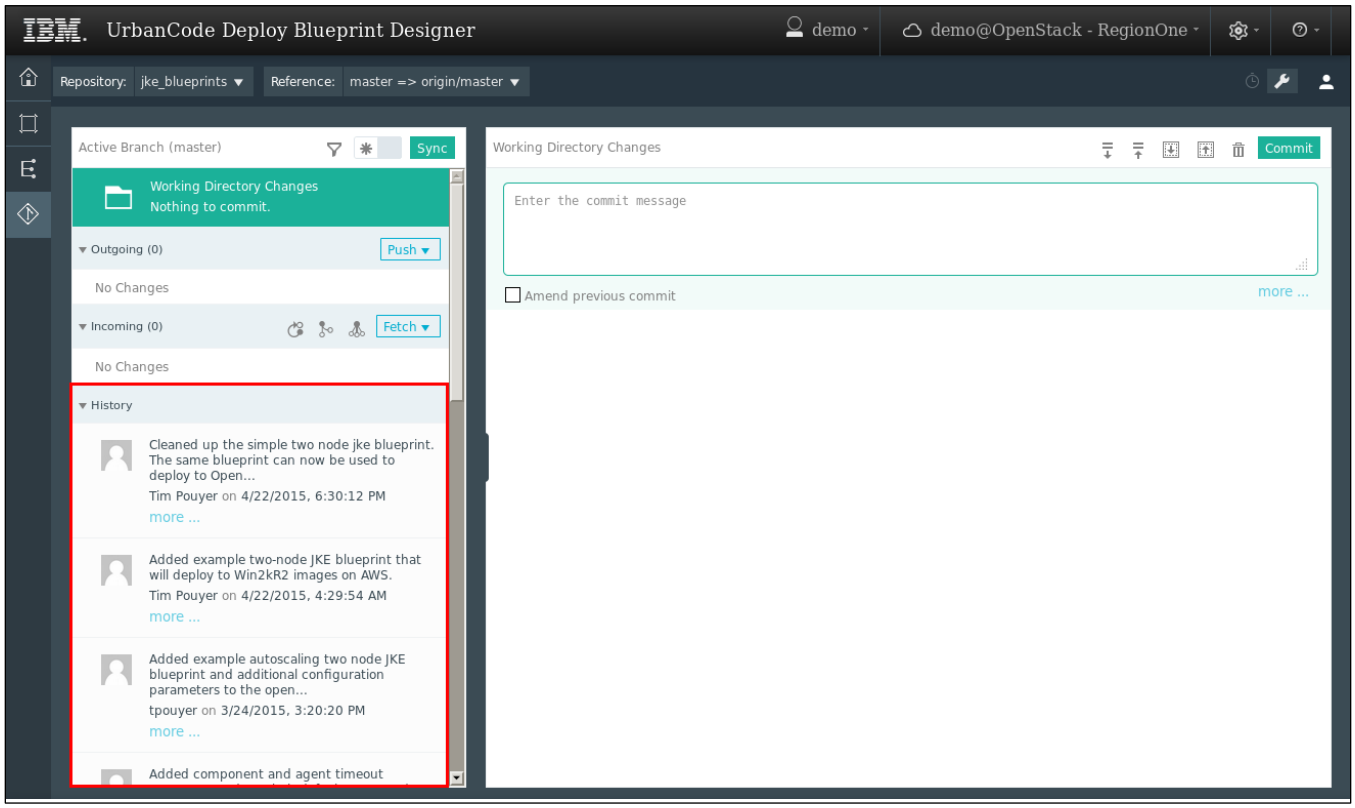
2. Next to the **Repository** field, click the drop-down arrow, and click **Clone Repository**. Currently, the default repository is open.
3. In the **Repository URL** field, type `https://github.com/<Your_ID>/jke_blueprints`, and click **Submit**.



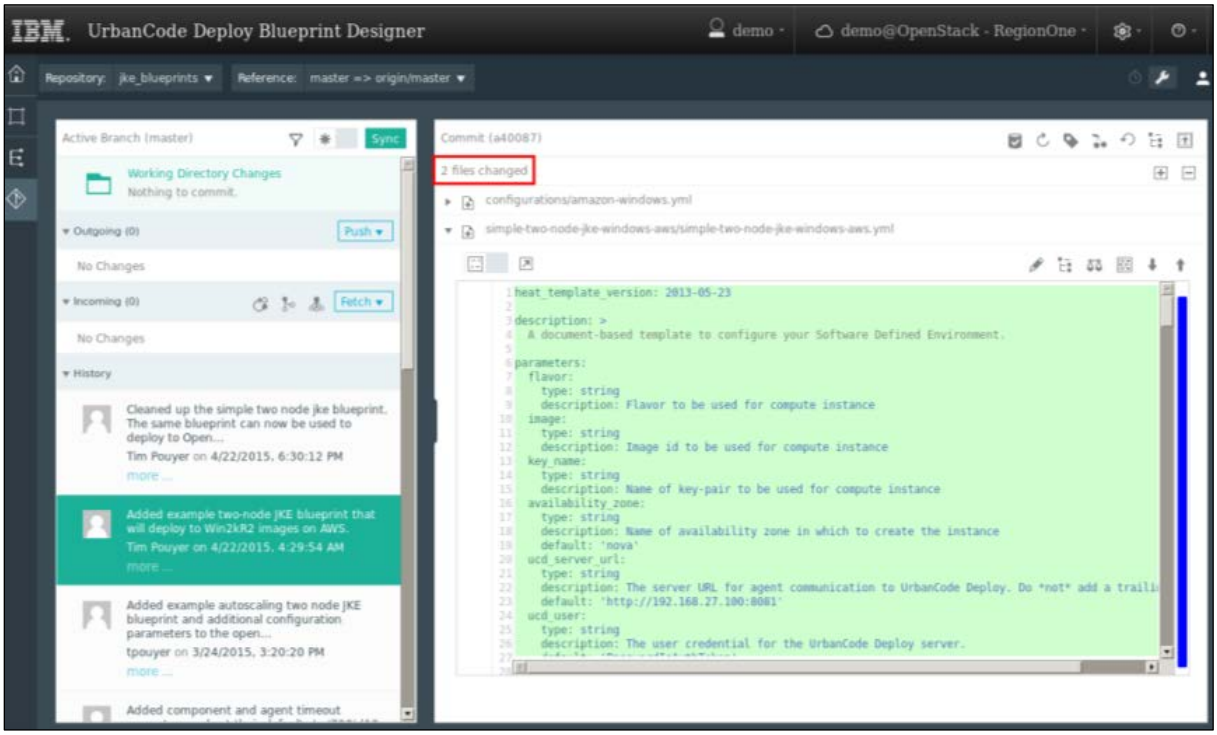
4. Click **jke_blueprints**, the cloned repository, to select and open it.



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



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



IBM Training

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.

