

Course Guide

# Design Application Deployment with IBM UrbanCode Deploy

Course code ZQ410G ERC 1.0



## June 2017 edition

### 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](http://www.ibm.com/legal/copytrade.shtml).

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

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

**© Copyright International Business Machines Corporation 2017.**

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

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

---

# Contents

<b>About this course .....</b>	<b>vii</b>
Course objectives .....	viii
Audience .....	viii
Prerequisites .....	viii
Course agenda .....	ix
<b>Unit 1 Designing software deployment and deploying applications .....</b>	<b>1-1</b>
Unit objectives .....	1-2
Topic: Accelerating software delivery and improving quality with IBM DevOps .....	1-3
A deployment is the promotion of an application from one environment to the next .....	1-4
Deployment problems are serious business .....	1-5
Agile development highlights bottlenecks .....	1-6
IBM DevOps promotes continuous delivery of software through the entire pipeline .....	1-7
IBM DevOps strategies results in a smooth flow of the delivery pipeline from development to production .....	1-8
IBM DevOps is working for organizations across industries .....	1-9
IBM UrbanCode products support the continuous delivery pipeline .....	1-10
IBM UrbanCode Deploy supports the DevOps philosophy .....	1-11
Topic: Defining the key concepts and workflow in IBM UrbanCode Deploy .....	1-12
A modular approach permits the division of responsibilities in DevOps organizations .....	1-13
Application models include components, processes, and environments .....	1-14
A component is the smallest functional unit of an application .....	1-15
Processes are lists of steps and connections between those steps .....	1-16
An environment is a collection of resources that hosts an application .....	1-17
The objects for modeling software deployments work together .....	1-18
The core installation of IBM UrbanCode Deploy includes a server, agents, and a license server .....	1-20
A basic workflow consists of some typical steps .....	1-21
Unit summary .....	1-22
<b>Unit 2 Configuring components and component processes .....</b>	<b>2-1</b>
Unit objectives .....	2-2
Topic: Component overview .....	2-3
A component is the smallest functional unit of an application .....	2-4
Components describe what gets deployed .....	2-5
Each component represents artifacts from a single source .....	2-6
A component contains versioned files, component processes, and properties .....	2-7
Topic: Creating components and importing component artifacts .....	2-8
Component artifacts can come from various source-code management sources .....	2-9
Components have versions that ensure the appropriate component instances are deployed .....	2-11
The agent is typically on the same computer where the artifacts are located .....	2-12

A new version of the component is created every time a component's artifacts are modified and imported .....	2-13
The version import history shows the status of attempted imports .....	2-14
A version contains a set of deployable artifacts .....	2-15
Change logs provide information about modifications to components .....	2-16
Topic: Creating processes to deploy components .....	2-17
Component processes deploy, configure, or uninstall component versions .....	2-18
Components contain processes that perform operations on component files .....	2-19
Component processes are created with the process editor .....	2-20
Process steps are selected from a menu of standard steps that replace typical deployment scripts and manual processes .....	2-21
A number of plug-ins come with the product and many others are available that can be downloaded and installed .....	2-22
The palette provides automation steps that you can drag onto the process editor .....	2-23
Processes are versioned artifacts .....	2-24
The Changes tab tracks the creation date and author of the versions .....	2-25
Processes use different kinds of steps and joining techniques .....	2-26
Conditional processes use flags to set conditions on connections .....	2-27
Switch steps branch processes based on the value of a property .....	2-28
Branching steps run multiple steps at the same time .....	2-29
Topic: Storing information with properties .....	2-30
Properties are variables that store information about different elements .....	2-31
Components can have several types of properties .....	2-32
Properties are referred to by scope .....	2-34
Properties can also be referred to without scope .....	2-35
Property scope determines the precedence of properties with the same name .....	2-36
Properties are primarily set from the Configuration tab .....	2-37
Unit summary .....	2-38
Exercises: Configuring components and component processes .....	2-39
<b>Unit 3 Defining deployments with resources, environments, and application processes.....</b>	<b>3-1</b>
Unit objectives .....	3-2
Topic: Managing components in applications .....	3-3
Applications associate resources with environments and define processes to run deployments .....	3-4
An application groups components together .....	3-5
Applications manage components and map the dependencies between them .....	3-6
Topic: Configuring the agent as a deployment-target host .....	3-7
Applications determine the what, where, and how of a deployment .....	3-8
An environment associates components with an agent on the target host .....	3-9
An agent is a lightweight process that usually runs on a deployment-target host .....	3-10
Agents are installed from the user interface or the command prompt .....	3-11
Agents open direct connections to the server .....	3-12
An agent consists of a worker process and a monitor process .....	3-13
Many agent features are managed from IBM UrbanCode Deploy .....	3-14
Agent pools manage agents that are installed in different environments .....	3-15
Agent relays coordinate communication between agents and the server .....	3-16
Topic: Organizing resources in a resource tree .....	3-17
A resource tree organizes what is being deployed and to where .....	3-18

Resource groups enable collections of resources to be easily reused . . . . .	3-19
Agent-type resources represent the computers where components are deployed . . . . .	3-20
Component-type resources represent the components that are deployed to target environments . . . . .	3-21
The Resource Tree page displays resources in a hierarchical tree structure . . . . .	3-22
Topic: Setting up deployment targets with environments . . . . .	3-23
Environments can have different topologies, configurations, and settings . . . . .	3-24
Environments represent the target to which your files are being deployed . . . . .	3-25
Mapping resources and components to the environment provides a topology . . . . .	3-26
Resources associate agents with components . . . . .	3-27
Topic: Coordinating deployments with application processes . . . . .	3-28
An application process orchestrates the order of component process execution . . . . .	3-29
The application process is the how of the application model . . . . .	3-30
An application process determines what components are deployed together . . . . .	3-31
Application processes are created with the process editor . . . . .	3-32
Application processes provide overall deployment orchestration . . . . .	3-33
With all of the application pieces in place, you are ready for deployment . . . . .	3-34
Unit summary . . . . .	3-35
Exercises: Defining deployments with resources, environments, and application processes . . . . .	3-36

<b>Unit 4 Deploying applications to target environments . . . . .</b>	<b>4-1</b>
Unit objectives . . . . .	4-2
Topic: Deployment to an environment . . . . .	4-3
A deployment is the promotion of an application from one environment to the next . . . . .	4-4
A deployment configures and installs a deployment package in a specific environment . . . . .	4-5
You run a deployment by running an application process in a target environment . . . . .	4-6
The application process request shows each step in the process . . . . .	4-7
The Execution section displays the deployment details . . . . .	4-8
Topic: Examining inventory and deployment information . . . . .	4-9
Inventory and history are retained for environments and resources . . . . .	4-10
You can compare inventories between environments . . . . .	4-11
File differences are tracked between the environments . . . . .	4-12
The History tab show the process requests for a specific environment . . . . .	4-13
The Usage tab shows the environment inventory for each component . . . . .	4-14
Deployment reports contain historical information about deployments . . . . .	4-15
Topic: Deploying applications with snapshots . . . . .	4-16
A snapshot is collection of specific versions of components and processes . . . . .	4-17
Snapshots ensure consistent deployments from one environment to another . . . . .	4-18
A snapshot captures a picture of the application's current state . . . . .	4-19
Previewing deployments for impact analysis . . . . .	4-20
Deploying a snapshot deploys the components in the snapshot . . . . .	4-21
Application and component processes are the same . . . . .	4-22
The inventory is updated after running the snapshot . . . . .	4-23
Environments can be configured to lock snapshot versions or configuration . . . . .	4-24
You can lock a snapshot at environment creation . . . . .	4-25
Unit summary . . . . .	4-26
Exercises: Deploying applications to target environments . . . . .	4-27

<b>Unit 5 Setting up server security, approvals, and quality gates.....</b>	<b>5-1</b>
Unit objectives .....	5-2
Topic: Configuring server security .....	5-3
The security system for the server consists of an authentication realm, authorization realm, roles, and teams .....	5-4
You can access the security system from any of these links on the Settings tab .....	5-5
Authorization realms manage user groups .....	5-6
An authentication realm is a source of information about users .....	5-7
A role is a set of permissions .....	5-8
Groups identify users with the same role .....	5-9
Keep the roles simple but ensure sufficiency for performing work .....	5-10
Groups grant permissions to multiple users .....	5-11
Users are granted permissions by being assigned to teams .....	5-12
Restrict permissions by assigning teams to objects .....	5-13
Security reports provide information about user roles and privileges .....	5-14
Topic: Setting up notifications .....	5-15
Notifications can be sent when user-defined trigger events occur .....	5-16
IBM UrbanCode Deploy requires an external SMTP mail server to send notifications .....	5-17
Notifications are sent to appropriate parties based on scheme .....	5-18
Notifications include triggering events and roles .....	5-19
Notifications are sent to appropriate parties based on scheme .....	5-20
Topic: Setting up an approval process .....	5-21
Approvals provide more visibility into deployments for audit trails .....	5-22
An approval process specifies the job that needs approval and the role of the approver .....	5-23
When a request for approval is made, the users with the corresponding role are notified through email .....	5-24
The Deployment Details report tracks approval status .....	5-25
Topic: Implementing environment gates to improve quality .....	5-26
Gates specify the conditions that must be met for deployment .....	5-27
Using statuses and gates requires initial configuration .....	5-28
Version statuses identify when component versions meet criteria .....	5-29
Environment gates check the version tags on component versions .....	5-30
Statuses can be added to versions automatically from component processes .....	5-31
Adding version statuses to component versions becomes part of the process .....	5-32
Version statuses are tracked in each component .....	5-33
Select the latest components with a status .....	5-34
Unit summary .....	5-35
Exercises: Setting up server security .....	5-36

# About this course

IBM Training



IBM

**Design Application Deployment with IBM UrbanCode Deploy**

Course code ZQ410 ERC 1.0

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

IBM UrbanCode Deploy is a tool for standardizing and simplifying the process of deploying software components to each environment in your development cycle.

This is a beginner-level course for users of IBM UrbanCode Deploy, such as administrators, development teams, and operations leads. In this course, you learn how to plan and automate the deployment of applications to test environments. Hands-on labs use IBM UrbanCode Deploy and cover deploying a simple web application. You create components, create an application that contains those components, and then deploy the components to an environment.

The lab image for course ZQ410 is hosted on Skytap/IRLP. There is no image preparation required.

For information about other related courses, visit the IBM Training website:

<http://www.ibm.com/training>

Details	
<b>Delivery method</b>	Classroom or instructor-led online (ILO)
<b>Course level</b>	ERC 1.0. This course is an updated course.
<b>Product and version</b>	IBM UrbanCode Deploy 6.2.2
<b>Recommended duration</b>	1 day
<b>Skill level</b>	Beginner

IBM Training



## Course objectives

- Describe how IBM UrbanCode Deploy supports the principles and practices of DevOps
- List the main concepts and features of IBM UrbanCode Deploy
- Plan and automate the deployment of applications to test environments
- Configure a security model with roles, approvals, and quality statuses and gates

# Audience

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

# Prerequisites

Before you take this course, you should have a basic understanding of client-server architecture and multitier application models.

## Course agenda

- Unit 1: Designing software deployment and deploying applications
- Unit 2: Configuring components and component processes
- Unit 3: Defining deployment with resources, environments, and application processes
- Unit 4: Deploying applications to target environments
- Unit 5: Setting up server security, approvals, and quality gates

### Course agenda

The course contains the following units:

1. [Designing software deployment and deploying applications](#)

IBM UrbanCode Deploy is a deployment automation tool that standardizes and simplifies the process of deploying software components to each environment in the development cycle. In this unit, you learn about typical deployment problems and key deployment elements and concepts.

2. [Configuring components and component processes](#)

Components represent deployable items along with user-defined processes that operate on them, usually by deploying them. In this unit, you learn how to create a component, import component versions, create component processes, and work with properties.

3. [Defining deployments with resources, environments, and application processes](#)

Applications identify the component versions that must be deployed together. They also define the different environments that the components must go through on the way to production. In this unit, you learn to create an application and add components, create environments and add resources, and create application processes.

4. [Deploying applications to target environments](#)

You run a deployment by running an application process in a target environment. In this unit, you learn how to deploy to an environment, create and deploy snapshots, and compare environments to view the inventory in each one.

5. [Setting up server security, approvals, and quality gates](#)

IBM UrbanCode Deploy uses a flexible team-based and role-based security model that maps to your organizational structure. In this unit, you learn how to define and configure roles, set up approvals and notifications, and use quality statuses and gates.

# **Unit 1 Designing software deployment and deploying applications**

IBM Training



## **Designing software deployment and deploying applications**

[Unit 1: Designing software deployment and deploying applications](#)

© Copyright IBM Corporation 2017

IBM UrbanCode Deploy is a deployment automation tool that standardizes and simplifies the process of deploying software components to each environment in the development cycle. In this unit, you learn about typical deployment problems and key deployment elements and concepts.

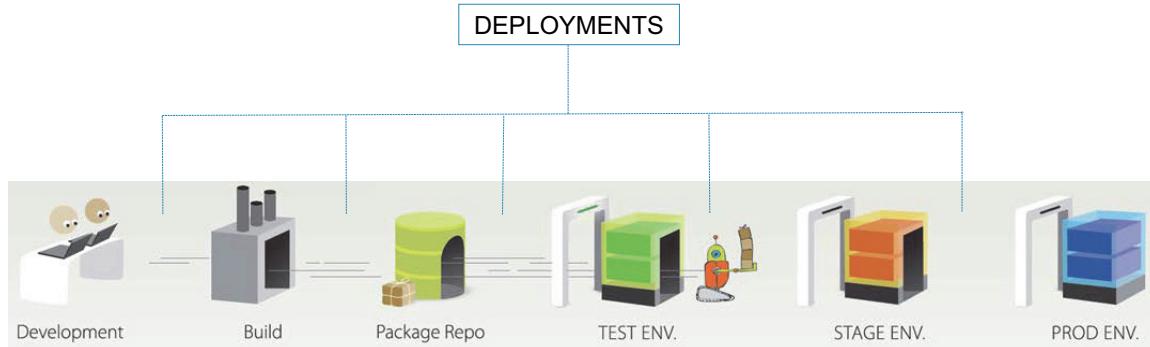
## Unit objectives

- Describe typical deployment problems
- Explain the benefits of a DevOps culture
- Define model-driven deployment
- Define key concepts and terms
- Describe the workflow for a basic project

## Topics

- ▶ Accelerating software delivery and improving quality with IBM DevOps
- Defining key concepts and workflow in IBM UrbanCode Deploy

## A deployment is the promotion of an application from one environment to the next



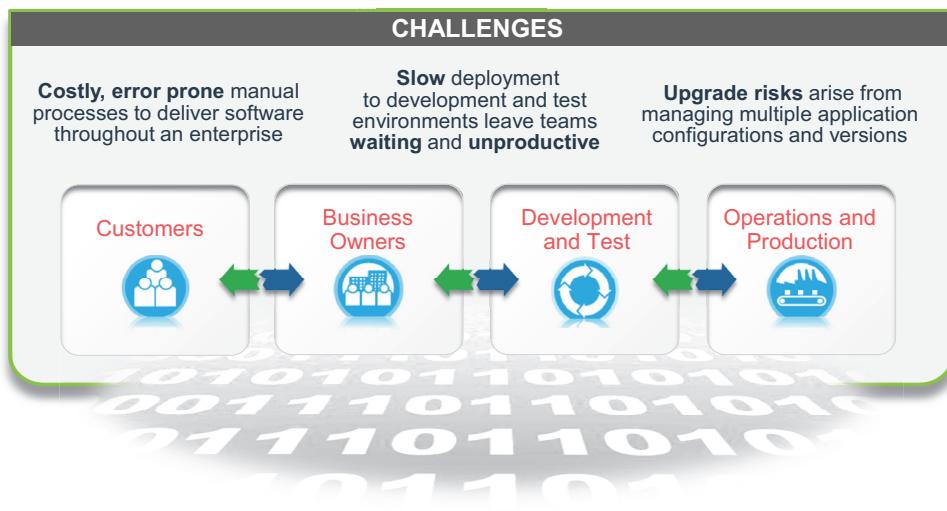
*A deployment is the promotion of an application from one environment to the next*

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 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.

## Deployment problems are serious business



Unit 1: Designing software deployment and deploying applications

© Copyright IBM Corporation 2017

### Deployment problems are serious business

Some typical deployment challenges include these examples:

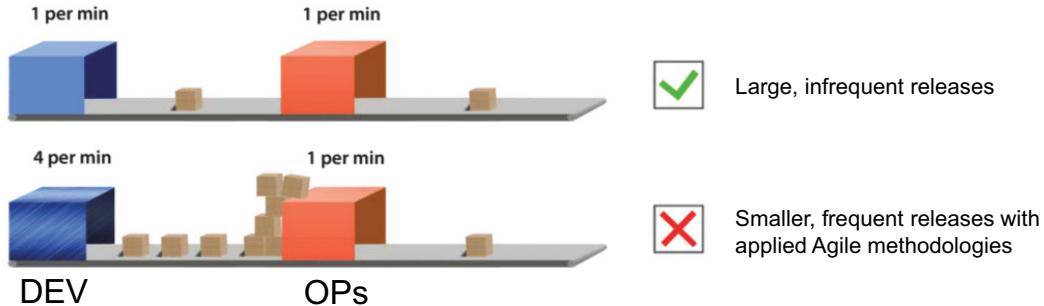
- Differences in development and operations environments and procedures cause failures
- Manual processes for deployments are error prone and lack repeatability and speed
- Major releases take days; hundreds of people are organized by spreadsheet
- No process that alerts team members to errors
- Status can only be obtained through many different systems

Deployment problems can cause an organization both financial loss and damage to its reputation.

A few examples include the following cases:

- Knight Capital loses \$440 million because of a misconfigured release (in only 45 minutes).
- New Zealand's biggest phone company, Telecom, paid \$2.7 million to approximately 47,000 customers who were overcharged after a software glitch.
- A bad software upgrade at RBS Bank left millions unable to access money for 4 days.

## Agile development highlights bottlenecks



### Agile development highlights bottlenecks

Agile methodologies embraced *continuous integration*, a practice in which software developers frequently integrate their work with that of other members of the development team. This process helped development teams to produce smaller releases faster, getting new features developed quickly, but operations teams did not have enough time to ensure stability through quality control and testing, which caused a development backlog.

Test and operations teams had increased pressures to keep up with the increased loads but continued to use traditional tools.

## IBM DevOps promotes continuous delivery of software through the entire pipeline

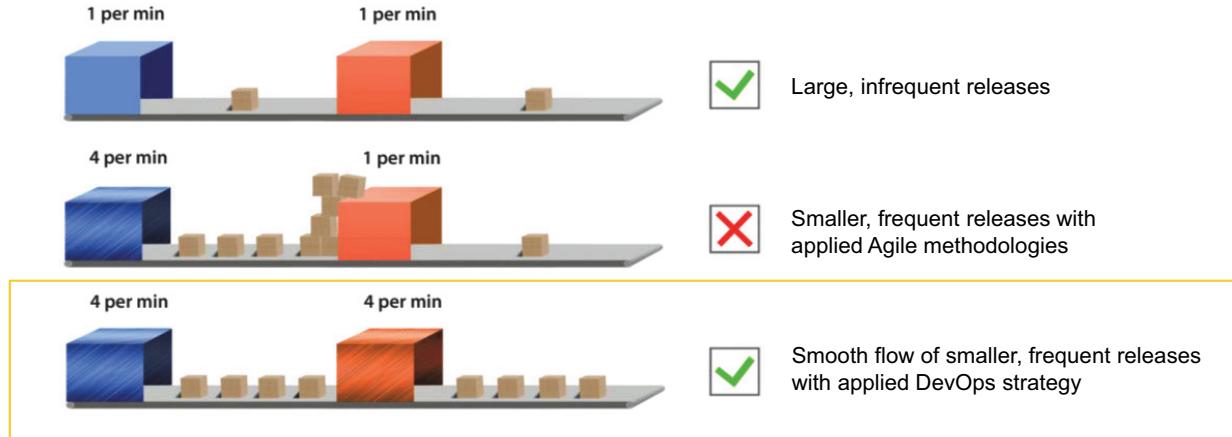


IBM DevOps promotes continuous delivery of software through the entire pipeline

DevOps came into the picture to alleviate some of the pressures between development and operations teams. In DevOps, operations and development engineers participate together in the entire software development lifecycle, from design through the development process to production support. It eliminates the barriers between developing software and production. IBM DevOps supports the philosophy of lean and Agile principles but also focuses on three aspects in software delivery: continuous testing, continuous delivery, and continuous monitoring.