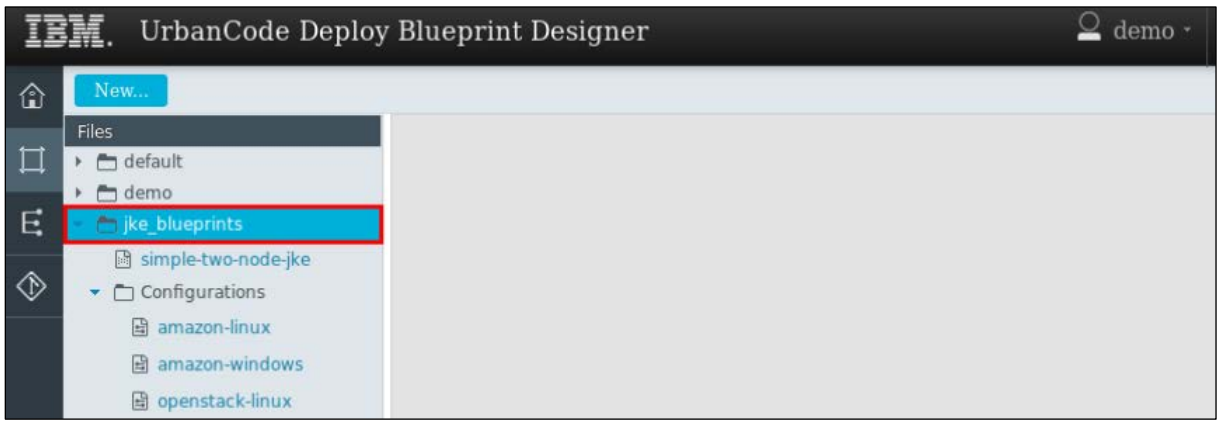
For more information about where to work and the exercise results, see the Tasks and Results section that follows.

Exercise 2:
Tasks and results

Task 1. Modify the configuration file.

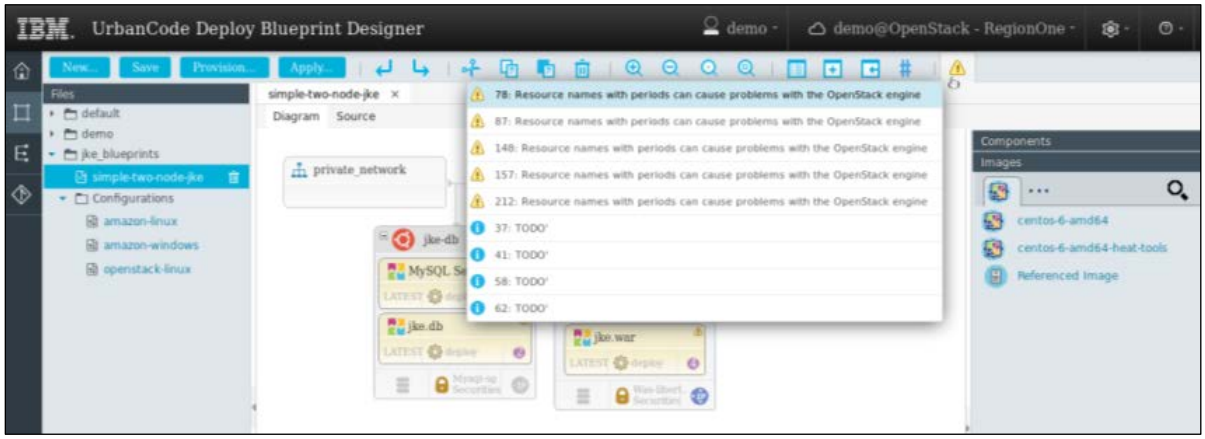
Working with your own copy of a remote project, you can modify the files to use in your cloud deployments.

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.

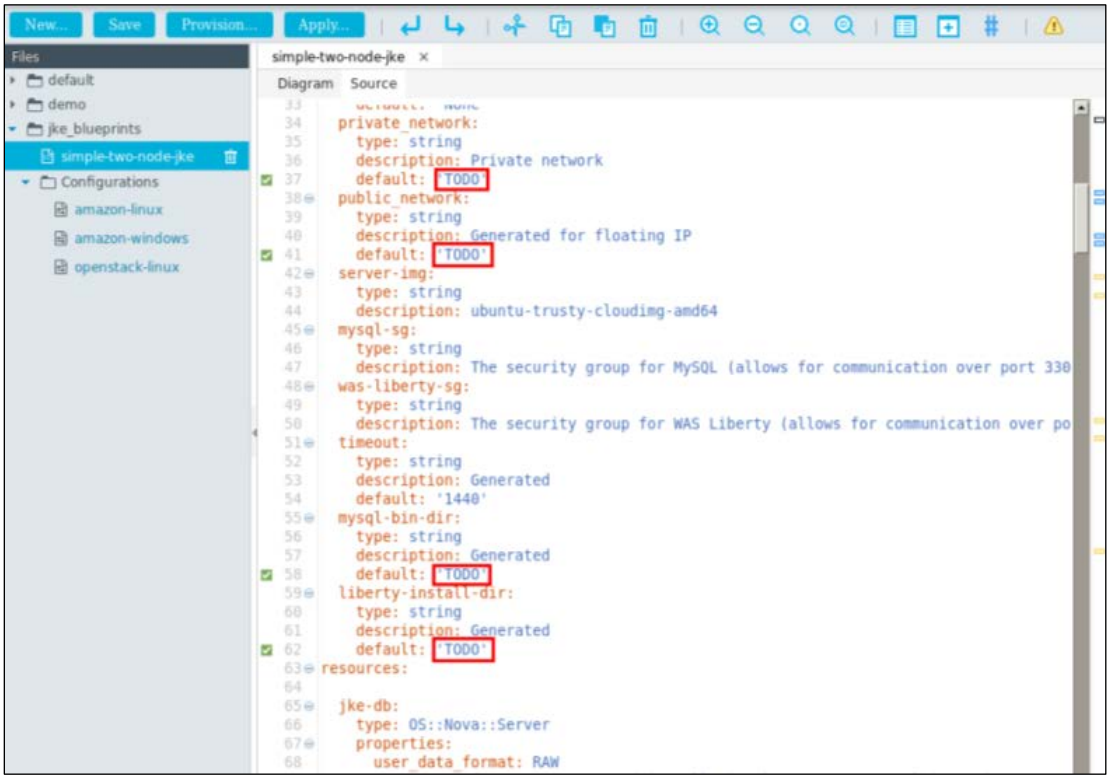


3. Click the **simple-two-node-jke** blueprint to open it.

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



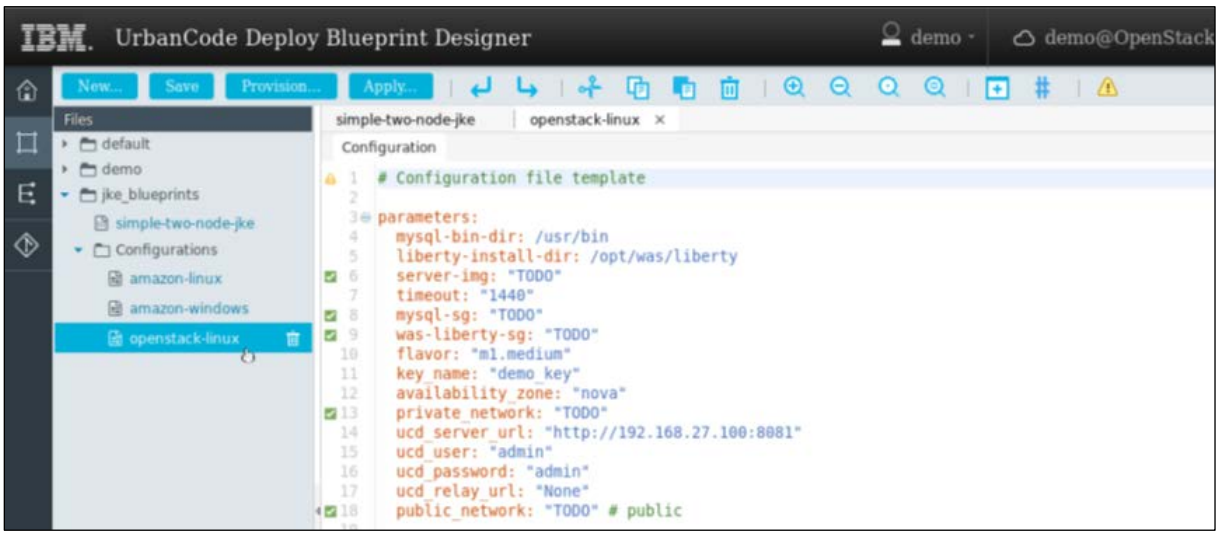
4. Click the **Source** tab to see the specific lines of code that need parameters.



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.

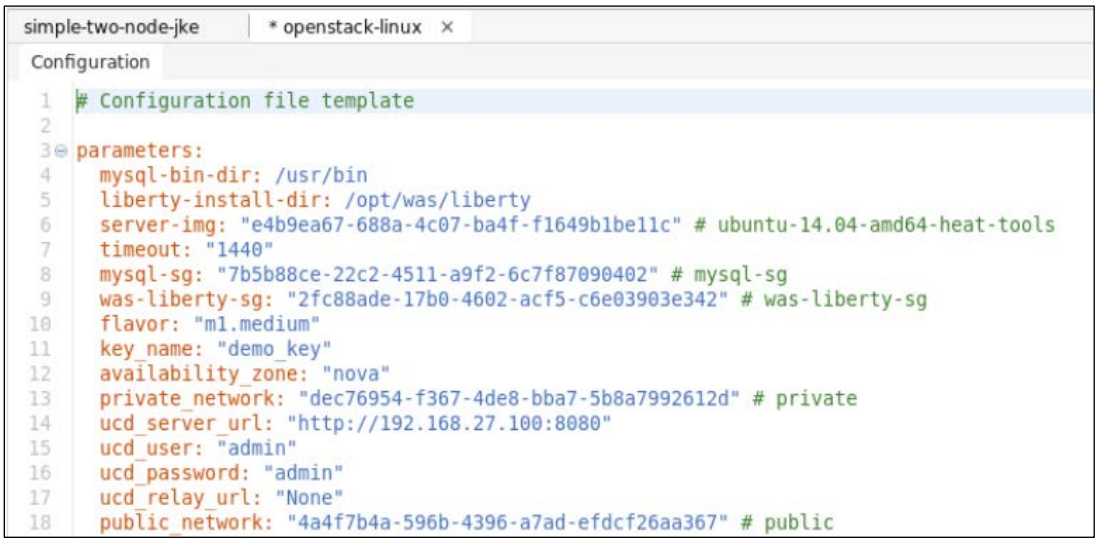


2. To import properties from the blueprint, click the **Add** icon  , select the **simple-two-node-jke** blueprint, and click **OK**.
3. In the configuration file, specify the properties for the deployment to the OpenStack cloud. To assign a value to a missing parameter, click the word

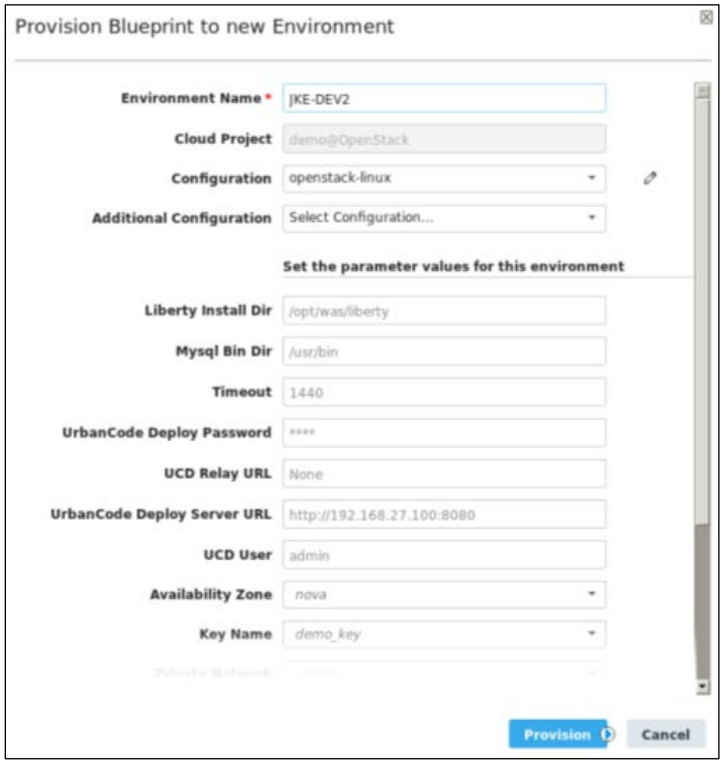
TODO to the right of a parameter, press Ctrl+Space bar at the same time, and select the following options:

- server-img: Ubuntu-14.04-amd64-heat-tools
- mysql-sg: mysql-sq
- was-liberty-sg: was-liberty-sq
- private_network: private
- public_network: public

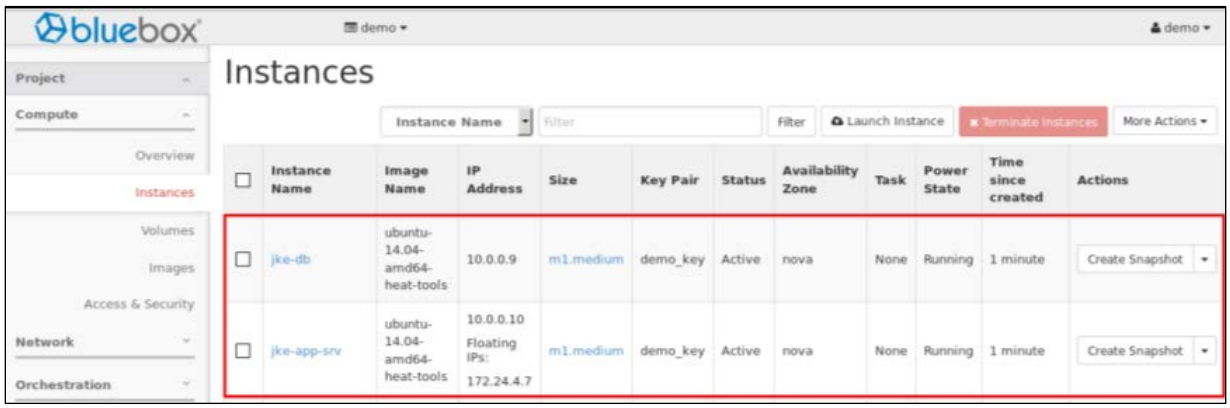
4. The IBM UrbanCode Server port number has changed since the creation of this file. In the **ucd_server_url** field, change the port number of the URL to `8080`.