- **Continuous testing** means testing earlier and continuously across the life cycle to achieve continuous feedback on quality. This process is also known as shift-left testing, which stresses integrating development and testing activities to ensure quality is built-in as early in the lifecycle as possible.
- **Continuous delivery** is the process of automating the deployment of the software to all of the environments to achieve consistent results. Using the same automated process with each environment improves efficiency and reduces the risk introduced by inconsistent processes.
- **Continuous monitoring** involves gathering feedback from monitoring data. This data comes from the servers running the application: from development, QA, and production or from metrics tools embedded in the application that capture user actions.

## IBM DevOps strategies results in a smooth flow of the delivery pipeline from development to production



*IBM DevOps strategies results in a smooth flow of the delivery pipeline from development to production*

Applying these DevOps strategies results in the smooth flow of smaller, frequent releases.

## IBM DevOps is working for organizations across industries

### Computer services



Synchrony Systems cuts development time by up to 90% with IBM Bluemix

[Learn more](#)

[Watch the video \(00:02:06\)](#)

### Healthcare



Healthcare information provider reduces deployment time from days to minutes

[Learn more](#)

### Banking



Rizal Commercial Banking Corp. transforms IT to gain 1.2M customers in one year

[Learn more](#)

### Computer services



DevOps approach helps increase release frequency by 300%

[Learn more](#)

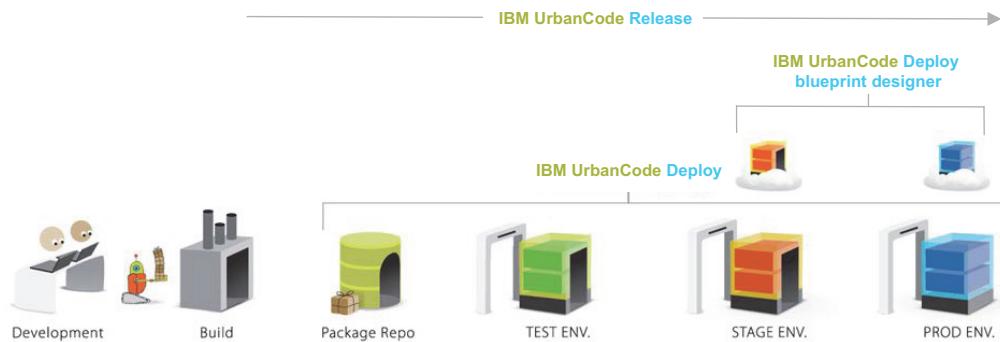
[Watch the video \(00:05:37\)](#)

### IBM DevOps is working for organizations across industries

DevOps works for a variety of industries. Common themes include cutting deployment time and increasing release frequency. Check out other IBM DevOps case studies at this location:

<https://www.ibm.com/ibm/devops/us/en/casestudies/#all>

## IBM UrbanCode products support the continuous delivery pipeline



Unit 1: Designing software deployment and deploying applications

© Copyright IBM Corporation 2017

### IBM UrbanCode products support the continuous delivery pipeline

Your delivery pipeline includes everything from development to production. Deployment is at the heart of the DevOps architecture. IBM UrbanCode Deploy is part of the DevOps story. It is a tool that supports the philosophy and facilitates continuous deployment.

For more information on the IBM UrbanCode Release course, see:

[https://www-03.ibm.com/services/learning/ites.wss/zz-en?pageType=course\\_description&cc=&courseCode=CQ450G](https://www-03.ibm.com/services/learning/ites.wss/zz-en?pageType=course_description&cc=&courseCode=CQ450G)

For more information on the IBM UrbanCode Deploy blueprint designer course, see:

[https://www-03.ibm.com/services/learning/ites.wss/zz-en?pageType=course\\_description&cc=&courseCode=CQ401G](https://www-03.ibm.com/services/learning/ites.wss/zz-en?pageType=course_description&cc=&courseCode=CQ401G)

## IBM UrbanCode Deploy supports the DevOps philosophy

- Automated, consistent deployments and application rollbacks
- Orchestration of changes on servers and tiers
- Configuration and security differences between environments
- Clear visibility: What is deployed where and who changed what
- Integration with middleware, provisioning, and service virtualization

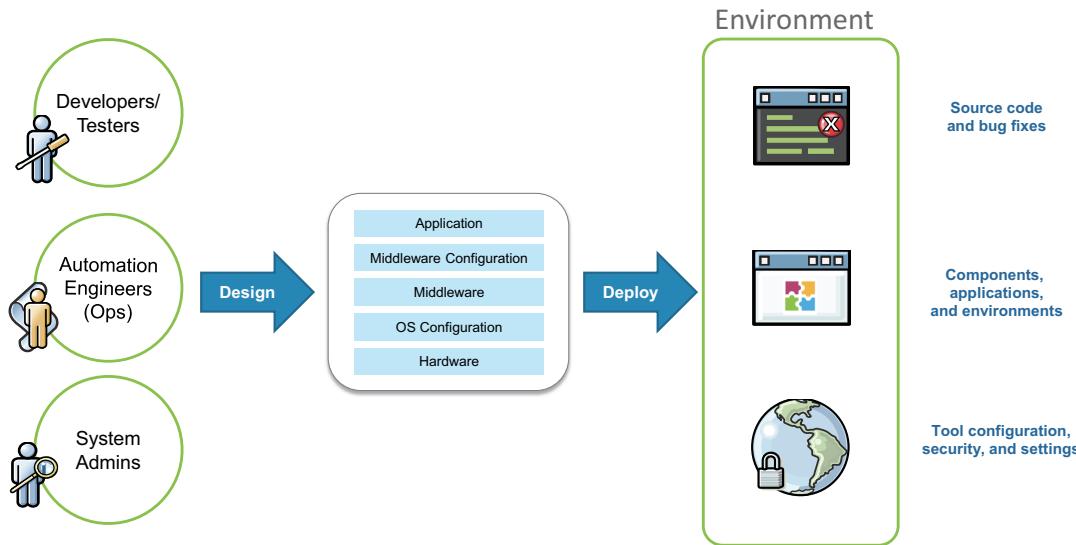
### *IBM UrbanCode Deploy supports the DevOps philosophy*

By using an environment management and deployment tool like IBM UrbanCode Deploy, you can design, deploy, and reuse environments quickly and help accelerate the delivery pipeline. IBM UrbanCode Deploy is an automation tool that manages the software components that get deployed, the middleware components and configurations that need to be updated, the database components that need to be changed, and the configuration changes to the environments to which these components are to be deployed. These tools also capture and automate the processes to carry out these deployments and configuration changes.

## Topics

- Accelerating software delivery and improving quality with IBM DevOps
- ▶ Defining the key concepts and workflow in IBM UrbanCode Deploy

## A modular approach permits the division of responsibilities in DevOps organizations



[Unit 1: Designing software deployment and deploying applications](#)

© Copyright IBM Corporation 2017

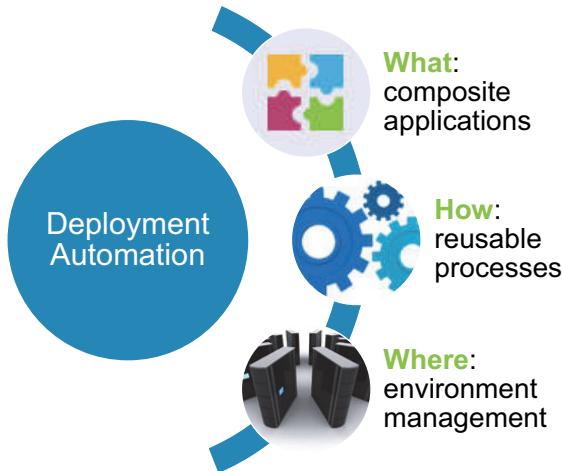
*A modular approach permits the division of responsibilities in DevOps organizations*

With IBM UrbanCode Deploy, you can separate development and operations responsibilities. 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, create templates, define the deployment environments, 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.

## Application models include components, processes, and environments



## Model-driven deployment

[Unit 1: Designing software deployment and deploying applications](#)

© Copyright IBM Corporation 2017

*Application models include components, processes, and environments*

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

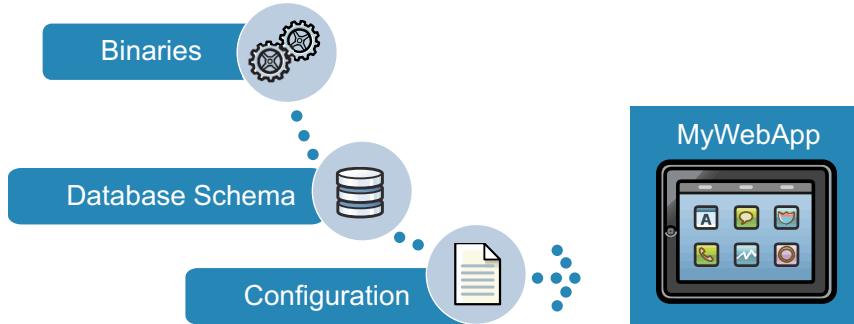
An application brings together the pieces of your deployment and includes three different parts:

1. It houses the components that you want to deploy.
2. It contains the application process that conducts deployment flow.
3. It contains information about the environment where you want to deploy.

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).

## A component is the smallest functional unit of an application

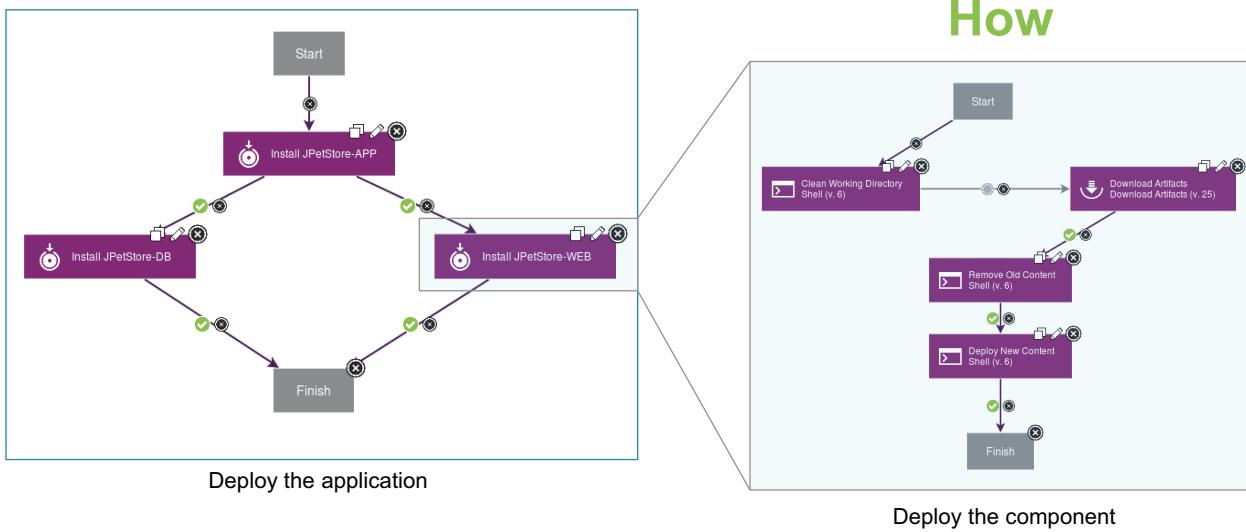
### What



*A component is the smallest functional unit of an application*

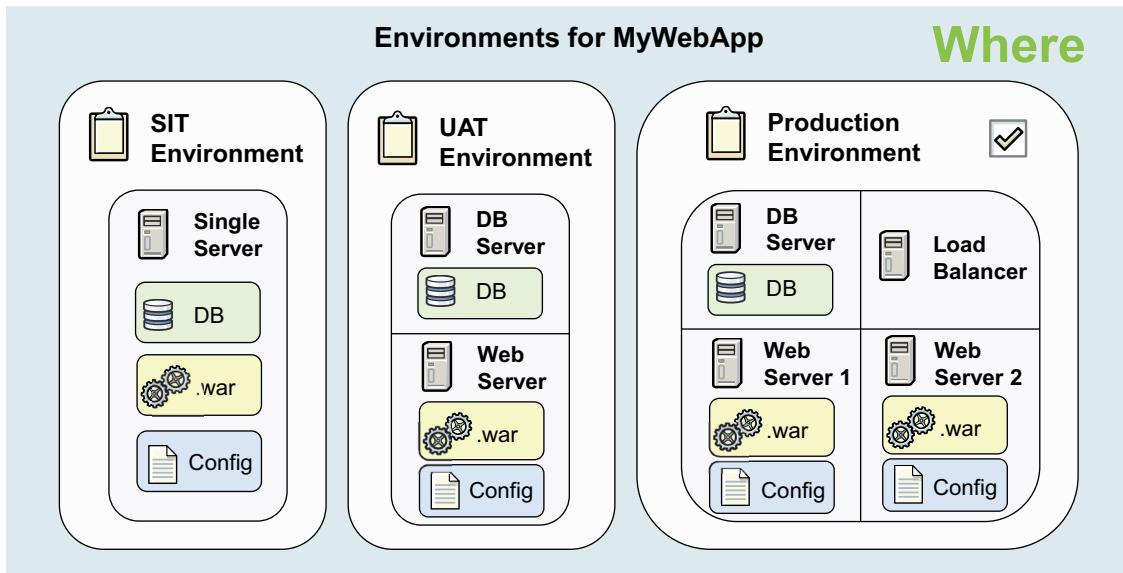
Components are containers for deployable artifacts and identify the “what” of a deployment. This includes any files, images, and databases that are associated with a software project.

## 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.

## An environment is a collection of resources that hosts an application



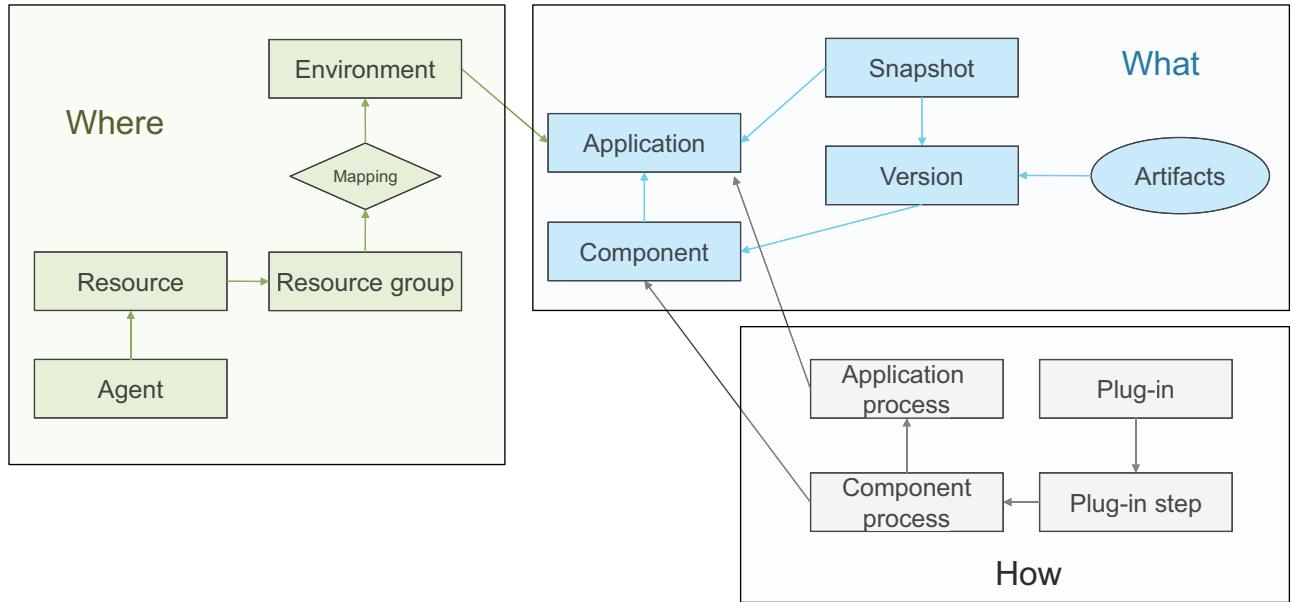
[Unit 1: Designing software deployment and deploying applications](#)

© Copyright IBM Corporation 2017

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.

## The objects for modeling software deployments work together



Unit 1: Designing software deployment and deploying applications

© Copyright IBM Corporation 2017

### The objects for modeling software deployments work together

The objects for modeling deployments fit into these categories:

#### What

- **Application:** A logical grouping of artifacts with self-contained definitions of automation and target environment
- **Component:** Files or configuration to be deployed, including the steps to deploy them
- **Version:** A container with the actual deployment artifacts, which are typically produced by an automated build
- **Snapshot:** A group of component versions that guarantees repeatability and reduces user error

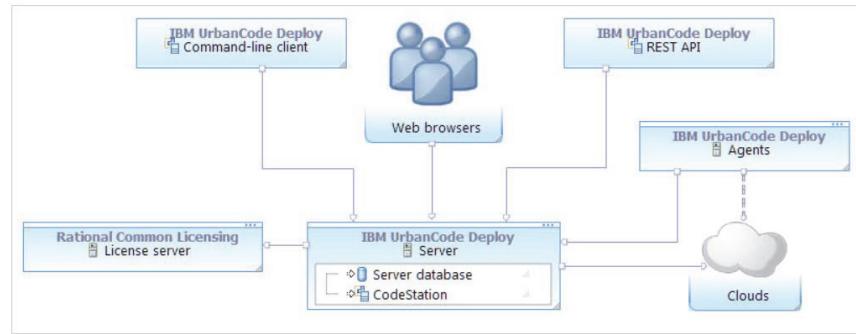
#### How

- **Plug-in:** Provides parameterized steps to perform automation; provided by IBM UrbanCode Deploy or through integrations
- **Component process:** Composes individual plug-in steps to define an automated process for a component
- **Application process:** Composes individual steps to define an automated process for a component

## Where

- **Agent:** Represents the IBM UrbanCode Deploy agent software that is running on a server and is installed on each server where commands must be executed
- **Resource:** Represents a deployment target, such as a physical server, virtual machine, or database
- **Resource group:** Represents a logical collection of resources that enable collections of resources to be easily reused
- **Environment:** Combines resources into a single logical deployment target

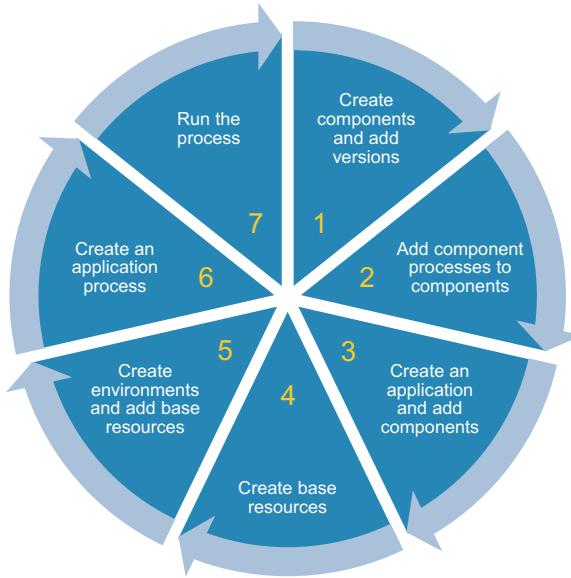
**The core installation of IBM UrbanCode Deploy includes a server, agents, and a license server**



*The core installation of IBM UrbanCode Deploy includes a server, agents, and a license server*

In this course, you access the server through web browsers, but you can also use REST API or the command-line client. Agents can be installed on cloud environments, virtual machines, containers, or physical systems.