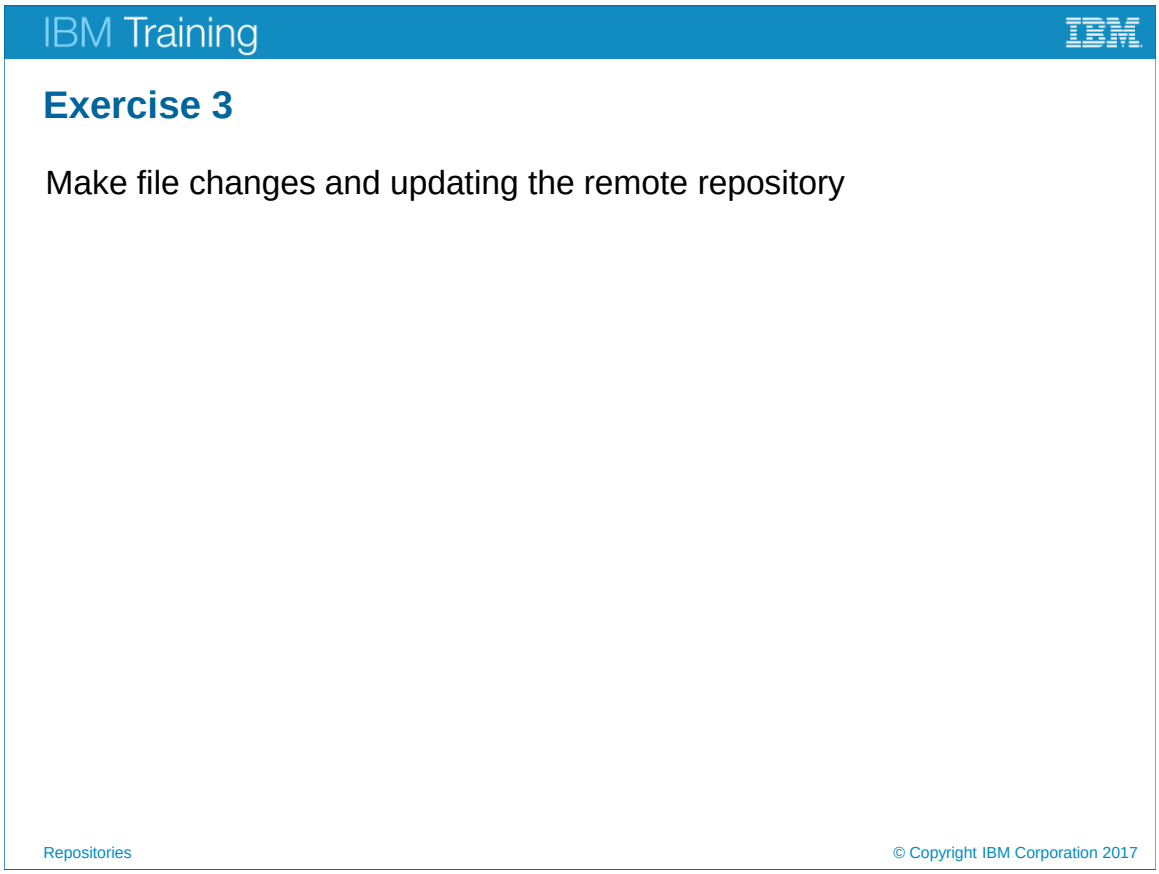


5. Save the configuration file.
6. From the **simple-two-node-jke** blueprint, click **Provision**.
7. In the “Provision Blueprint to new Environment” window, do the following steps:
   - In the **Environment Name** field, type `JKE-DEV2`.
   - In the **Configuration** field, select **openstack-linux**.



8. Click **Provision**.
9. 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:





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:

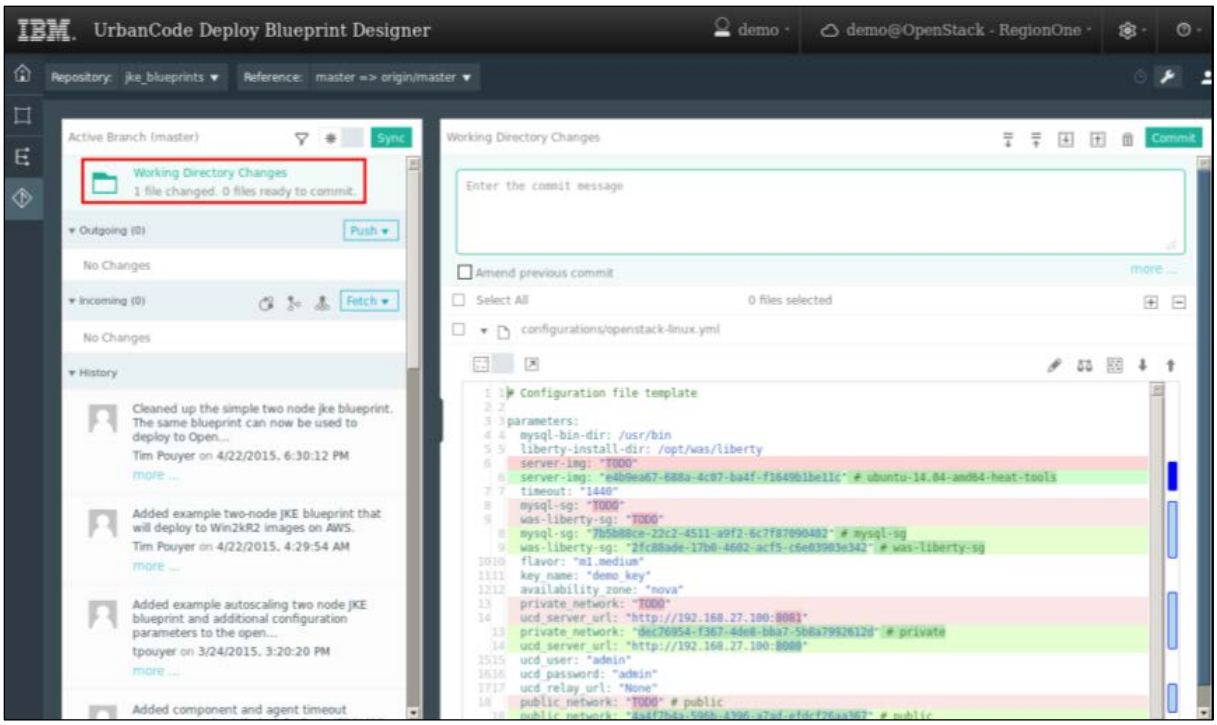
- Push changes to the remote repository.
- Verify changes in the remote repository.
- Push changes to local repositories.
- Verify changes in the local repository.