## A basic workflow consists of some typical steps



Unit 1: Designing software deployment and deploying applications

© Copyright IBM Corporation 2017

### A basic workflow consists of some typical steps

Modeling software deployment in IBM UrbanCode Deploy includes configuring components and component processes and then adding those components to applications. Then you use processes to deploy the components to environments.

A simple workflow might start with creating components and the component processes. Next, you might add those components to an application and create resources and environments. Finally, you create an application process and then run that deployment.

While this slide demonstrates a basic workflow, you can create the pieces of your project in a different order. For instance, you could create your base resources first.

## Unit summary

- This unit covered typical deployment problems, the benefits of a DevOps culture, and key concepts and terms in IBM UrbanCode Deploy
- A deployment is the promotion of an application from one environment to the next
- IBM UrbanCode Deploy supports the principles and practices of DevOps
- The goals of DevOps are continuous testing, continuous delivery, and continuous monitoring
- IBM UrbanCode Deploy focuses on model-driven deployment
- Application objects fit into the *what*, *where*, and *how* of the deployment model

# **Unit 2 Configuring components and component processes**

IBM Training



## **Configuring components and component processes**

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

Components represent deployable items along with user-defined processes that operate on them, usually by deploying them. In this unit, you learn how to create a component, import component versions, create component processes, and work with properties.

## Unit objectives

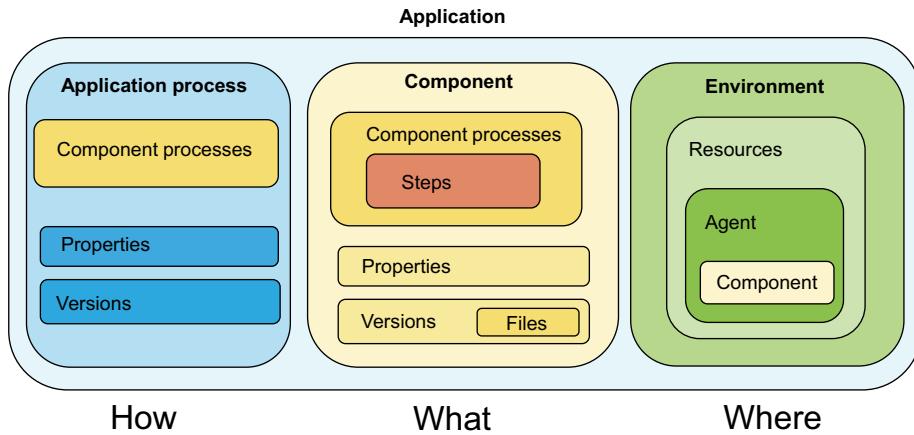
- Describe the purpose and structure of a component
- Create components and import component artifacts
- Create component processes to deploy components
- Store component information with properties

## Topics

### ▶ Components overview

- Creating components and importing component artifacts
- Creating processes to deploy components
- Storing information with properties

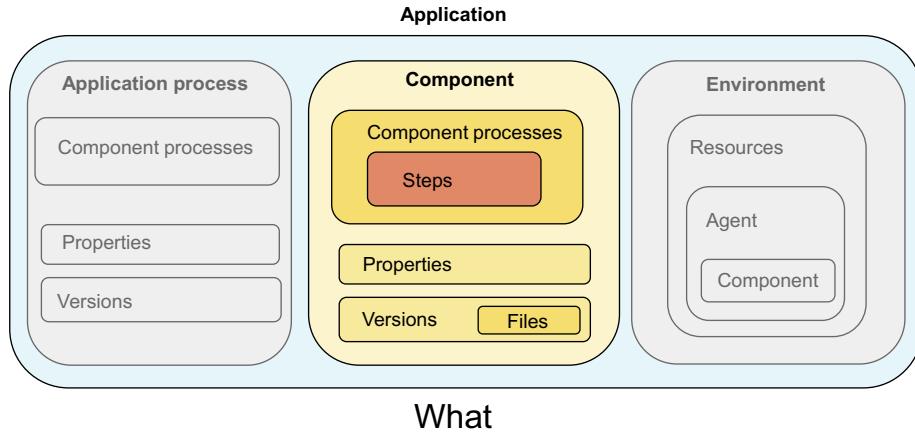
## A component is the smallest functional unit of an application



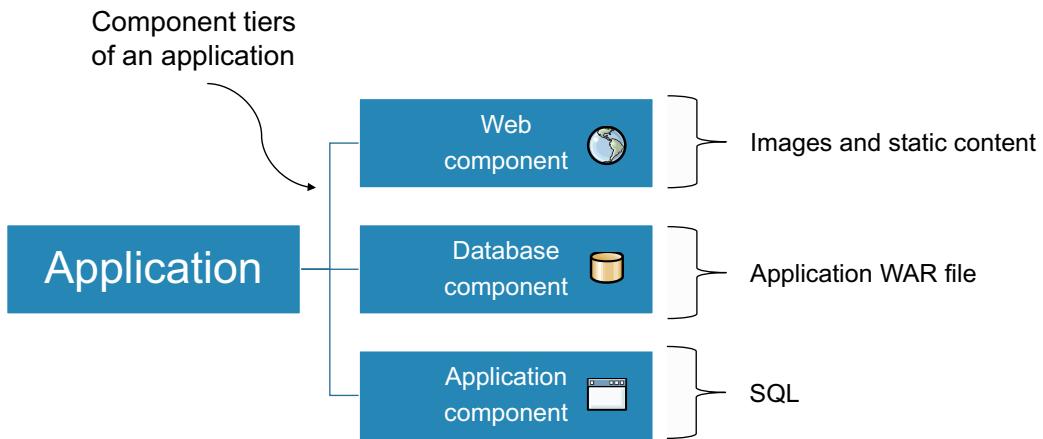
*A component is the smallest functional unit of an application*

You will see this representation of the objects in IBM UrbanCode Deploy for the next two units. This slide shows a visual that groups the parts of the product that make up an application.

## Components describe *what* gets deployed



This section focuses on **components**, which are containers for deployable items and their deployment instructions. A component usually contains a closely related set of artifacts, and you deploy its content on a single target.

**Each component represents artifacts from a single source**

---

Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

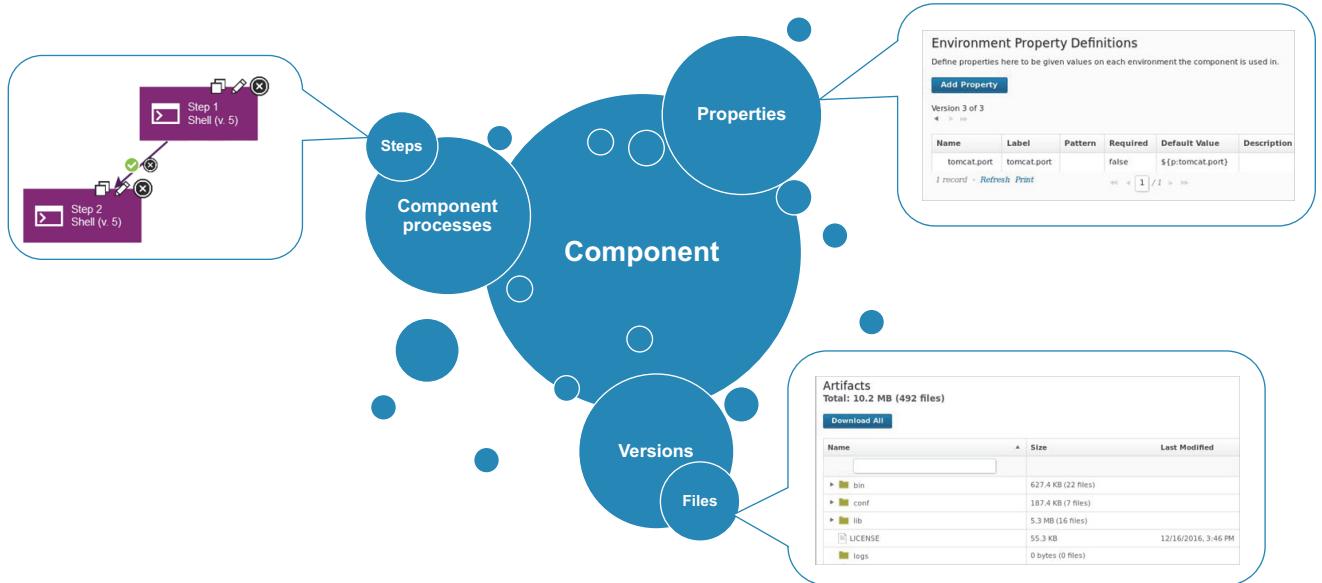
*Each component represents artifacts from a single source*

An application consists of a break down of components. In this example, the application has three component tiers:

- One for the web content
- One for the database content
- One for the application content

This arrangement gives flexibility to deployments. You can see what versions of each component are deployed to specific environments. The benefit of IBM UrbanCode Deploy is that it manages the source file and identifies which files have changes. Then it deploys only the changed files.

## A component contains versioned files, component processes, and properties



Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

*A component contains versioned files, component processes, and properties*

From a high-level view, a component has three major parts: versions, component processes, and properties.

- A **version** contains a set of artifacts. Versions ensure that the appropriate set of files is deployed.
- A **component process** is a list of commands that run on a target computer. Steps can manipulate files, run system commands, download files, and run programs.
- **Properties** are variables that store the locations of files, folders, servers, or other resources that might change.

## Topics

- Components overview
- ▶ **Creating components and importing component artifacts**
- Creating processes to deploy components
  - Storing information with properties

## Component artifacts can come from various source-code management sources

The screenshot shows the 'Create Component' dialog in the IBM UrbanCode Deploy interface. The dialog is titled 'Create Component' and contains fields for 'Name' (set to 'JPetStore-WEB'), 'Description', 'Teams', 'Component Template' (set to 'None'), and 'Component Type' (set to 'Standard'). Under 'Version Source Configuration', the 'Source Configuration Type' is set to 'File System (Versioned)' and the 'Base Path' is set to '/opt/PetStore/shared/web'. There are sections for 'Text File Extensions' (set to '.txt,.log,.properties') and 'Include Files' (set to '\*\*/\*'). The 'Exclude Files' section is empty. At the bottom, there are options for 'Import Versions Automatically' (unchecked), 'Copy to CodeStation' (checked), and 'Default Version Type' (set to 'full'). Below these are checkboxes for version import options: 'Use the system's default version import agent/tag.', 'Import new component versions using a single agent.' (selected), and 'Import new component versions using any agent with the specified ID.' The 'Agent for Version Imports' dropdown is set to 'agent-6892d1cfcf07'. The 'Cleanup Configuration' section includes 'Inherit Cleanup Settings' (checked) and 'Run Process after a Version is Created' (unchecked). A large callout arrow points from the 'Source Configuration Type' field in the dialog to a list of source control systems on the right. The list includes: AnthillPro, ClearCase, File System, File System (Versioned), Git, Maven, Rational Asset Manager, and Subversion.

Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

Component artifacts can come from various source-code management sources

When you create a component, you identify its type of artifacts and the location of the artifacts. In most cases, you add components by connecting the server to the system that hosts the artifacts, such as a Maven repository. In this example, and when you work through the lab, the versions are imported from the file system.

In the Create Component window, you need to consider a few points:

- **Component Template:** First, you decide if you want to base a new component on a template. With templates, you can reuse component processes and properties. You can save time when you create many components with a similar set of attributes. Although this course doesn't cover templates, you can consult the video demonstration in the notes section of this slide for more information. For more information about creating component templates, see this video:  
<https://developer.ibm.com/urbancode/videos/component-templates-ibm-urbancode-deploy-v6-0/>
- **Source Configuration Type:** The Source Configuration Type identifies the location of the artifacts. You can choose from many different source control systems.
- **Import Versions Automatically:** If you select the option to import versions automatically, the source location is periodically polled for new versions. Any versions that are found are automatically imported.
- **Run Process after a Version is Created:** In conjunction to automatically importing versions, at the bottom of the window, you can choose to run a process after a version is created. You specify an environment and an application process. Then any time a new version is automatically imported, the process runs. You can use this to automatically deploy a new version any time that it's created.

- **Copy to CodeStation:** By default, a complete copy of the content is imported into CodeStation, which leaves the artifacts in their original location untouched. CodeStation is the embedded artifact repository in IBM UrbanCode Deploy. It tracks file versions and maintains a complete history for all components.
- **Default Versions Type:** When you import a version, it can be full or incremental. You make this selection in the **Default Versions Type** field before you even import the component. A full version contains all component artifacts. An incremental version contains only artifacts that are different from the previous version. It's additive, like a patch.

## Components have versions that ensure the appropriate component instances are deployed

The screenshot shows the 'Components' tab in the IBM Rational Application Developer interface. The URL is 'Home > Components > jPetStore-WEB'. The component name is 'jPetStore-WEB'. The 'Versions' tab is selected. The 'Import New Versions' section shows a table with one record (version 1.0). The 'Currently Running Version Imports' section shows a table with one record (Manual Version Import) in the 'Executing' status.

Version	Statuses	Type	Created By	Date	Description	Actions
1.0	Statuses	Full	admin	12/12/2016, 7:16 PM		Compare Delete Copy

Import Type	Agent	Start	End	Status	Actions
Manual Version Import	agent-6892d1dcfe07	12/20/2016, 4:07 PM		Executing	Cancel

Components have versions that ensure the appropriate component instances are deployed

After you define the source for a component, you import its artifacts on the **Versions** tab. Each time a component is imported, including the first time, a new version of the component is created. This process is called **versioning**.

To manually import a new version, you click **Import New Versions**. The status of the import process displays in the Currently Running Version Imports section. You can see the agent that is importing the component.

## The agent is typically on the same computer where the artifacts are located

The screenshot shows two main sections of the IBM UrbanCode Deploy interface. On the left, the 'Components' page for 'JPetStore-WEB' displays a table of version imports, with one entry for a 'Manual Version Import' by 'agent-6892d1dcfe07' on 12/20/2016 at 4:07 PM. On the right, the 'System' settings page is shown, specifically the 'General Settings' section. A yellow box highlights the 'Agent for Version Imports' dropdown, which is set to 'agent-6892d1dcfe07'. An arrow points from the text 'The agent for importing component versions is defined in the system settings' to this highlighted dropdown.

[Unit 2: Configuring components and component processes](#)

© Copyright IBM Corporation 2017

*The agent is typically on the same computer where the artifacts are located*

Version artifacts are imported by the agent on that system. The agent copies the files from a repository or file system into CodeStation. You can choose the agent that imports versions in the system settings.

## A new version of the component is created every time a component's artifacts are modified and imported

New component version successfully imported

The screenshot shows the 'Components' section of the IBM UrbanCode Deploy interface. The 'Versions' tab is selected. A table lists two versions: '1.0' and '1.1'. The row for '1.1' is highlighted with a yellow box. An arrow points from the text 'New component version successfully imported' to this highlighted row. The table has columns for Version, Statuses, Type, Created By, Date, Description, and Actions.

Version	Statuses	Type	Created By	Date	Description	Actions
1.0		Full	admin	12/12/2016, 7:16 PM		Compare Delete Copy
1.1		Full	admin	12/12/2016, 7:16 PM		Compare Delete Copy

2 records - Refresh Print

Currently Running Version Imports

Import Type	Agent	Start	End	Status	Actions
				No executing version imports found.	

Refresh Print

*A new version of the component is created every time a component's artifacts are modified and imported*

A component might have several versions in CodeStation, and each version has a unique set of files.

## The version import history shows the status of attempted imports

The screenshot shows the 'Version Import History' table in the 'Configuration' tab of the JPetStore-WEB component. The table has columns: Import Type, Agent, Start, End, Status, and Actions. Five entries are listed, all failing due to an offline agent. An arrow points from the failed entry to the 'Error Log' panel below, which contains the message: 'The version import failed for the following reasons: The configured agent agent-bd0a7f9ef3f4 was offline.'

Import Type	Agent	Start	End	Status	Actions
Manual Version Import	agent-6892d1dcfe07	12/20/2016, 4:07 PM	12/20/2016, 4:07 PM	Success	[Edit, Delete]
Manual Version Import	agent-bd0a7f9ef3f4	12/20/2016, 4:06 PM	12/20/2016, 4:06 PM	Failed	[Edit, Delete]
Manual Version Import	agent-bd0a7f9ef3f4	12/20/2016, 4:06 PM	12/20/2016, 4:06 PM	Failed	[Edit, Delete]
Manual Version Import	agent-bd0a7f9ef3f4	12/20/2016, 4:06 PM	12/20/2016, 4:06 PM	Failed	[Edit, Delete]
Manual Version Import	agent-bd0a7f9ef3f4	12/12/2016, 7:16 PM	12/12/2016, 7:16 PM	Success	[Edit, Delete]

*The version import history shows the status of attempted imports*

You can see the version import history in the **Configuration** tab. If the version import failed, you can view the error log. Make sure that the files are available on the agent system and that the location of these files is correct on the component.

## A version contains a set of deployable artifacts

The screenshot shows two views of the IBM UrbanCode Deploy interface. On the left, the 'Components' view for 'JPetStore-WEB' is shown, with the 'Versions' tab selected. It lists two versions: 1.0 and 1.1. On the right, the 'Version: 1.1' details page is displayed, specifically the 'Artifacts' section. This section shows a table of imported files, including various banner images and a bird1.jpg file, with download links for each.

Name	Size	Last Modified	Actions
images	390.4 KB (86 files)		
banner_birds.gif	4.3 KB	12/12/2016, 7:09 PM	Download
banner_cats.gif	4.5 KB	12/12/2016, 7:09 PM	Download
banner_dogs.gif	4.0 KB	12/12/2016, 7:09 PM	Download
banner_fish.gif	4.3 KB	12/12/2016, 7:09 PM	Download
banner_reptiles.gif	4.4 KB	12/12/2016, 7:09 PM	Download
bird1.gif	11.7 KB	12/12/2016, 7:09 PM	Download
bird1.jpg	3.3 KB	12/12/2016, 7:09 PM	Download

A version contains a set of deployable artifacts

After a version is imported successfully, click the new version to view the files that were imported. From **Versions**, you see all files that are associated with a component. In this example, the images for the web server are listed in the Artifacts section.

## Change logs provide information about modifications to components

The screenshot shows the IBM UrbanCode Deploy interface. The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Components link is highlighted. Below the navigation is a breadcrumb trail: Home > Components > JPetStore-WEB. The main content area shows a table titled "Changes" with columns for Commit ID, User, Date, and Changes. One row is selected, showing commit ID 19 by user admin on 12/12/2016 at 7:16 PM. The "Changes" link in the table row is highlighted with a yellow box. A modal window is open, showing a table with three columns: Name, Before, and After. The "basePath" entry has its "Before" value as "/opt/JPetStore/shared/web" and its "After" value as "/opt/JPetStore/shared/web". An arrow points from the "basePath" row in the modal to the "Before" and "After" columns.