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

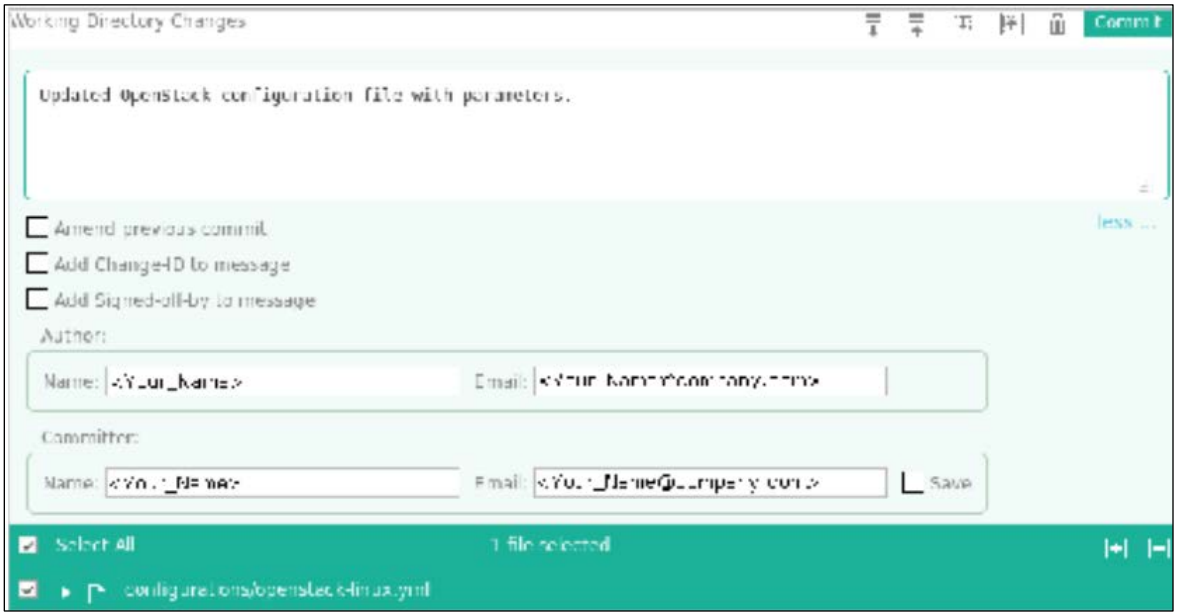
Exercise 3:
Tasks and results

Task 1. Push changes to the remote repository.

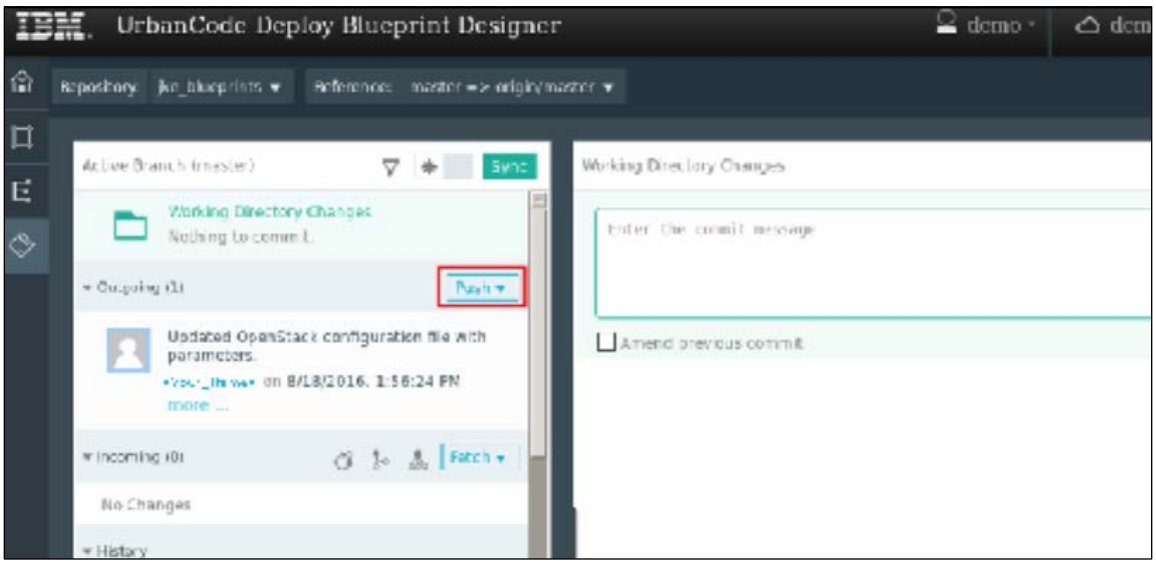
1. From the left navigation, click **Repository**. Notice the file change in your working directory.



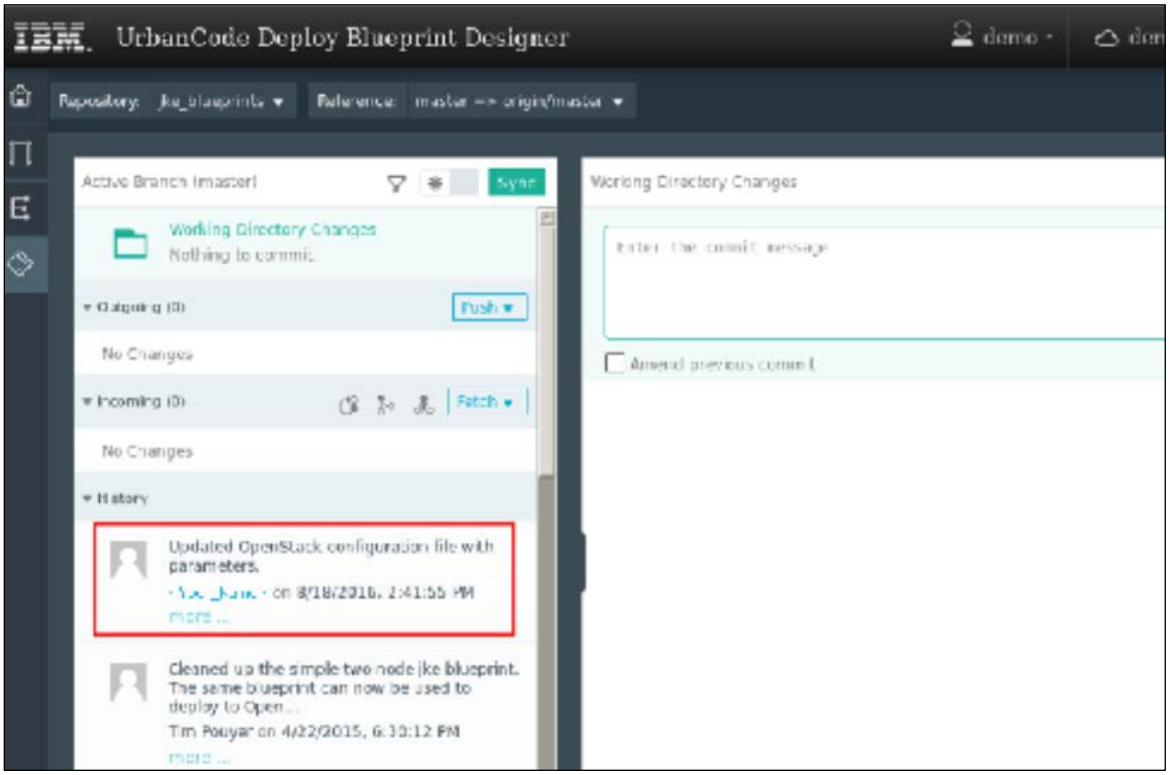
2. In the **Enter a commit message** field, type `Updated OpenStack configuration file with parameters`.
3. Under the message field, click **more**, and do the following steps:
   - In the Author section, type your name and email address.
   - In the Committer section, type your name and email address 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**.