Commit ID	User	Date	Changes
117	admin	12/20/2016, 4:07 PM	FileSystemVersionedComponentProperties v.2 (Changes)
93	admin	12/14/2016, 3:53 PM	Versions/1.0/Properties for Component v.2 (Changes)
27	admin	12/12/2016, 7:57 PM	Processes/Deploy web component v.6
26	admin	12/12/2016, 7:51 PM	Processes/Deploy web component v.5
25	admin	12/12/2016, 7:50 PM	Processes/Deploy web component v.4
24	admin	12/12/2016, 7:45 PM	Processes/Deploy web component v.3
23	admin	12/12/2016, 7:45 PM	Processes/Deploy web component v.2
22	admin	12/12/2016, 7:28 PM	Processes/Deploy web component v.1
21	admin	12/12/2016, 7:16 PM	Versions/1.0/Properties for Component v.1 (Changes)
20	admin	12/12/2016, 7:16 PM	Versions/1.1/Properties for Component v.1 (Changes)
19	admin	12/12/2016, 7:16 PM	Version Property Definitions v.1 FileSystemVersionedComponentProperties v.1 (Changes) Ad-Hoc Properties v.1 (Changes) Environment Property Definitions v.1 Template v.1 (Changes)

Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

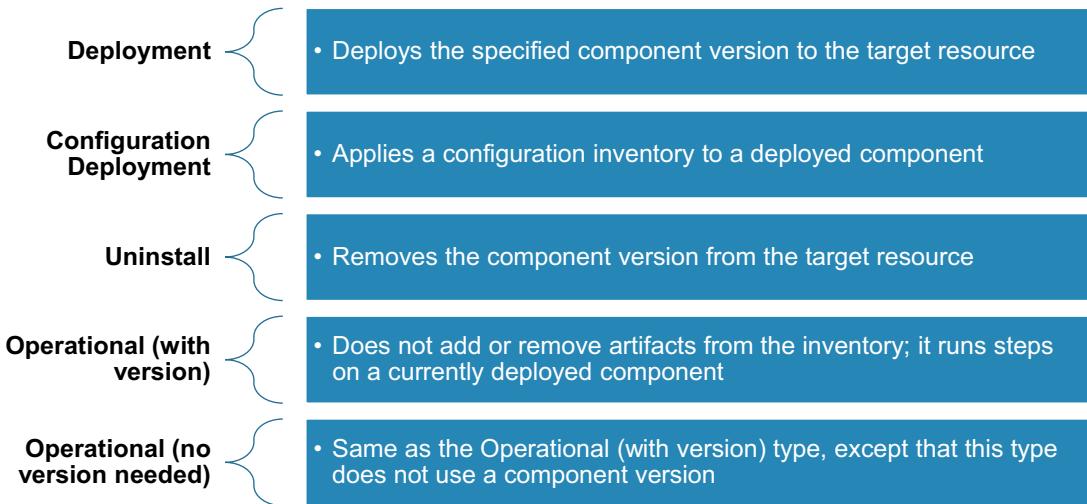
Change logs provide information about modifications to components

The **Changes** tab tracks changes made to the component. You can see who modified the component and the date and time the change was made. This information provides you with an audit trail of who is changing things. In this example, you can see that the `basePath` property was changed.

## Topics

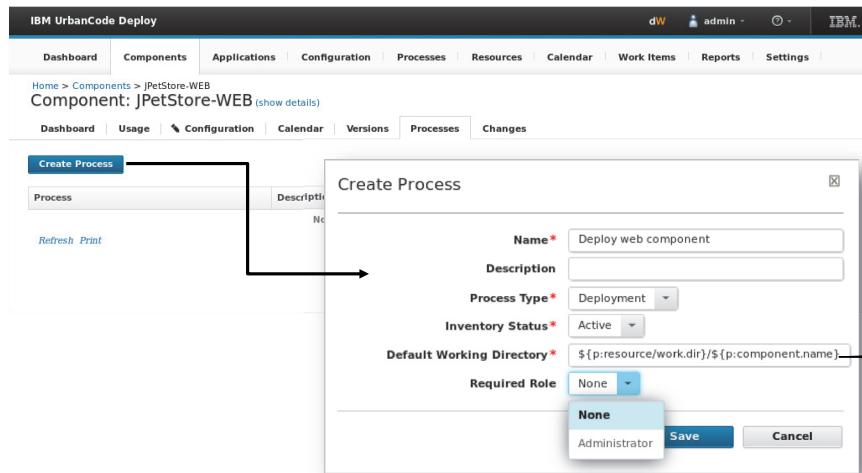
- Components overview
  - Creating components and importing component artifacts
-  **Creating processes to deploy components**
- Storing information with properties

## Component processes deploy, configure, or uninstall component versions



At this point, you created your components and imported the versions. Now you see how the actual deployment happens. You deploy components with component processes. Processes define what IBM UrbanCode Deploy does with a component's artifacts. For example, a component process can run tasks on a single component, such as deploying it, uninstalling it, or running configuration tasks on it.

## Components contain processes that perform operations on component files



`${p:resource/work.dir}` is the default working directory for the agent

`${p:component.name}` is the component name

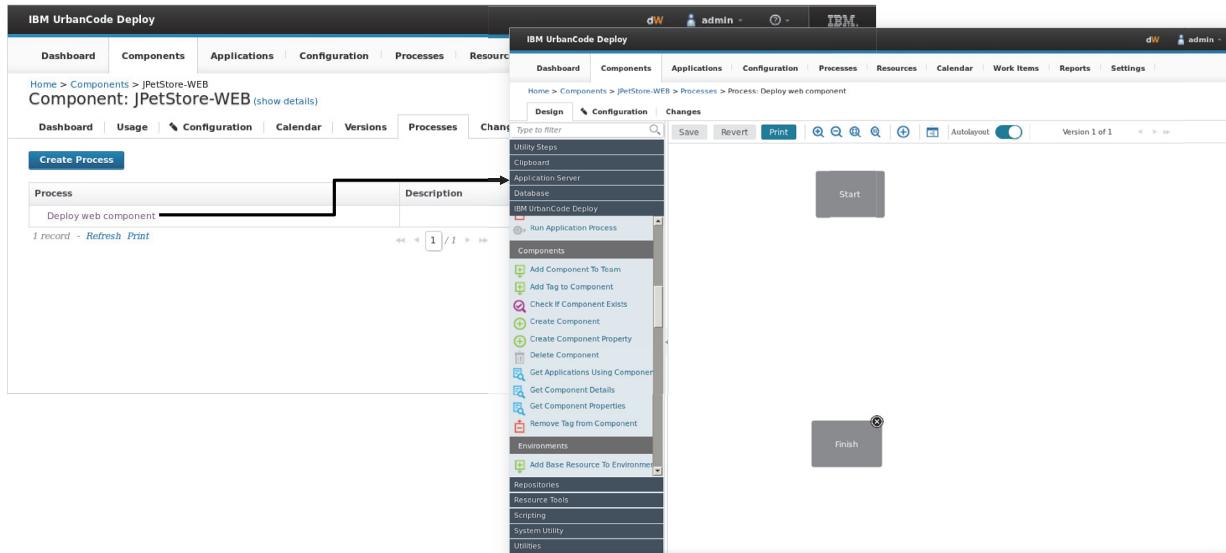
### Components contain processes that perform operations on component files

You create a component process in two steps: first, you configure basic information, such as the name and process type; then you use the process editor to assemble the process. A component process is defined for a specific component.

When you create a process, you need to make a few settings:

- **Process type:** The process type determines how the component handles updating the inventory in IBM UrbanCode Deploy. Here you decide whether your process will deploy components or configuration or if it will uninstall a component.
- **Inventory status:** The Inventory status is a status that is applied to a component version after the processes successfully runs. An Active status indicates that the component version is deployed to its target resource.
- **Default working directory:** The default working directory defines the location that the agent uses to run the process. You can see the default value in the example, where  `${p:resource/work.dir}` is the default working directory for the agent and  `${p:component.name}` is the component name. The default value resolves to `agent_directory\work\component_name_directory` and is stored in the `componentProcess.defaultWorkDir` property.
- **Required role:** You can set a value in the **Required Role** field to restrict access to a specific role of users who can run this process. The available options are derived from the IBM UrbanCode Deploy security system. The default value is **None**, meaning that anyone can run the process.

## Component processes are created with the process editor



Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

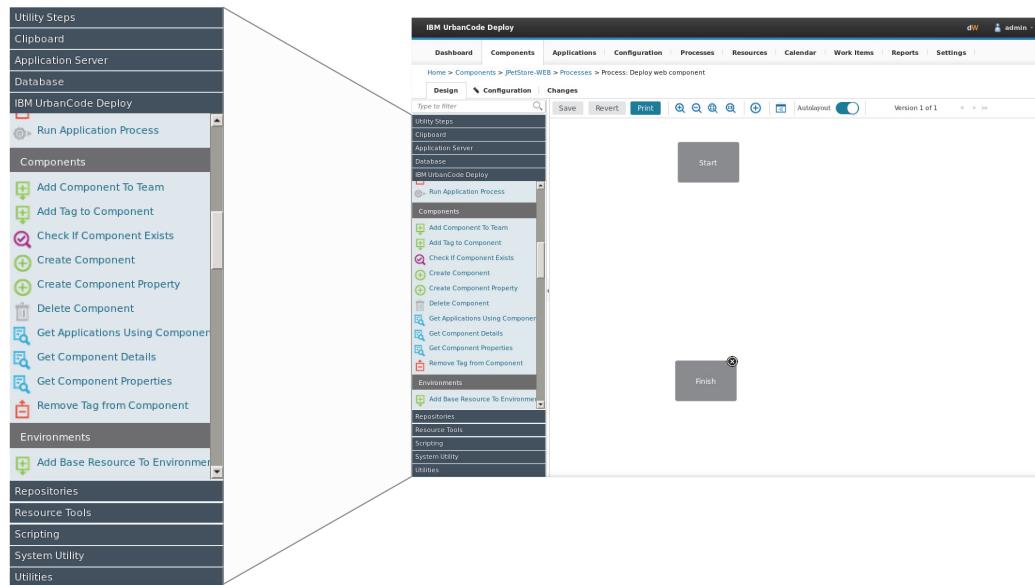
Component processes are created with the process editor

Component processes are created with the process editor. To open the process editor, click the process name. The process editor has a **Steps** palette on the left and a design space on the right. A process begins with the Start step and must end with the Finish step.

## IBM Training



Process steps are selected from a menu of standard steps that replace typical deployment scripts and manual processes



Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

Process steps are selected from a menu of standard steps that replace typical deployment scripts and manual processes

The available steps in the palette are determined by the installed plug-ins. Plug-ins consist of distinct processes called steps. Each step consists of these items:

- Properties
- A command that runs the step
- Post-processing instructions

Step properties can serve various purposes, from providing input to the command to supplying some, or all, the actual command itself.

Some plug-ins are included with the product, but you can install others. IBM UrbanCode Deploy provides steps for several utility processes, such as inventory management, and workflow control.

## A number of plug-ins come with the product and many others are available that can be downloaded and installed

Plugin	Description
File Utils	The FileUtils plugin allows users to perform folder and file level tasks as part of their deployment process. Among other things, it includes steps for deleting or creating directories and replacing tokens in a file.
General Utilities	Utilities for basic tasks that don't fit in a specific plugin.
Groovy	The Groovy plugin is an automation based plugin that provides steps for executing user defined Groovy scripts.
IBM UrbanCode Deploy Applications	Plugin for creating and managing IBM UrbanCode Deploy applications.
IBM UrbanCode Deploy Components	Plugin for creating and managing IBM UrbanCode Deploy components.
IBM UrbanCode Deploy Configuration Management	Plugin for downloading and uploading configuration templates from/to IBM UrbanCode Deploy
IBM UrbanCode Deploy Environments	Plugin for creating and managing IBM UrbanCode Deploy environments.
IBM UrbanCode Deploy Processes	Supports interaction with processes and process requests.
IBM UrbanCode Deploy Resources	Manages IBM UrbanCode Deploy Resources
IBM UrbanCode Deploy Versioned File Storage	A plugin to allow IBM UrbanCode Deploy to upload artifacts to CodeStation.

16 records - Refresh Print    1 / 2    < >

Most Popular

Jenkins 2.0 2.0.811128 Source Config Plugin IBM UrbanCode Deploy	Jenkins 1.2.9 Source Config Plugin IBM UrbanCode Deploy	WebSphere Application Server – Deployment 98 IBM UrbanCode Deploy	Docker 3 IBM UrbanCode Deploy	IBM WebSphere Liberty 5 IBM UrbanCode Deploy
Apprenda Source Config Plugin IBM UrbanCode Deploy	PowerShell 4 IBM UrbanCode Deploy	WebSphere Application Server – Install 40 IBM UrbanCode Deploy	WebSphere Application Server – Configure 40 IBM UrbanCode Deploy	Docker Build 4 UrbanCode Build

Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

*A number of plug-ins come with the product and many others are available that can be downloaded and installed*

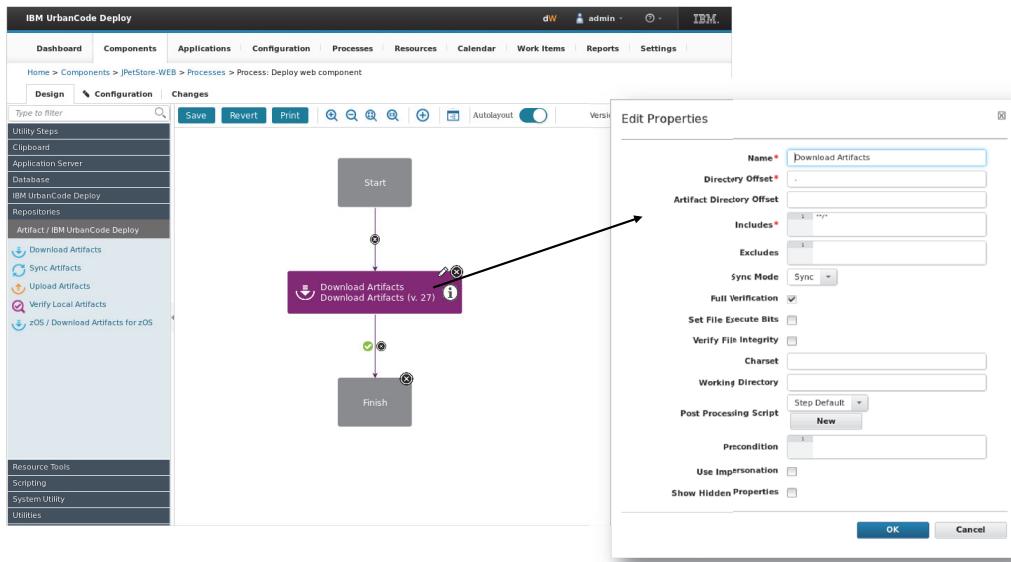
Other process steps are provided by plug-ins that provide integration with common deployment tools and application servers. A full list of plug-ins can be found on DeveloperWorks at this address:

<https://developer.ibm.com/urbancode/plugins/>

Plug-ins provide processing and integration functions. There are two types of plug-ins:

- **Source-type plug-ins** integrate with external systems to import artifacts and create component versions. Source-type plug-in properties are typically defined when a component is created.
- **Automation-type plug-ins** provide process steps that manipulate components, typically by deploying them. You can set automation-type plug-in properties at design time in the process editor or at run time in the user interface.

## The palette provides automation steps that you can drag onto the process editor



Unit 2: Configuring components and component processes

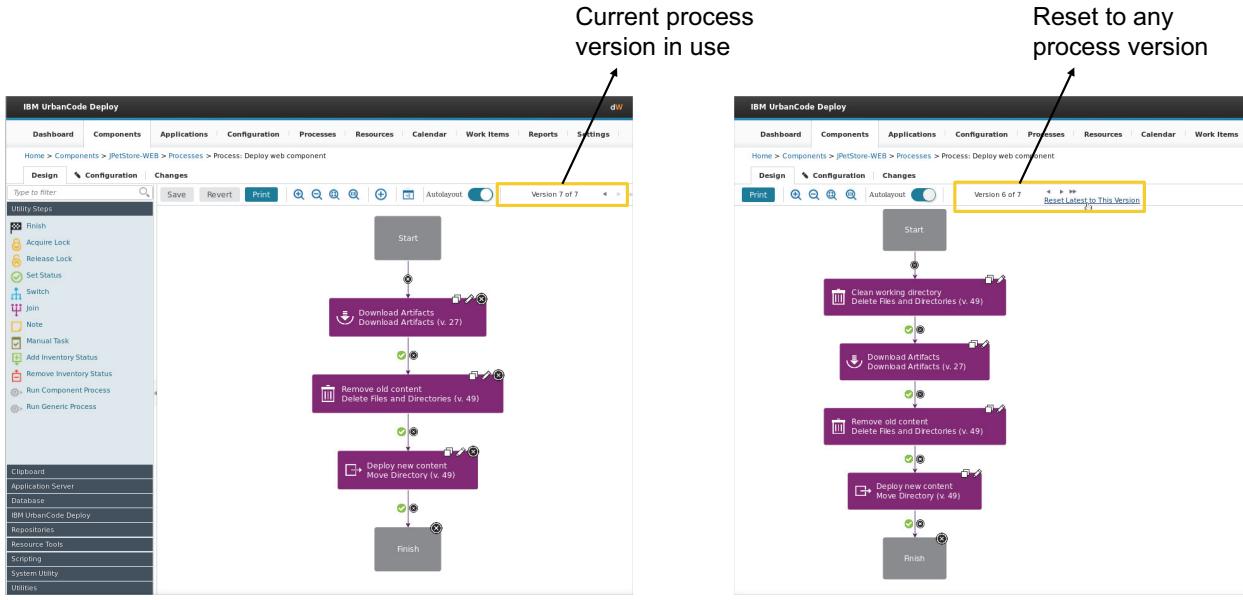
© Copyright IBM Corporation 2017

*The palette provides automation steps that you can drag onto the process editor*

The process editor is a visual drag-and-drop editor where you can drag process steps from the palette onto the design space and configure them as you go. As more steps are placed, you visually define their relationships with one another.

When you drag a step into the process editor, you can edit the fields for that step. In this example, the Download Artifacts step is prepopulated with information that you can use.

## Processes are versioned artifacts



Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

### Processes are versioned artifacts

Each time you save a process, a new version of that process is saved. You can scroll through the versions to view and then reset to any process version that you previously saved.

## The Changes tab tracks the creation date and author of the versions

The screenshot shows the IBM UrbanCode Deploy interface with the 'Changes' tab selected. The page displays a table of version history with columns for Version, User, Date, and Comment. A yellow box highlights the 'Changes' tab in the navigation bar. A bracket points from the text 'Tracks who made changes and when' to the 'User' column of the table.