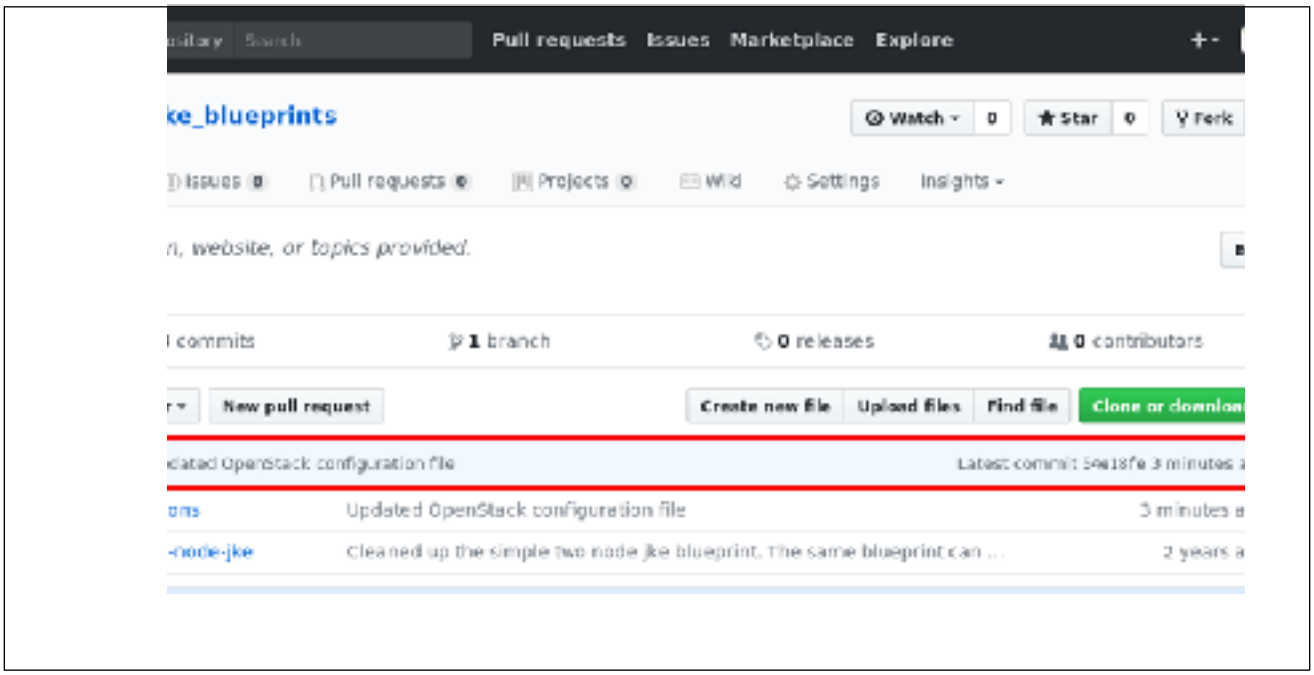
6. Type your user name and password, and click **Submit**.
   Now your change is pushed to the remote repository, and you can see the version in the History section.



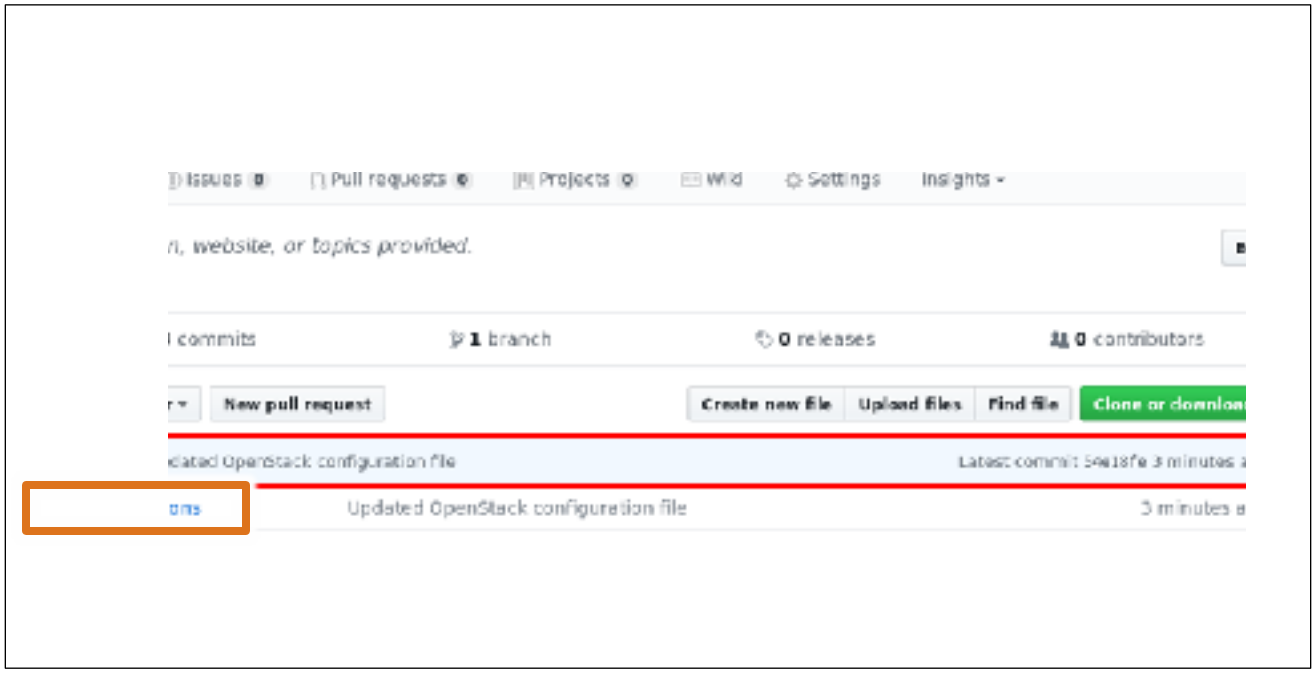
Task 2. Verify changes in the remote repository.

You can see recent project activity from your remote project.

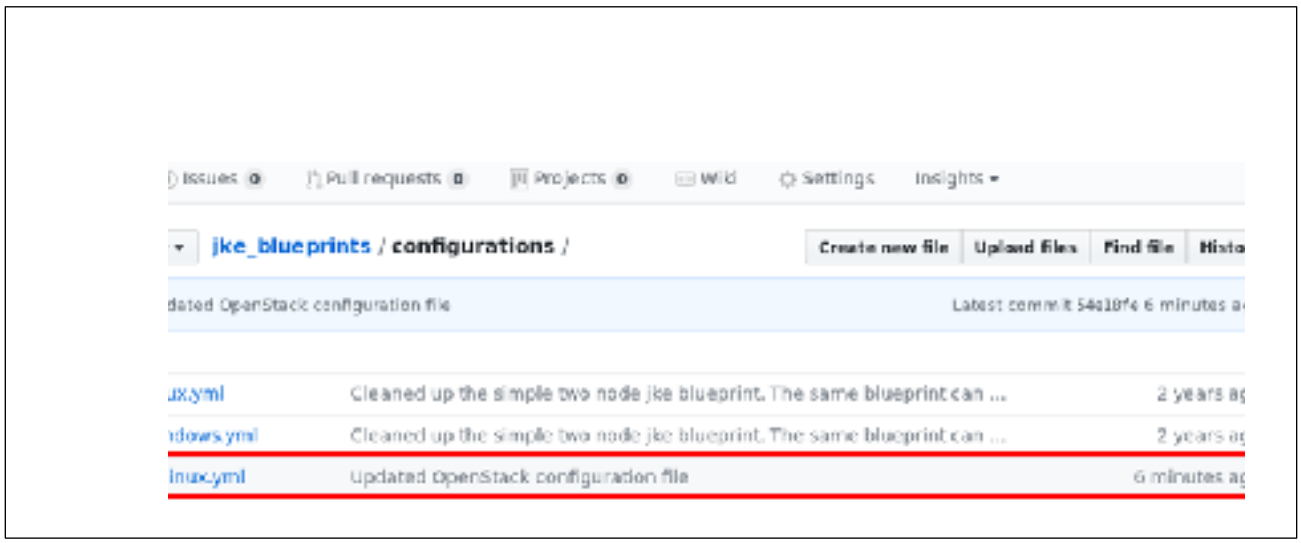
1. Switch back to the tab with your GitHub project and refresh the browser. From the **Code** tab, you see the latest changes to the project.