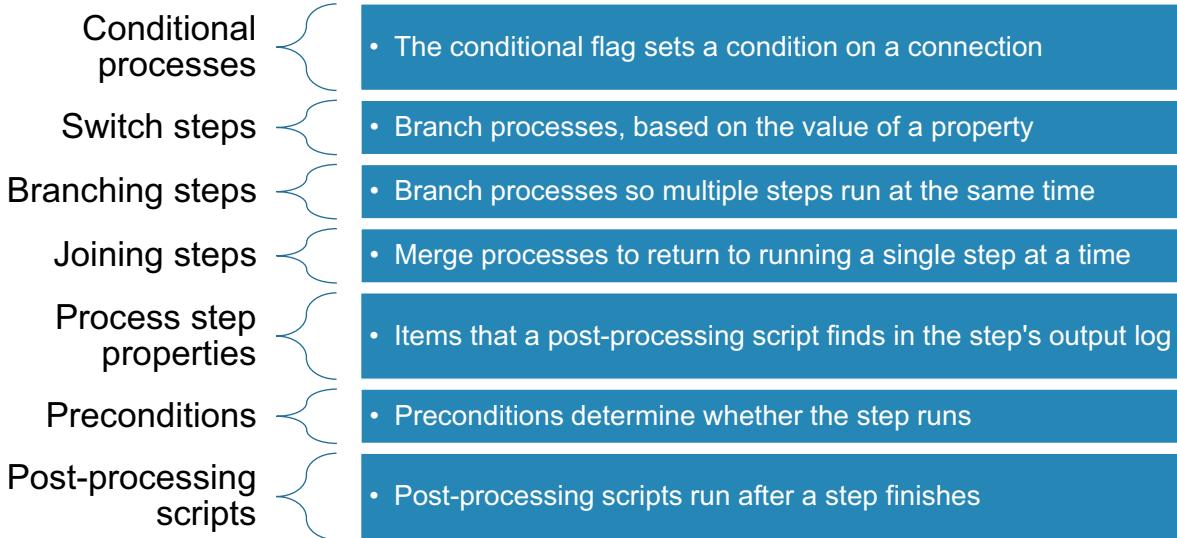
Version	User	Date	Comment
8	admin	12/29/2016, 8:58 PM	
7	admin	12/29/2016, 8:55 PM	
6	admin	12/12/2016, 7:57 PM	
5	admin	12/12/2016, 7:51 PM	
4	admin	12/12/2016, 7:50 PM	
3	admin	12/12/2016, 7:45 PM	
2	admin	12/12/2016, 7:45 PM	
1	admin	12/12/2016, 7:28 PM	

Tracks who made changes and when

*The Changes tab tracks the creation date and author of the versions*

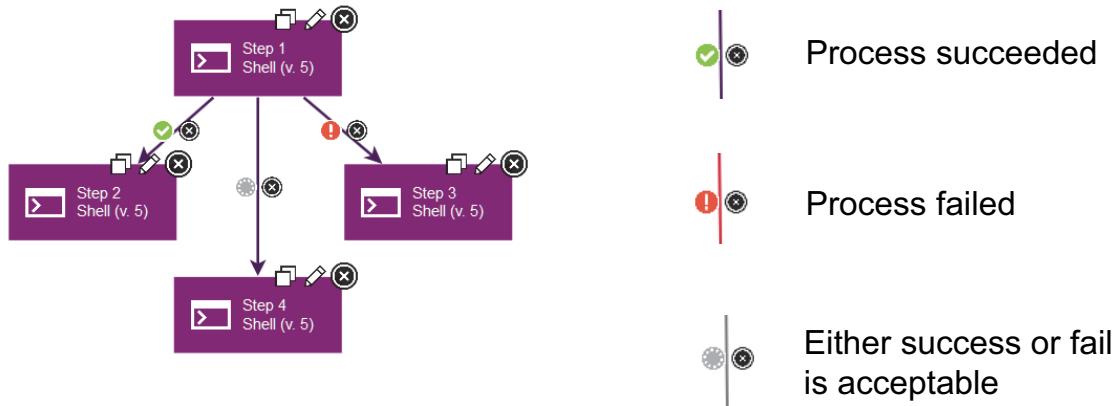
When a process is created, information is gathered to inform when a change was made and who made it.

## Processes use different kinds of steps and joining techniques



A component process can be as simple as a single step or contain numerous relationships, branches, and process switches. When you design processes, you use different kinds of process steps and joining techniques.

## Conditional processes use flags to set conditions on connections

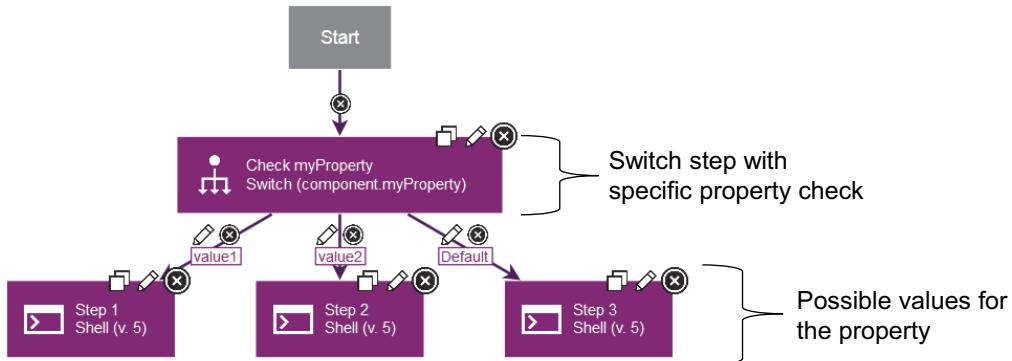


By default, all connections between steps have the flag set to **Success**, which is the green circle with a check. This setting means that the originating step must finish processing before the next step starts to process. If you set the flag to **Fail**, which is the red circle with an explanation point, then the step must have a failed process to reach the next step in that branch. And if you set the flag to a gray circle, the process moves to the next step, whether the previous step's process completed or failed.

The possible conditional flags are as follows:

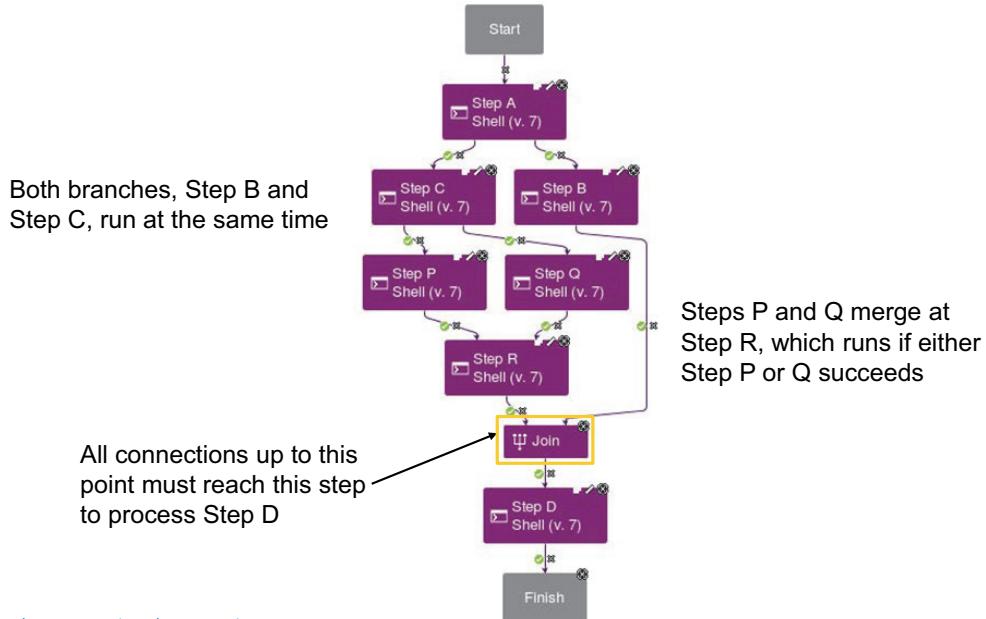
- **Success**: The process is complete.
- **Fail**: The process did not finish.
- **Both**: The process is complete or the process did not finish. Either status is acceptable, and the next step can start.

## Switch steps branch processes based on the value of a property



When you add a switch step, you specify a property. Then you add two or more outgoing connections from the step, each with a possible value for that property. At run time, the process follows the connection with the matching property value.

## Branching steps runs multiple steps at the same time



Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

*Branching steps run multiple steps at the same time*

You can branch processes so that multiple steps run at the same time by adding multiple outgoing connections from a step. You can use a Join step to merge processes into running a single step at a time. Join steps indicate that all connections to the Join step must succeed. If any of the steps that connect to a Join step fail or do not run, the Join step causes the process to fail. In this example, all steps leading up to Steps R and B must complete, because Steps R and B both connect to a Join step.

## Topics

- Components overview
  - Creating components and importing component artifacts
  - Creating processes to deploy components
-  Storing information with properties

## Properties are variables that store information about different elements

The screenshot shows the 'Edit Properties' dialog for a step named 'Upgrade DB'. The URL field is highlighted with a yellow box. The dialog includes fields for Driver Classname (com.mysql.jdbc.Driver), DB Driver Jar (lib/mysql-connector-java 5.1.20-bin.jar), User (jpetsstore), Password (\*\*\*\*), SQL File path (.), SQL File Include (\*.sql), Current Version SQL (SELECT VER FROM DE\_VERSION WHERE RELEASE\_NAME = ?), Delete Version SQL (DELETE FROM DE\_VERSION WHERE RELEASE\_NAME = ?), Update Version SQL (INSERT INTO DE\_VERSION (RELEASE\_NAME,VER) VALUES(?,?)), Working Directory, Post Processing Script (Step Default), Precondition, and Use Impersonation.

Properties are pointers  
within a step to a  
predefined property

Name	Value	Description	Actions
db.url	jdbc:mysql://localhost:3306/petstore_xl	The URL is the MySQL database relative to the target system	Edit; Delete
tomcat.home	ApacheTomcat/tomcat	The Tomcat home folder on the target computer	Edit; Delete
tomcat.managerPort	http://192.168.27.100:8085/manager/xsd	The location of the Tomcat manager application	Edit; Delete
tomcat.port	8085	The port for the environment	Edit; Delete
tomcat.redirectPort	8443		Edit; Delete
tomcat.shutdownPort	8005		Edit; Delete
tomcat.start	./apachectl start	The location of the startup script for tomcat	Edit; Delete

Properties that are defined for  
specific environments

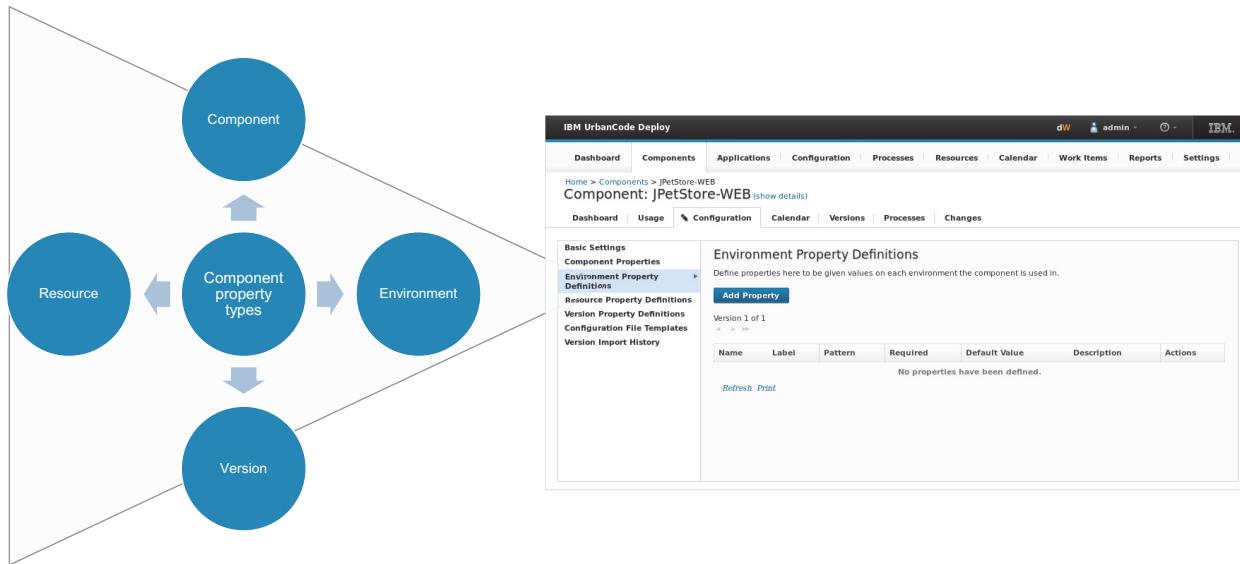
Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

Properties are variables that store information about different elements

Properties are pointers to values. They store information about components, environments, processes, applications, and global properties for the system. In this example of a component step, the URL field points to a value that is defined for the environment. At run time, the environment property, in this case the db.url value, will be substituted here.

## Components can have several types of properties



Unit 2: Configuring components and component processes

© Copyright IBM Corporation 2017

### Components can have several types of properties

**Component properties:** Component properties are values defined and set directly on the component. They stay the same for all versions of the component. So any time you import a new version, the properties are already set.

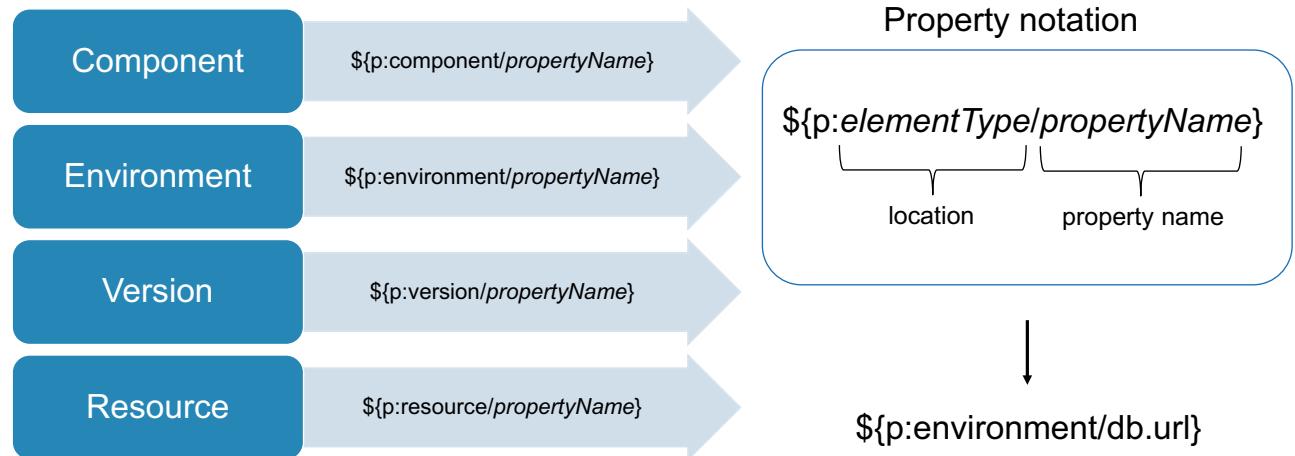
**Property definitions:** Components have a special type of property called property definitions. Component property definitions can be used for environments, versions, and resources. You use definitions when you have a value that the component uses that needs to change each time you use that component. These properties are defined on the component, but their values are set in other places. For example, *component environment properties* are defined on a component, but the values are not specified until the component is deployed to an environment. In the previous slide, the URL field pointed to a predefined environment property called db.url. You can therefore assign a different database URL to each environment. You might want to do so if you have a test database at one URL for your development environment and a production database at another.

For example, if different database users have access to the development and test databases, you can set the database user name and password in each environment. Then you use a component environment property to point the correct database URL for each environment.

- **Component version properties** are defined on a component, but the values are specified on component versions. This type of property stores information that changes with each version of the component, such as a version number. Each component version can have a different value for its component version property.
- **Component resource properties** are defined on components, but they are not given a value until the component is added to an environment as a resource. Then the property behaves much like a resource property. Because a component can be added to an environment multiple

times, each instance of the component can have a different value for the component resource property. If you need to use two copies of a component in the same environment but use a different value between them, use a resource property definition.

## Properties are referred to by scope



### Properties are referred to by scope

Properties are referred to by scope. **Scope** is the type of artifact that contains the property, such as the environment or component. When you refer to a property, you use the notation that starts with a dollar sign; open curly brace; *p*, which stands for property; and then a colon. Then you identify the artifact, followed by a slash, the property name, and a closed curly brace.

`${p:component/prop.name}`, where the *p* stands for *property*. It means to look in this location with this name.

Keep in consideration where values change; that is where you want to set the properties.

This slide shows the most commonly used component properties and definition. Application, environments, resources, the system, and agents also have properties. For more information, see the IBM Knowledge Center:

[https://www.ibm.com/support/knowledgecenter/SS4GSP\\_6.2.3/com.ibm.udeploy.doc/topics/ud\\_properties\\_overview.html](https://www.ibm.com/support/knowledgecenter/SS4GSP_6.2.3/com.ibm.udeploy.doc/topics/ud_properties_overview.html)

## Properties can also be referred to without scope

### Property notation **with** scope

`${p:elementType/propertyName}`

location      property name

### Property notation **without** scope

`${p:propertyName}`

property name



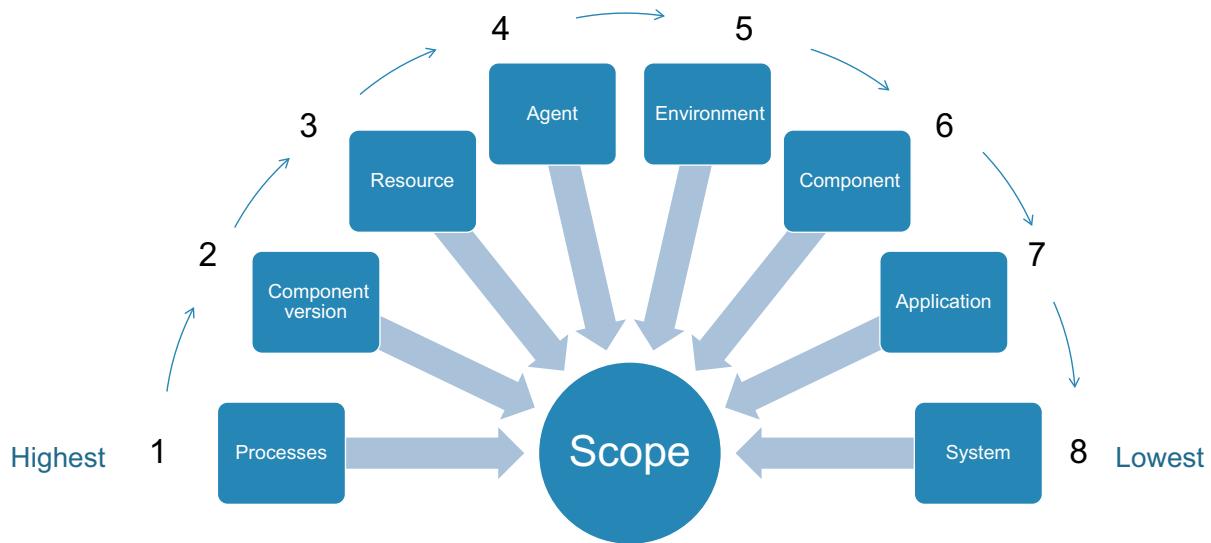
`${p:environment/db.url}`



`${p:db.url}`

There might be situations where you don't want to reference a specific artifact. In this case, you can refer to a property without scope by leaving out the artifact ( `${p:propertyName}`). For example, with the environment variable, db.url, you can refer to it with the artifact name, environment, as you see on the left ( `${p:environment/db.url}`). Or you can refer to the same environment variable without scope by removing the artifact name, as shown in the right ( `${p:db.url}`). In this way, you can refer to properties defined in other places, such as in the application, system, or resource.

## Property scope determines the precedence of properties with the same name



*Property scope determines the precedence of properties with the same name*

If a property is defined in multiple places, its value is determined by the property's order of precedence. You can use the order of precedence to override properties in specific contexts. If you refer to a property without scope, then the property with the highest order of precedence is used.

This slide shows the order of precedence from highest to lowest. The highest precedence is given to properties that are referenced in processes. The lowest precedence is given to properties referenced in the system.

In this way, you can override properties to change the application behavior without changing the code. For example, an application might have a database for testing and a separate database for production use. In this case, you might create an application property, such as `databaseLocation`, to store the testing database location. The application and component processes can refer to the property without specifying the application scope, like this:  `${p:databaseLocation}`. Then, in the production environment, you can override the application property with an environment property that specifies the production database location. The processes still refer to  `${p:databaseLocation}`, but in the context of the production environment, the environment property takes precedence; so the application uses the production database.

## Properties are primarily set from the Configuration tab

The screenshot shows the 'Environment Properties' section of the configuration tab. It displays a table of properties with columns for Name, Value, Description, and Actions (Edit, Delete). The properties listed are:

Name	Value	Description	Actions
db.url	jdbc:mysql://localhost:3306/jpetstore_sit	The URL to the MySQL database, relative to the target system	Edit Delete
tomcat.home	/opt/webservers/sit-apache-tomcat	The Tomcat home folder on the target computer	Edit Delete
tomcat.managerurl	http://192.168.27.100:8085/manager/text	The location of the Tomcat manager application	Edit Delete
tomcat.port	8085	The port for the environment	Edit Delete
tomcat.redirect.port	8445		Edit Delete
tomcat.shutdown.port	8005		Edit Delete
tomcat.start	/opt/webservers/sit-apache-tomcat/bin/startu p.sh	The location of the startup script for Tomcat	Edit Delete

Below the table, it says '7 records - Refresh Print' and has navigation buttons for rows 1/10.

You can also set properties here:

- Processes
- Rest API and command line
- Output properties

### Properties are primarily set from the Configuration tab

You can set properties in the following locations:

- **Configuration tab:** You can manage properties for applications, components, and environments.
- **Processes:** Several different steps are available to create and set properties in processes.
- **REST API and command-line client:** Several commands in the REST API and in the command-line client can set properties.
- **Output properties:** You can pass properties from one step to another within a process.

Properties for applications, processes, components, component versions, resources, and agents are available on their respective **Configuration** tabs. System (global) properties are available on the **Settings** tab by clicking **Settings > Properties**.

## Unit summary

- A component has three major parts: versions, processes, and properties
- Components are added by connecting the server to the system that hosts the artifacts
- A component version contains a set of deployable artifacts
- A component process is a series of user-defined steps that operate on a component's artifacts
- Component properties are values that are assigned on individual components
- Property definitions are defined on the component, but their values are set in other places

## Exercises: Configuring components and component processes

- Create components and add user-defined properties to them
- Import artifacts to create component versions
- Create component processes





# **Unit 3 Defining deployments with resources, environments, and application processes**

IBM Training



## **Defining deployments with resources, environments, and application processes**

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

Applications identify the component versions that must be deployed together. They also define the different environments that the components must go through on the way to production. In this unit, you learn to create an application and add components, create environments and add resources, and create application processes.

## Unit objectives

- Describe the structure of an application
- Explain how agents communicate with the server
- Describe the properties used for resources
- Create an environment for deployments
- Map resources to environments
- Create an application process to install components

## Topics

### ▶ Managing components in applications

- Configuring the agent as a deployment-target host
- Organizing resources in a resource tree
- Setting up deployment targets with environments
- Coordinating deployments with application processes

## IBM Training



### Applications associate resources with environments and define processes to run deployments

The screenshot shows two views of the IBM UrbanCode Deploy application management interface. On the left, a modal window titled 'Create Application' is open, displaying fields for Name (jPetStore), Description, Teams (None), Application Template (None), and Notification Scheme (None). A 'Save' button is at the bottom. On the right, the main application list view shows a single record for 'jPetStore'. A bracket labeled 'Application container' spans both the creation form and the list view.

An application is a container for components, processes, and resources. When you create an application, you determine a few things:

- You'll want to identify who has access to the application. You can grant access to teams, and you can set the application security type.
- You decide if you want to base your application on a template. Application templates can help you to be sure that new applications contain the types of components, environments, and resources that you specify. You can also specify the application properties, environment properties, and environment gates for an application. Although this course doesn't cover templates, you can consult the video demonstration in the notes section of this slide for more information.

For more information about creating application templates, see the video:

<https://developer.ibm.com/urbancode/videos/creating-application-templates/>

For more information about creating applications based on templates, see the video:

<https://developer.ibm.com/urbancode/videos/creating-applications-based-on-templates-urbancode-deploy-v-6-2/>

- You can choose to use a Notification Scheme. A notification scheme sends out notifications that are based on events, such as when an application deployment fails or succeeds.

## An application groups components together

The screenshot shows the IBM UrbanCode Deploy interface. In the top-left corner, there's a sidebar with links like Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below this, a main content area has tabs for Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Applications tab is selected. Under Applications, there are buttons for Create Application, Import Applications, Actions..., and Flat list. A search bar labeled 'Name' contains the text 'JPetStore'. A yellow box highlights the 'JPetStore' entry in the dropdown. A callout arrow points from this entry to a larger modal window titled 'Add a Component'. This modal has a 'Select a Component\*' dropdown menu. Inside the dropdown, three items are listed: 'JPetStore-APP' (selected), 'JPetStore-DB', and 'JPetStore-WEB'. A yellow box highlights the 'JPetStore-APP' item. Another callout arrow points from this highlighted item to a text box on the right that reads: 'Associates components with an application'.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

An application groups components together

After defining the application itself, the next step is to identify the components that it manages.

## Applications manage components and map the dependencies between them

Components available to the application

The screenshot shows the 'Components' tab selected in the navigation bar. The table lists three components: jPetStore-APP, jPetStore-DB, and jPetStore-WEB. Each row has 'Run Process' and 'Remove' actions. A yellow bracket on the left side of the table groups the three rows under the heading 'Components available to the application'.

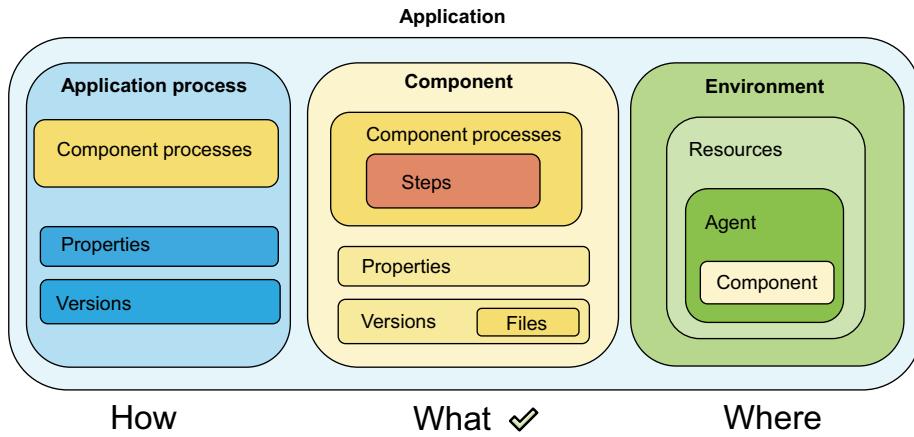
Component	Description	Actions
jPetStore-APP		Run Process Remove
jPetStore-DB		Run Process Remove
jPetStore-WEB		Run Process Remove

When you associate components with an application, they are available to be used in deployments.

## Topics

- Managing components in applications
- ▶ Configuring the agent as a deployment-target host
  - Organizing resources in a resource tree
  - Setting up deployment targets with environments
  - Coordinating deployments with application processes

## Applications determine the *what, where, and how* of a deployment



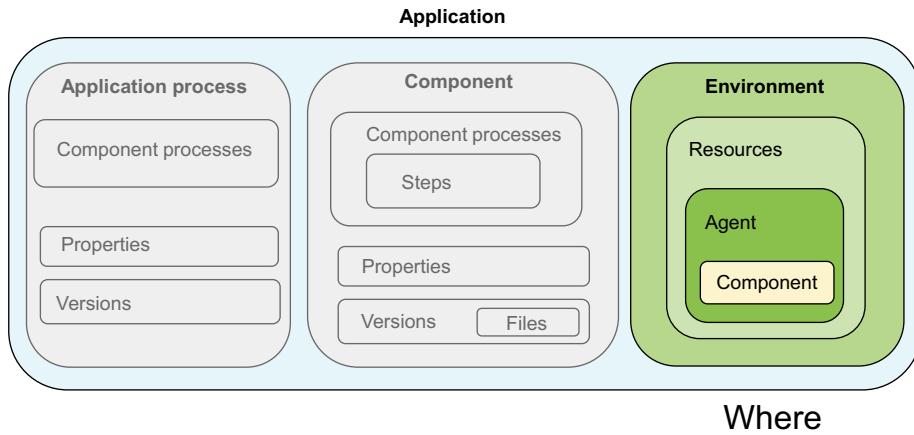
[Unit 3: Defining deployments with resources, environments, and application processes](#)  
*Applications determine the what, where, and how of a deployment*

© Copyright IBM Corporation 2017

The introduction of this course described how applications bring together the pieces of a deployment. This application model provides the container for what gets deployed (the components), where it gets deployed (the environments), and how it gets deployed (the processes).

In the previous unit, you learned how to set up the components. In this next section, you learn how to set up an environment.

## An environment associates components with an agent on the target host



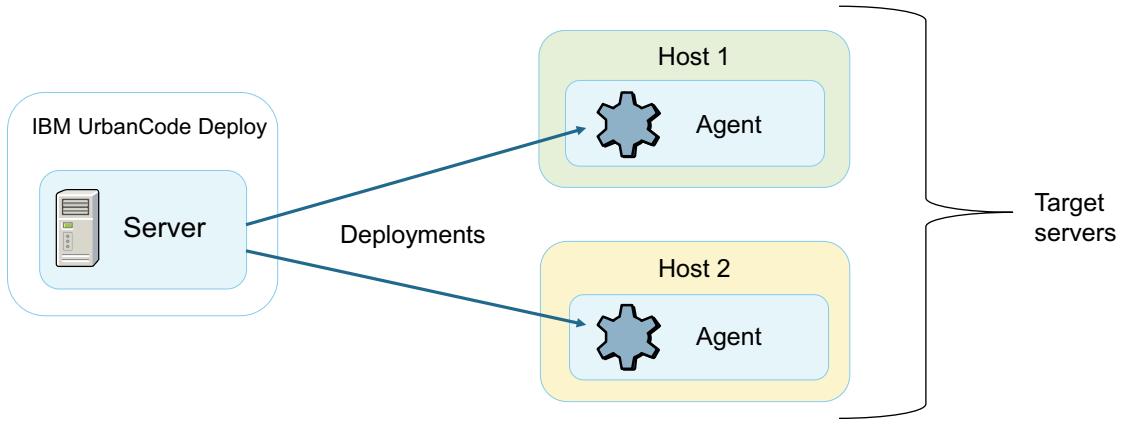
[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

An environment associates components with an agent on the target host

The environment is the “where” of the application model. First, you need to configure an agent. After that, you can create the resources and environments. Next, you associate the resources with the environments.

## An agent is a lightweight process that usually runs on a deployment-target host



[Unit 3: Defining deployments with resources, environments, and application processes](#)  
An agent is a lightweight process that usually runs on a deployment-target host

© Copyright IBM Corporation 2017

Agents are processes that run on the target environment. They communicate with the server and perform the automated steps that are defined in a component process.

Usually, an agent runs on the same host where the resources that it handles are located.

## Agents are installed from the user interface or the command prompt

The screenshot shows the 'Agents' tab selected in the 'Resources' section of the IBM UrbanCode Deploy interface. The page displays a table of agents with the following columns: Name, Description, Status, Date Created, Last Contact, License, Version, and Relay. Two agents are listed:

Name	Description	Status	Date Created	Last Contact	License	Version	Relay
agent-6892d1dcfe07		Online	12/14/2016, 5:20 PM	1/9/2017, 4:00 PM	Unlicensed	6.2.2.0.825094	
agent-bd0a7f9ef3f4		Offline	12/9/2016, 6:31 PM	12/15/2016, 3:25 PM	Unlicensed	6.2.2.0.825094	

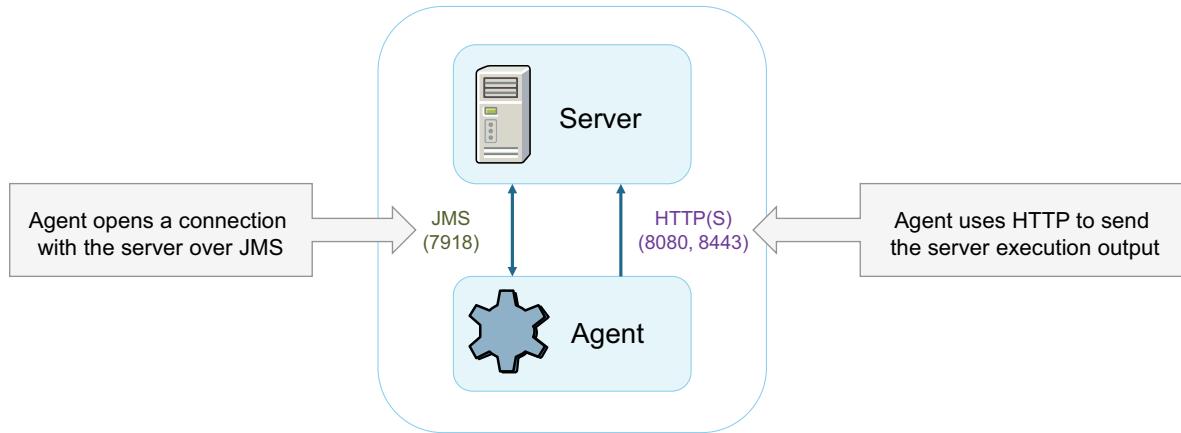
At the top of the page, there are buttons for 'Install New Agent', 'Discover Available Network Hosts', and 'Actions...'. A status bar at the bottom indicates 'Min. Recommended 6.2.2.0' and 'Min. Required 4.8.5'.

[Unit 3: Defining deployments with resources, environments, and application processes](#)  
Agents are installed from the user interface or the command prompt

© Copyright IBM Corporation 2017

Agents are installed with the installation scripts that are provided with the IBM UrbanCode Deploy installation files. When you bring an agent online for the first time, it creates a resource with the same name as the agent.

## Agents open direct connections to the server



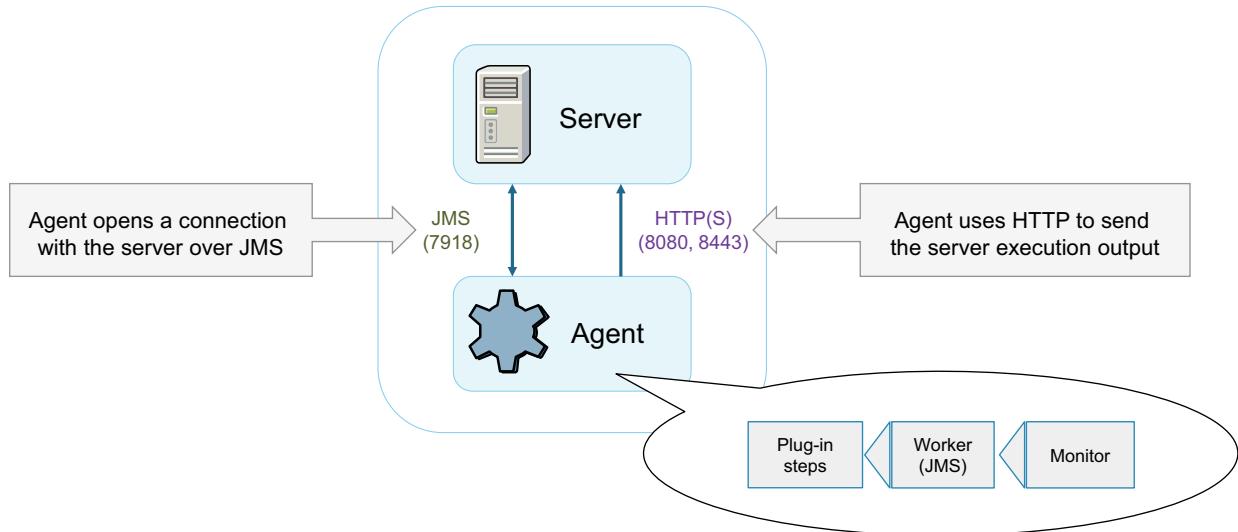
[Unit 3: Defining deployments with resources, environments, and application processes](#)  
*Agents open direct connections to the server*

© Copyright IBM Corporation 2017

When the agent is installed on the deployment target, it opens access to transfer files and execute commands. All processes that the IBM UrbanCode Deploy server requests, including packaging, configuration, and deployments, run on hardware that is assigned to agents.

Agent communications employ JMS, HTTP, and HTTPS protocols to communicate with the server. First, the agent uses JMS to establish a connection with the server. Over JMS, the server instructs the agent to run a plug-in step and provides a URL for HTTP connectivity. The agent downloads the required plug-in, runs the step, and then sends execution output back to the server over HTTP.

## An agent consists of a worker process and a monitor process



[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

An agent consists of a worker process and a monitor process

Although an agent is typically considered a single process, an agent consists of both a worker process and a monitor process.

The worker process runs the deployment work after it receives commands from the server. Work commands come from plug-in steps. The worker process uses JMS for system communications and HTTP REST services when it completes plug-in steps or retrieves information from the server.

The monitor process is a service that manages the worker process by starting and stopping, handling restarts, upgrades, and security. The agent monitor service uses JMS for all server communications and for sending commands, such as the run step command, to the worker process.

## Many agent features are managed from IBM UrbanCode Deploy

Name	Description	Status	Date Created	Last Contact	License	Version	Relay
agent-6892d1dcfe07		Online	12/14/2016, 5:20 PM	1/3/2017, 4:45 PM	Unlicensed	6.2.2.0.825094	
agent-bd0a7f9ef3f4		Offline	12/9/2016, 6:31 PM	12/15/2016, 3:25 PM	Unlicensed	6.2.2.0.825094	

[Unit 3: Defining deployments with resources, environments, and application processes](#)  
Many agent features are managed from IBM UrbanCode Deploy

© Copyright IBM Corporation 2017

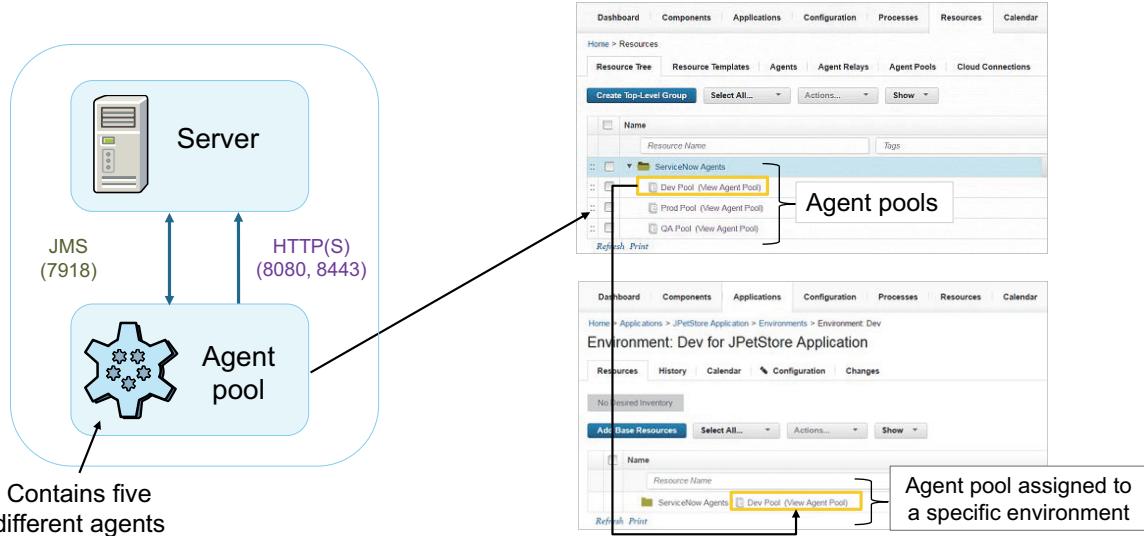
After an agent is installed, you can manage many of its features from the **Resources** tab with the monitor process. You can edit, restart, upgrade, test, deactivate, or delete the agent.