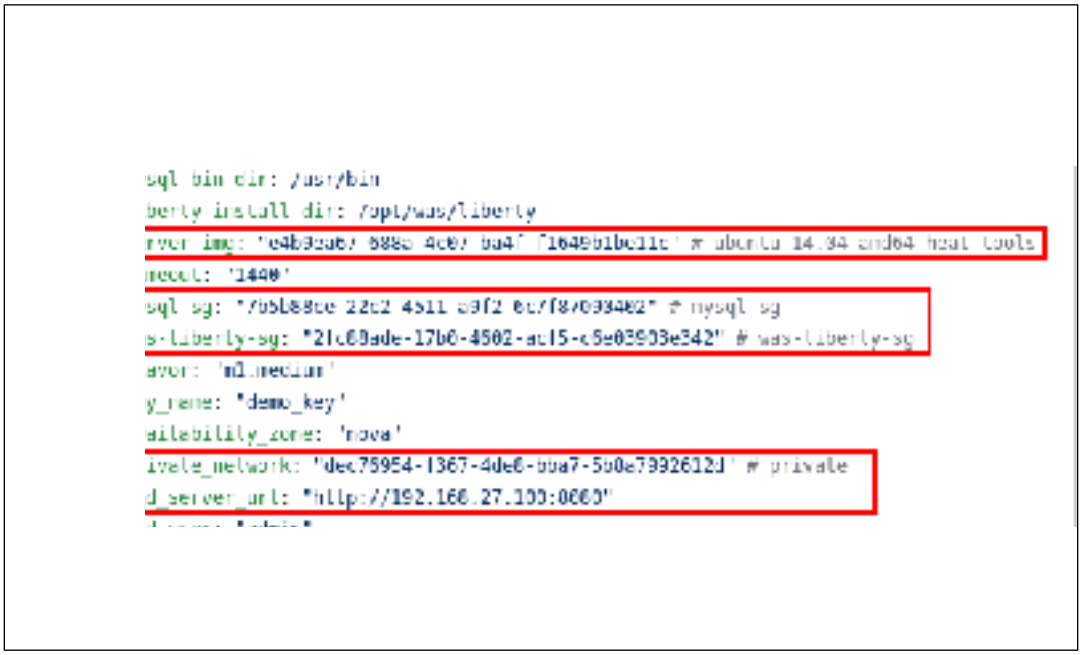
2. To see the files, click **configurations**.



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



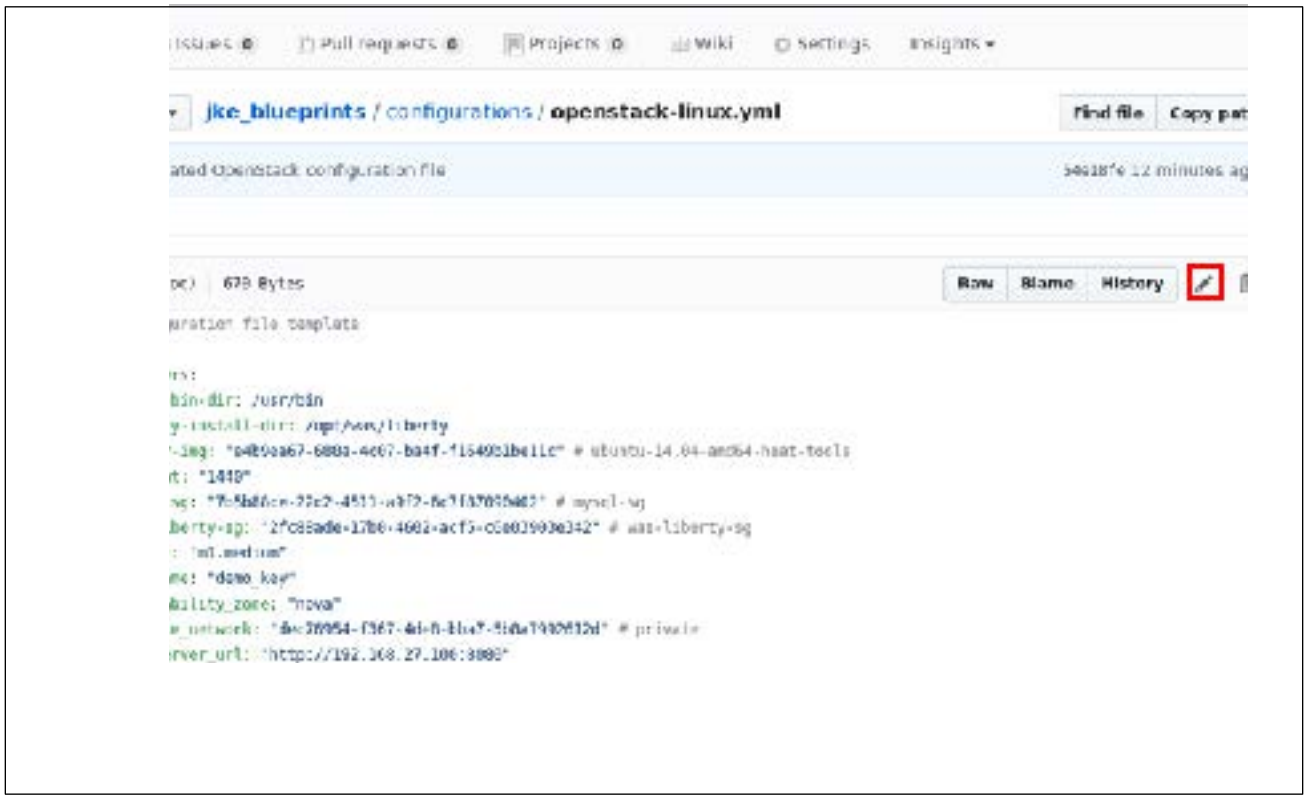
4. You can see the updated file with the substituted parameters:



Task 3. Push changes to local repositories.