Agents must be online in order to run deployments.

In the status column, you view its state:

- **Offline:** No connection
- **Connecting:** JMS connectivity is available, but there is no HTTP connection. When an agent is in this state, the server can send a signal for upgrade or restart.
- **Online:** HTTP and JMS connectivity

## Agent pools manage agents that are installed in different environments



Unit 3: Defining deployments with resources, environments, and application processes

Agent pools manage agents that are installed in different environments

© Copyright IBM Corporation 2017

To provide relief to an overburdened agent, you can use an agent pool to spread the deployment processing work among participating agents. You assign agents to pools and pools are assigned to resources, just like lone agents. When an agent pool is assigned to a resource, work items are sent to eligible agents.

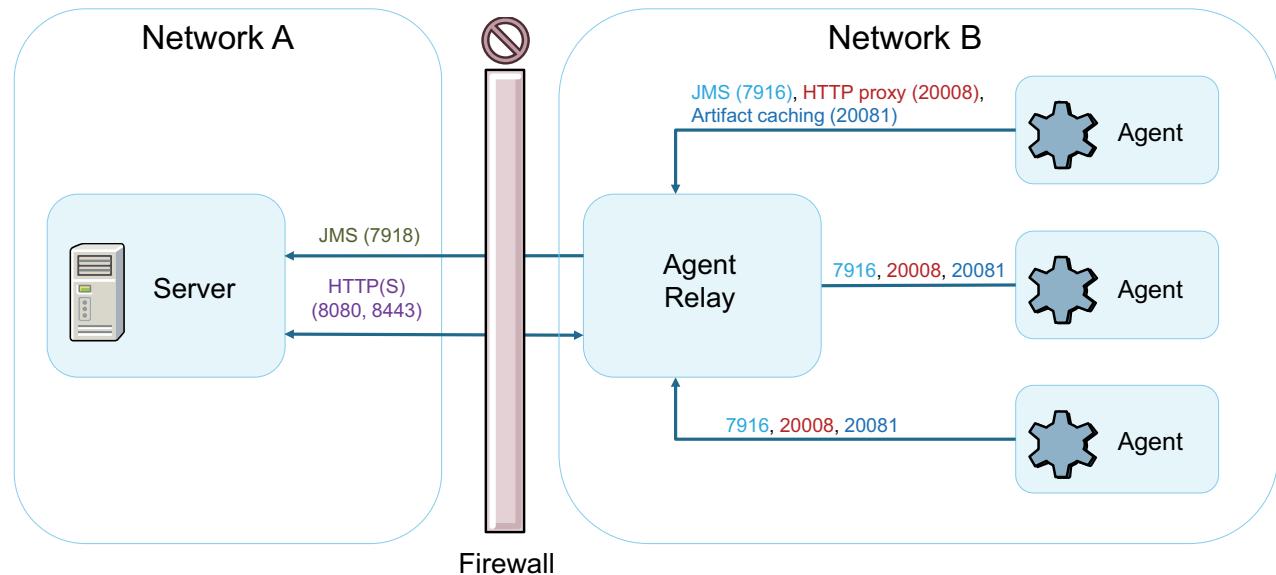
The **Resource Tree** tab shows a typical design of the structure of an agent pool. Notice the high-level folder called ServiceNow Agents. The folder contains three agent pools: Dev Pool, QA Pool, and Prod Pool. Agents are assigned within a pool specifically configured for the environment.

In the JPetStore environment graphic, notice that the Dev Pool agent pool is mapped to the Dev Environment. For each environment (QA and Prod), the Environments page displays the agent pools associated with the specified environment. With this configuration, the application process runs only on the available agents in the agent pool associated with the environment.



**Note:** The agent that does the actual work is chosen randomly from available online agents in the pool. The current workload that is running on the agents in the pool is not considered. The agent that was most recently selected to do work is not exempt from subsequent selections. The selected agent is always randomly selected from available online agents in the pool.

## Agent relays coordinate communication between agents and the server



Unit 3: Defining deployments with resources, environments, and application processes

© Copyright IBM Corporation 2017

Agent relays coordinate communication between agents and the server

In simple configurations, agents communicate directly with the UrbanCode server, as mentioned earlier. In more complex situations, this communication becomes inefficient or impossible. Some examples are when the number of agents increase from tens to hundreds to thousands, network topology becomes complex, or agent-server communication crosses firewalls with port restrictions.

The Agent Relay helps alleviate these challenges by consolidating agent traffic and communicating with remote agents.

- **They can connect large groups of agents to the primary server:** Instead of each agent connecting directly to the server, agents connect to agent relays, which are then connect directly to the server. Using agent relays in this way reduces load on the server because the server has fewer direct connections.
- **They can simplify communication across networks and firewalls:** If multiple agents are in a remote network without an agent relay, each agent must connect to the server individually. In this case, each agent must have network permission to connect to the server, including firewall permissions. With an agent relay, the agents connect only to the relay, and the relay is the only system that contacts the server directly, such as with a VPN or tunnel.

The diagram shows the main default ports that are involved in communication between agents, agent relays, and the server.

## Topics

- Managing components in applications
  - Configuring the agent as a deployment-target host
- ▶ **Organizing resources in a resource tree**
- Setting up deployment targets with environments
  - Coordinating deployments with application processes

---

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

*Topic: Organizing resources in a resource tree*

After installing an agent, you can identify the resources for the deployment.

## A resource tree organizes **what is being deployed and to where**

The screenshot shows the 'Resources' tab selected in the navigation bar. A callout arrow points from the text 'Top-level container' to the 'Create Top-Level Group' button, which is highlighted with a yellow box. A second callout arrow points from the 'Create Resource' dialog box to the 'Name' field, which contains the value 'Lab resources'. The dialog also includes fields for 'Description' and 'Teams'.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

A resource tree organizes what is being deployed and to where

Resources are created on the **Resource Tree** tab. Resources associate agents with components. After they are mapped to an application environment, they provide deployment targets.

## Resource groups enable collections of resources to be easily reused

The screenshot shows the 'Resources' section of the IBM UrbanCode Deploy web interface. A context menu is open over a resource named 'Lab resources'. The menu options include: Create Top-Level Group, Select All..., Actions..., Show, Compare or Synchronize, Define New Template, Synchronize With Template, Add From Template, Add Group (which is highlighted), Add Agent, Add Agent Pool, and Delete. A callout line points from the 'Add Group' option in the menu to a 'Create Resource' dialog box. The dialog box has fields for Name (SIT environment), Description, and Teams. It also includes checkboxes for 'Include Agents Automatically' and 'Inherit Teams From Parent'. Buttons for Save and Cancel are at the bottom.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

Resource groups enable collections of resources to be easily reused

First, you create a top-level group for all the resources within that group. A resource group is a useful organizational element. You can break down a complex environment into a number of blocks and manage them across hundreds of applications.

## Agent-type resources represent the computers where components are deployed

The screenshot shows the 'Resources' section of the IBM UrbanCode Deploy web interface. A context menu is open over a resource named 'SIT environment'. The menu options include: Compare or Synchronize, Define New Template, Synchronize With Template, Add From Template, Add Group, Add Agent (which is highlighted), Add Agent Pool, and Delete. A callout points from this menu to a larger 'Create Resource' dialog box. The 'Create Resource' dialog has the title 'Create Resource' with a question mark icon. It contains fields for 'Agent' (a dropdown menu showing 'agent-6892d1dcfe07' and 'agent-bd0a7f9ef3f4'), 'Description' (with the value 'agent-6892d1dcfe07'), 'Inherit Teams From Parent' (unchecked), 'Teams' (checkbox), and 'Default Impersonation' (checkbox). Below these fields is a note: 'Default impersonation can be configured here. Any steps which do not specify their own impersonation settings will fall back to the settings provided here.' At the bottom are 'Save' and 'Cancel' buttons.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

*Agent-type resources represent the computers where components are deployed*

Next, you add an agent. An agent lives on a resource and performs deployment operations.

## IBM Training



## Component-type resources represent the components that are deployed to target environments

The screenshot shows the 'Create Resource' dialog box open over a resource tree. The tree view on the left shows a hierarchy with 'Lab resources' and 'SIT environment' nodes. A context menu is open over the 'agent-6892d1dcfe07 (View Agent)' node, with the 'Add Component' option highlighted. The 'Create Resource' dialog has the following fields:

- Component:** JPetStore-WEB
- Name:** JPetStore-WEB
- Description:** (empty)
- Inherit Teams From Parent:** checked
- Teams:** (empty)
- Default Impersonation:** (checkbox)

Below the dialog, a note states: "Default impersonation can be configured here. Any steps which do not specify their own impersonation settings will fall back to the settings provided here."

At the bottom of the dialog, there are 'Save' and 'Cancel' buttons.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

Component-type resources represent the components that are deployed to target environments

Finally, you associate the components. A component-type resource can point directly to the agent or agent pool that deploys it, or it can point to another resource in a hierarchical chain of resources.

## The Resource Tree page displays resources in a hierarchical tree structure

The screenshot shows the 'Resource Tree' page in the IBM UrbanCode Deploy interface. The page has a top navigation bar with links like Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below the navigation is a breadcrumb trail: Home > Resources. The main area is titled 'Resource Tree' and contains a table with columns: Name, Inventory, Status, and Description. The table data is as follows:

Name	Inventory	Status	Description
Resource group			
Lab resources			
SIT environment			
agent-6892d1dcfe07 (View Agent)		Online	Agent
JPetStore-APP (View Component)			Components
JPetStore-DB (View Component)			
JPetStore-WEB (View Component)			

Annotations with arrows point from labels to specific rows: 'Resource group' points to the first row, 'Agent' points to the 'agent...' row, and 'Components' points to the last three rows under 'SIT environment'. There are also yellow boxes highlighting the 'SIT environment' row and the 'agent...' row.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

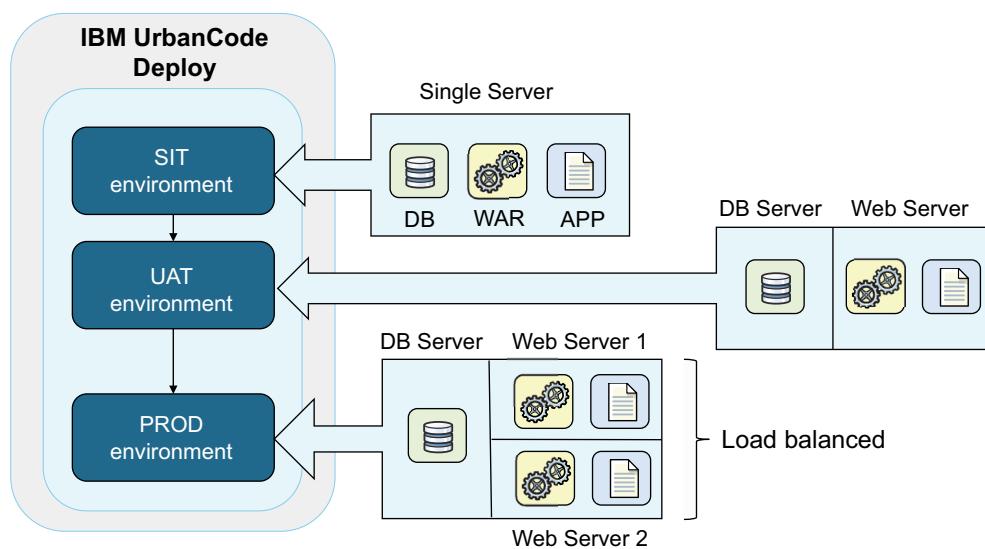
*The Resource Tree page displays resources in a hierarchical tree structure*

A resource tree shows a hierarchical representation of the mapping between logical and physical. Each row in a resource tree represents a resource and has an Action menu that is associated with it. The available actions depend on whether the resource represents an agent, agent pool, component, or organizational group.

## Topics

- Managing components in applications
  - Configuring the agent as a deployment-target host
  - Organizing resources in a resource tree
- ▶ **Setting up deployment targets with environments**
- Coordinating deployments with application processes

## Environments can have different topologies, configurations, and settings



[Unit 3: Defining deployments with resources, environments, and application processes](#)  
*Environments can have different topologies, configurations, and settings*

© Copyright IBM Corporation 2017

An environment is the application's mechanism for bringing together components with the agent that deploys them. Now that you have an agent with mapped components, you can create the environments and associate the resources. Environments typically include host systems and agents. When a deployment is run, it is always done so in an environment.

Environments can have different topologies.

The diagram shows three different server topologies:

- The SIT environment has a single server with an installed agent that contains all three components: Database, Web, and Configuration.
- The UAT environment has two servers with an installed agent on each. The database server contains the database component and the web server contains the web and configuration components.
- The PROD environment has a more distributed and production-ready topology. Each server has an installed agent with the database server that still contains the database component, but there are load-balanced web servers that each contain a web and configuration component.

## Environments represent the target to which your files are being deployed

The screenshot shows the IBM UrbanCode Deploy interface. On the left, a modal window titled 'Create Environment' is open, showing fields for Name (SIT environment), Blueprint, Teams, Require Approvals, Exempt Processes, and Lock Snapshots. A color palette is used to select a color for the environment. On the right, the main application window shows a list of environments: UAT environment (green), QA environment (purple), and SIT environment (blue). Each entry includes a snapshot status (None) and a compliance count (0 / 0).

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

Environments represent the target to which your files are being deployed

You create an environment for each target to which you deploy an application. Environments are typically modeled on some stage of the software project lifecycle, such as development, quality assurance, or production. The initial environment is typically uncontrolled and often is used to create snapshots. A snapshot represents a set of component versions that are known to work together. You look at snapshots later in this course.

Creating an environment involves selecting one or more agent resources with mapped components.

## Mapping resources and components to the environment provides a topology

The screenshot shows two main windows of the IBM UrbanCode Deploy application.

**Left Window:** Shows the 'Environments' list for the JPetStore application. It lists three environments: UAT environment, QA environment, and SIT environment. The SIT environment is highlighted with a blue selection bar. A black arrow points from the 'SIT environment' label to the 'Add Base Resources' button in the 'Add Resource to Environment' dialog.

**Right Window:** Shows the 'Environment: SIT environment for JPetStore' configuration page. The 'Add Resource to Environment' dialog is open, showing a tree view of resources under 'agent-6892d1dcfe07'. The tree includes 'Lab resources', 'SIT environment', and three components: 'JPetStore-APP', 'JPetStore-DB', and 'JPetStore-WEB'. A callout box labeled 'Agent with mapped components' points to this tree view.

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

Mapping resources and components to the environment provides a topology

Before you can run a deployment, you must define at least one application environment that associates components with an agent on the target host.

## Resources associate agents with components

IBM UrbanCode Deploy

Dashboard | Components | Applications | Configuration | Processes | Resources | Calendar | Work Items | Reports | Settings |

Home > Applications > JPetStore > Environments > Environment: SIT environment

Environment: SIT environment for JPetStore

Resources History Calendar Configuration Changes

No Desired Inventory

Add Base Resources Select All... Actions... Show Expand All Collapse All

Name	Inventory	Status	Description
Lab resources / SIT environment / agent-6892d10cfe07 (View Agent)		Online	
JPetStore-APP (View Component)			
JPetStore-DB (View Component)			
JPetStore-WEB (View Component)			

Refresh Print

Components that the agent will deploy to an environment

Agent

[Unit 3: Defining deployments with resources, environments, and application processes](#)  
Resources associate agents with components

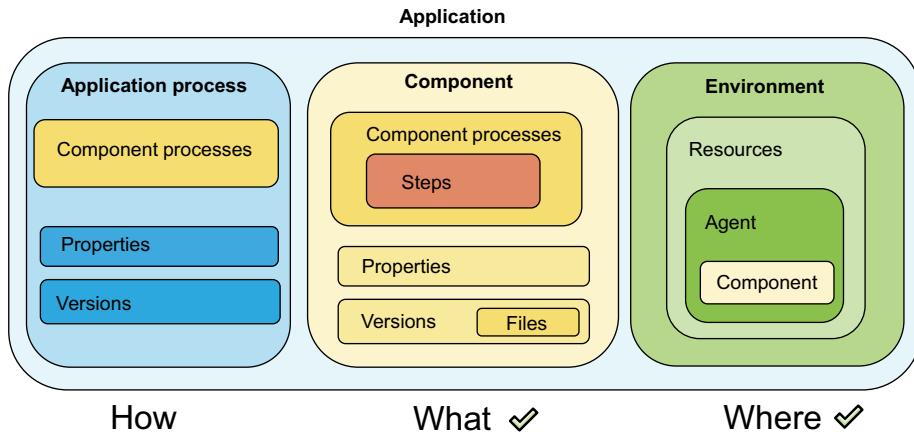
© Copyright IBM Corporation 2017

After components are mapped to an application environment, resources provide the deployment targets.

## Topics

- Managing components in applications
  - Configuring the agent as a deployment-target host
  - Organizing resources in a resource tree
  - Setting up deployment targets with environments
- ▶ Coordinating deployments with application processes

## An application process orchestrates the order of component process execution



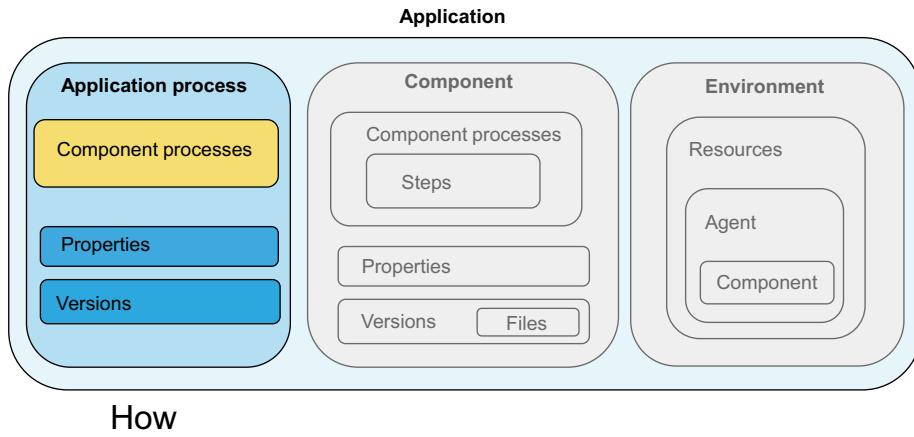
[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

*An application process orchestrates the order of component process execution*

In many cases, application processes call component processes. For example, an application process can call the component processes that deploy their associated components.