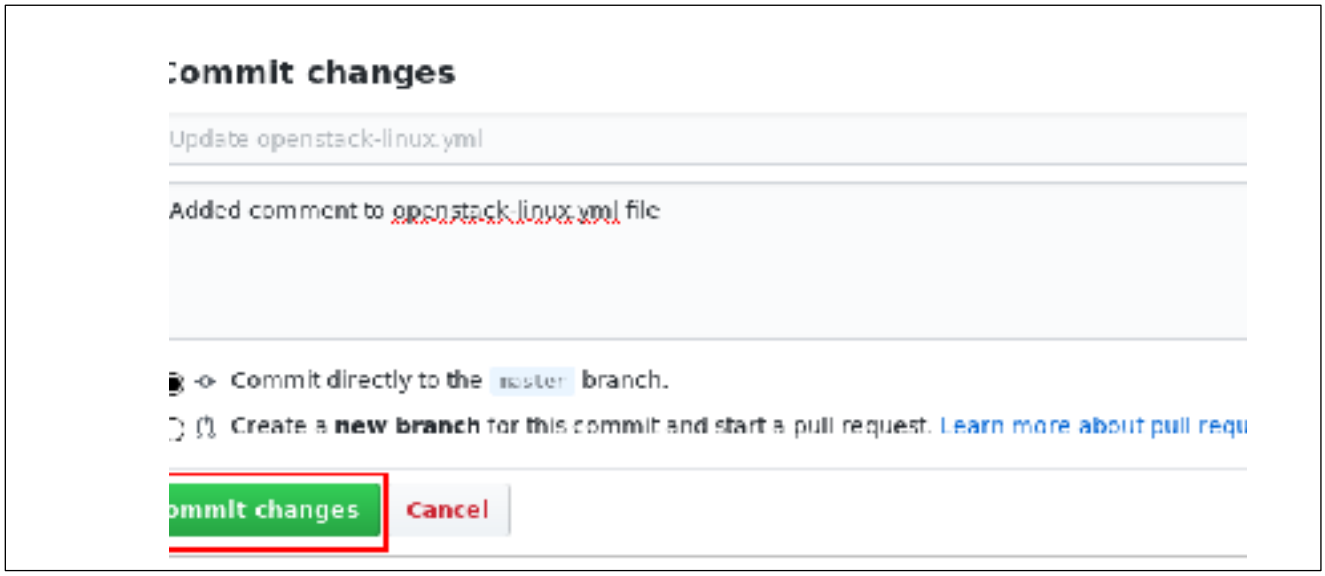
Make a change in the remote repository and push it to your local copy in the Blueprint Designer.

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



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

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.

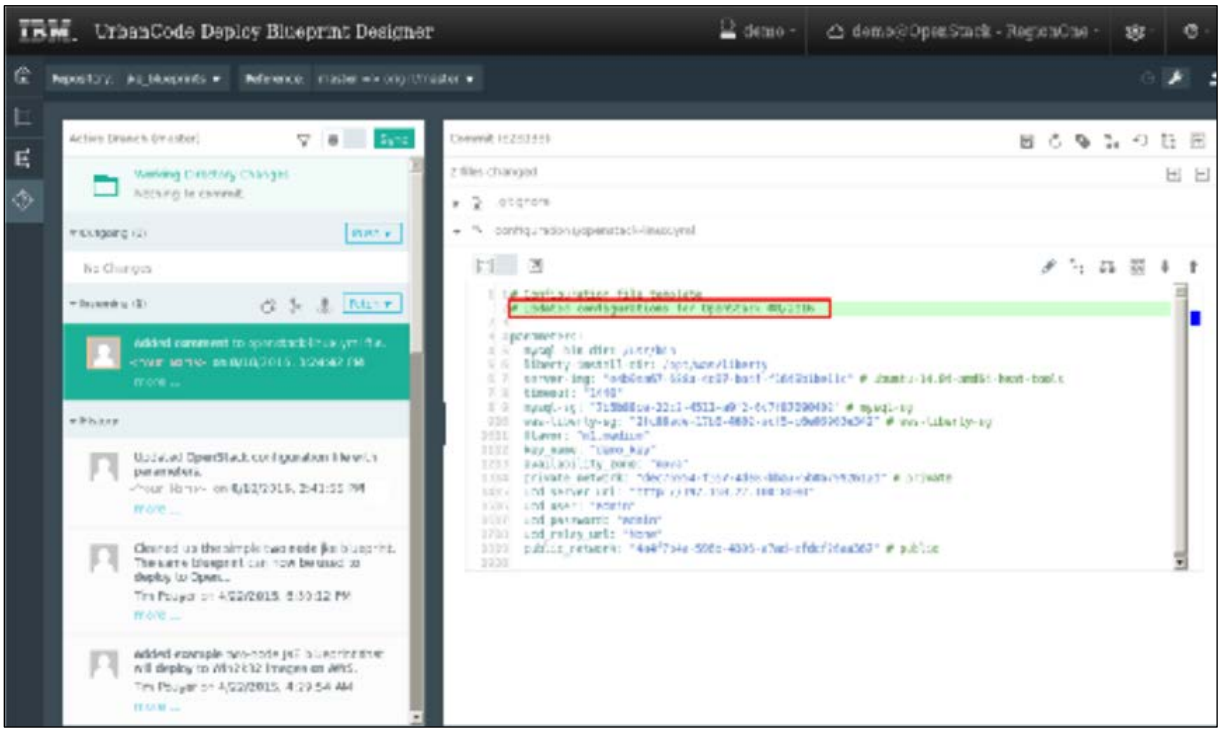


5. Click **Commit changes**.

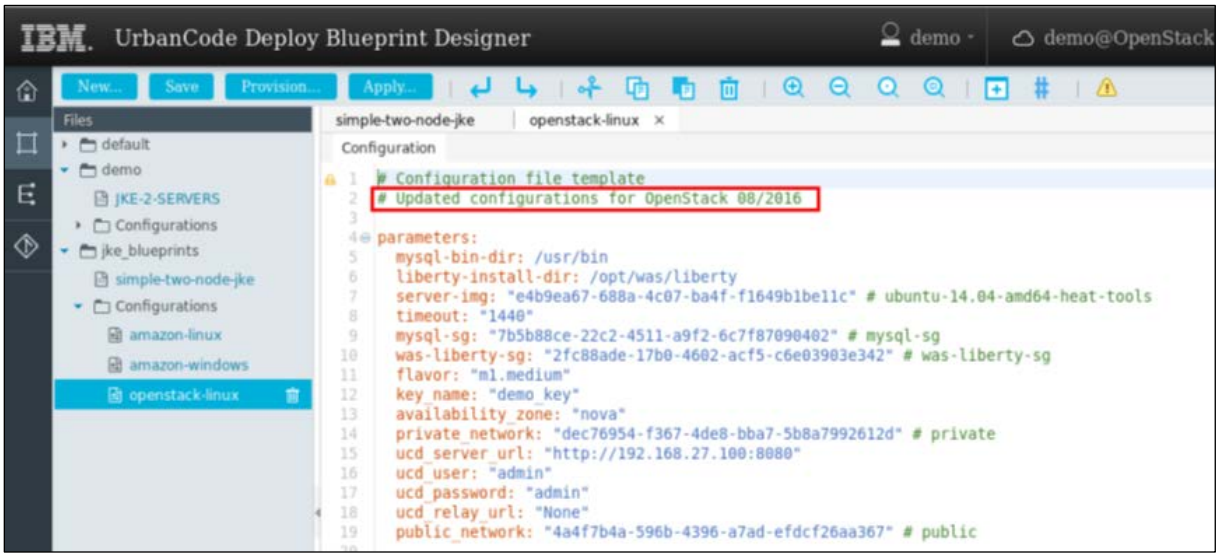
Task 4. Verify changes in the local repository.

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