## The application process is the *how* of the application model

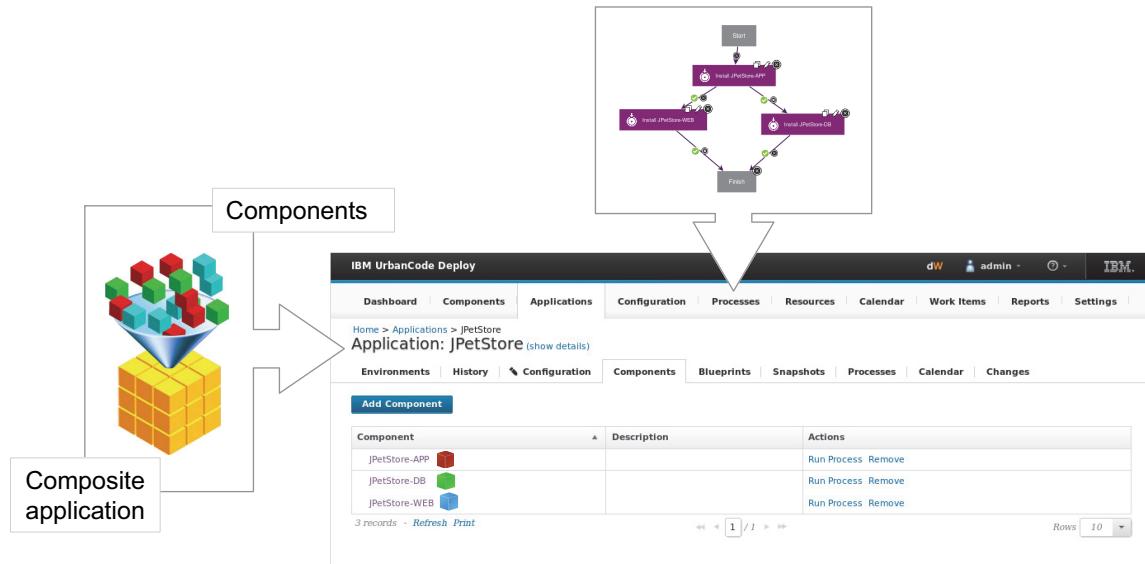


[Unit 3: Defining deployments with resources, environments, and application processes](#)  
*The application process is the how of the application model*

© Copyright IBM Corporation 2017

In this next section, you look at application processes, which describe the “how” in the application model.

## An application process determines what components are deployed together



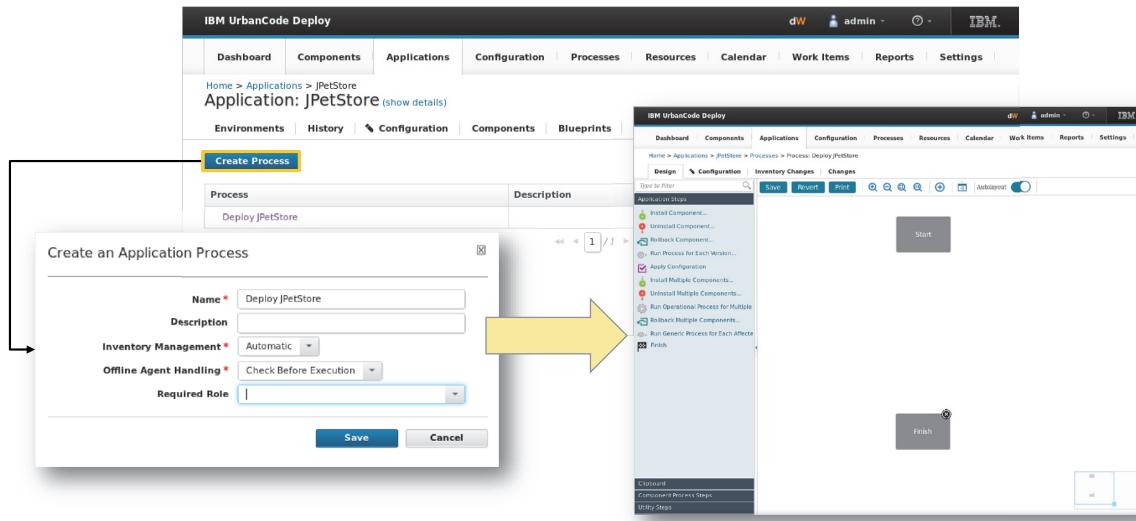
[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

An application process determines what components are deployed together

When you create application processes, you not only determine which components are grouped together, but you also identify the flow of the deployment. Component processes are run in a particular order as managed by the application process.

## Application processes are created with the process editor



[Unit 3: Defining deployments with resources, environments, and application processes](#)  
*Application processes are created with the process editor*

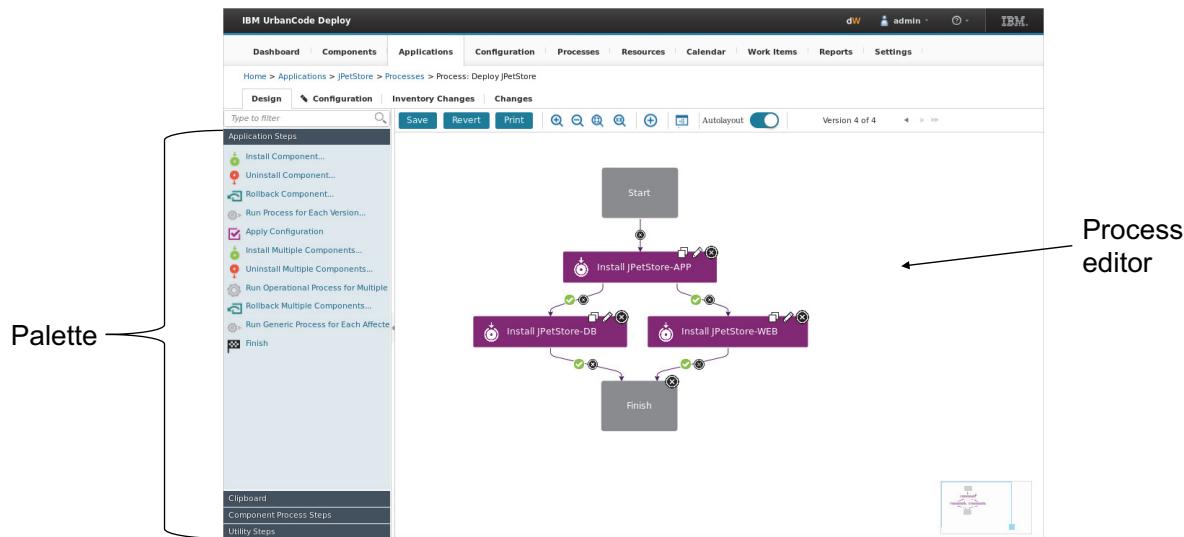
© Copyright IBM Corporation 2017

Application processes, like component processes, are created with the process editor. They are authored with a visual drag-and-drop editor and composed of steps that call the component processes.

Similar to component processes, you can use several step types, such as these examples:

- Conditional processes
- Switch steps
- Branching and joining steps
- Process step properties
- Post-processing preconditions

## Application processes provide overall deployment orchestration

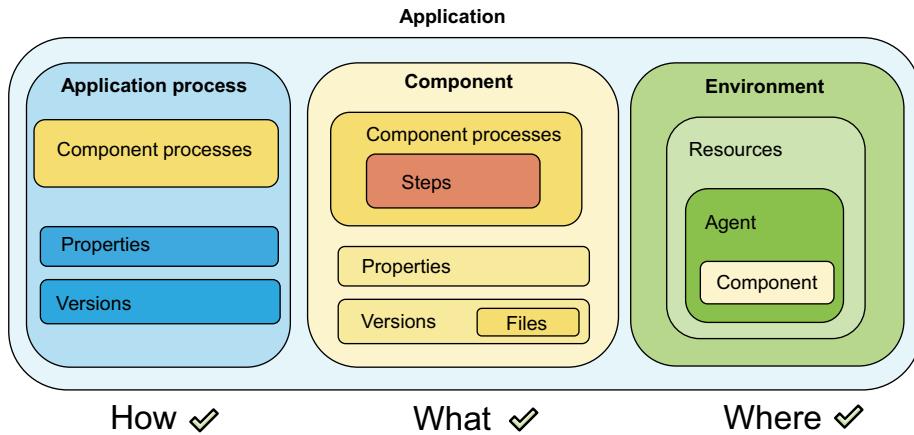


[Unit 3: Defining deployments with resources, environments, and application processes](#)  
*Application processes provide overall deployment orchestration*

© Copyright IBM Corporation 2017

An application process plays a coordination role. Steps in an application process represent the various component processes and tasks that are required to complete a deployment. While it looks like a component process, these steps are not from plug-ins. Steps for applications are part of IBM UrbanCode Deploy.

With all of the application pieces in place, you are ready for deployment



[Unit 3: Defining deployments with resources, environments, and application processes](#)  
With all of the application pieces in place, you are ready for deployment

© Copyright IBM Corporation 2017

## Unit summary

- An application is a container for components, processes, and resources
- Agents are processes that run on the target environment
- Environments represent the target to which your files are being deployed
- Application processes orchestrate the order of component process execution

## Exercises: Defining deployments with resources, environments, and application processes



- Create an application and associate components
- Create environments and associate base resources
- Create an application process to deploy components

---

[Unit 3: Defining deployments with resources, environments, and application processes](#)

© Copyright IBM Corporation 2017

*Exercises: Defining deployments with resources, environments, and application processes*

# **Unit 4 Deploying applications to target environments**

IBM Training



## **Deploying applications to target environments**

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

You run a deployment by running an application process in a target environment. In this unit, you learn how to deploy to an environment, create and deploy snapshots, and compare environments to view the inventory in each one.

## Objectives

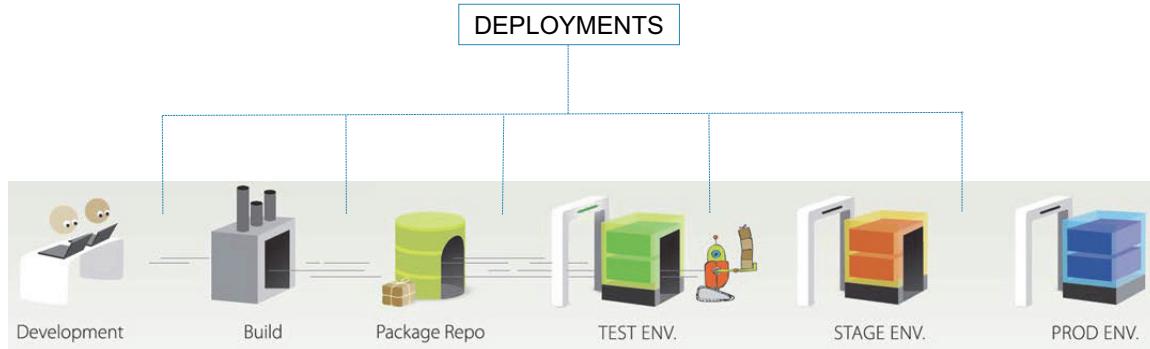
- Describe what happens in a deployment
- Describe the difference between full and incremental deployments
- Describe why and how to use snapshots
- View which versions are deployed to environments
- View the file differences between the environments
- Preview deployments for impact analysis
- Migrate changes from one environment to another

## Topics

### ▶ Deploying to an environment

- Examining application environment inventory
- Deploying applications with snapshots

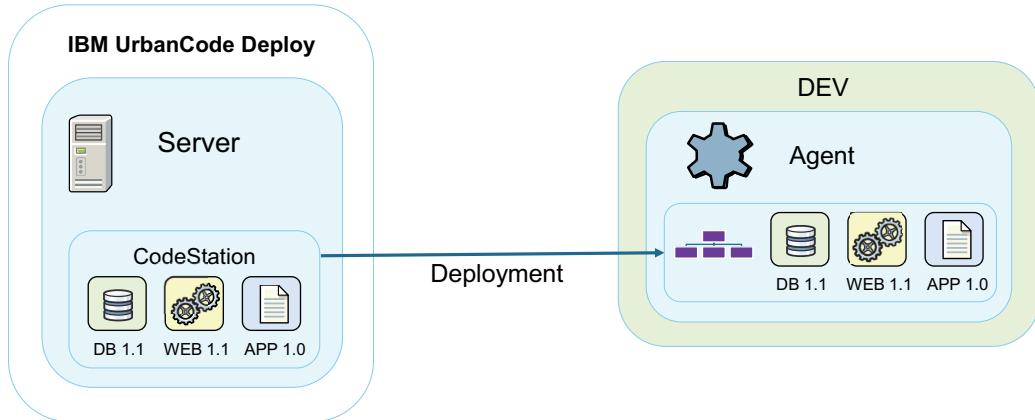
**A deployment is the promotion of an application from one environment to the next**



*A deployment is the promotion of an application from one environment to the next*

As mentioned in the first unit, a deployment is the promotion of an application from one environment to the next.

## A deployment configures and installs a deployment package in a specific environment



[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

*A deployment configures and installs a deployment package in a specific environment*

The IBM UrbanCode Deploy server provides a file transfer capability for deploying files to target servers. A deployment involves selecting component versions from a repository, moving those components to the target server, and completing the required tasks for bringing those parts of the application online.

## You run a deployment by running an application process in a target environment

The screenshot shows the IBM UrbanCode Deploy application interface. In the main window, under the 'Applications' tab, the 'JPetStore' application is selected. A modal dialog titled 'Run Process on SIT Environment' is open. Inside this dialog, the 'Process' dropdown is set to 'Deploy JPetStore'. A callout arrow points from the 'Component Versions' section of the main dialog to a separate modal window titled 'Component Versions'. This modal has a dropdown menu open with options: 'Select For All...', 'Latest Available' (which is highlighted), 'Versions With Name...', and 'Current Environment Inventory'. Below this is a table with three records:

	Current Environment Inventory	Versions to Deploy
None (Clear All)	None	Add...
JPetStore-WEB	None	Add...
JPetStore-DB	None	Add...

At the bottom right of the modal is an 'OK' button.

[Unit 4: Deploying applications to target environments](#)

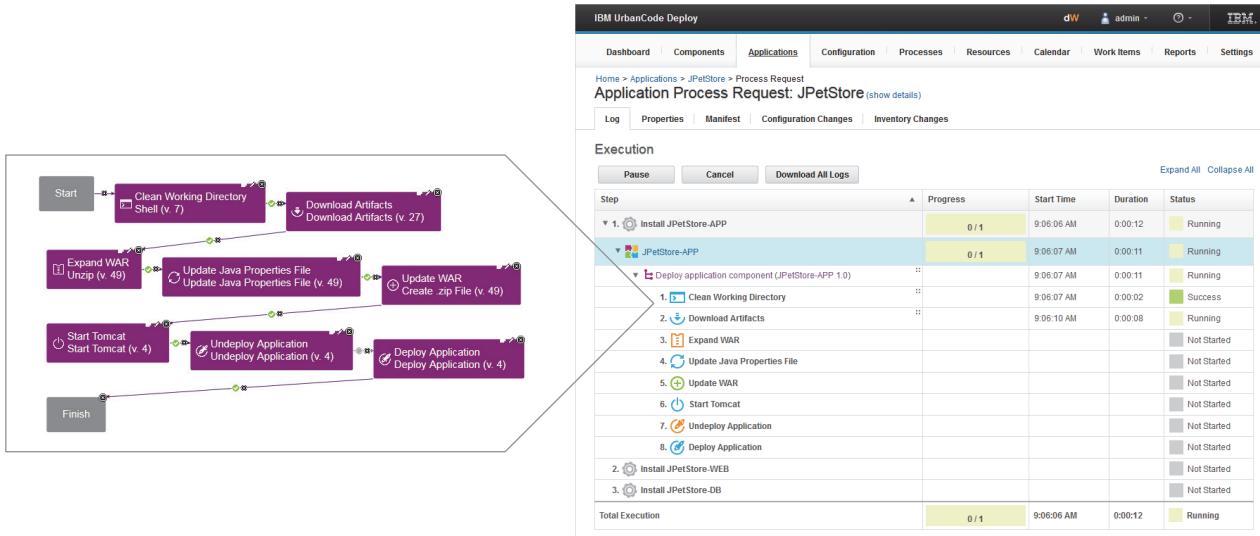
© Copyright IBM Corporation 2017

You run a deployment by running an application process in a target environment

When you start a deployment, you identify the application process and the component versions to deploy. You can select any process that is defined for the parent application, and you can use any components that are associated with the application.

If you want to deploy only the changed versions since the previous deployment, select the **Only Changed Versions** check box. If you clear this check box, all of the component versions that you select are deployed, even if they are already in the inventory.

## The application process request shows each step in the process



Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

*The application process request shows each step in the process*

When an application process starts, the Application Process Request page displays information about the deployment status. Notice that the steps in the log are the same as the component process.

## The Execution section displays the deployment details

**Execution**

Step	Start Time	Duration	Status
1. Install JPetStore-APP	9:06:06 AM	0:00:30	Success
2. Install JPetStore-WEB	9:06:37 AM	0:00:18	Success
3. Install JPetStore-DB	9:06:37 AM	0:00:15	Success
Total Execution	9:06:06 AM	0:00:49	Success

**Execution**

Step	Start Time	Duration	Status
1. Install JPetStore-APP	9:06:06 AM	0:00:31	Success
2. Download Artifacts	9:06:10 AM	0:00:09	Success
3. Install JPetStore-WEB	9:06:18 AM	0:00:15	Success
4. Install JPetStore-DB	9:06:33 AM	0:00:15	Success

[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

*The Execution section displays the deployment details*

After a process finishes, the Execution section shows the status. You can click the bars to get more details about the processes and steps.

## Topics

- Deploying to an environment
- ▶ Examining inventory and deployment information
- Deploying applications with snapshots

## Inventory and history are retained for environments and resources

Shows which component versions are deployed on the environment

Compliance shows whether actual component inventory matches expected inventory

Component	Version	Date Deployed	Compliance	Actions
JPetStore-WEB	1.1 (+ 1 more)	1/29/2017, 10:17 AM	Compliant (2/2)	<a href="#">View Request</a>
JPetStore-DB	1.1 (View Details)	1/29/2017, 10:17 AM	Compliant (1/1)	<a href="#">View Request</a>
JPetStore-APP	1.0 (View Details)	1/29/2017, 9:05 AM	Compliant (1/1)	<a href="#">View Request</a>

Component	Version	Date Deployed	Compliance	Actions
JPetStore-APP	1.0 (View Details)	1/23/2017, 10:28 AM	Compliant (1/1)	<a href="#">View Request</a>
JPetStore-DB	1.0 (View Details)	1/23/2017, 10:28 AM	Compliant (1/1)	<a href="#">View Request</a>
JPetStore-WEB	1.0 (View Details)	1/23/2017, 10:28 AM	Compliant (1/1)	<a href="#">View Request</a>

Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

### Inventory and history are retained for environments and resources

From the **Environments** tab of an application, you can see an inventory system. The inventory system tracks the specified state of each environment and what has been successfully deployed to each target resource. Compliancy indicates whether the environment contains those component versions.

The inventory and compliance for an environment are updated each time that you run an application process or deploy a snapshot to the environment.

In this example, three deployable components are deployed to two environments. You can see that the SIT environment has newer versions.

## You can compare inventories between environments

The screenshot shows two windows of the IBM UrbanCode Deploy application. The left window displays the 'Environments' list for the 'JPetStore' application, where the 'SIT Environment' is selected. A context menu is open over the 'Compare' button, with 'Copy' highlighted. An arrow points from this menu to a second window on the right, which is titled 'Environment Comparison: SIT Environment and UAT Environment'. This second window shows a table comparing components across two environments:

Component	Version	Status	SIT Environment	UAT Environment
JPetStore-APP	1.0	Active	✓	✓
JPetStore-DB	1.1	Active	✓	✓
JPetStore-DB	1.0	Active		✓
JPetStore-WEB	1.1	Active	✓	
JPetStore-WEB	1.0	Active	✓	✓

Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

You can compare inventories between environments

IBM UrbanCode Deploy maintains an inventory of every artifact that is deployed to each environment and also tracks the differences between them. You can compare two environments to see differences in the versions that they contain.

## File differences are tracked between the environments

The screenshot shows the IBM UrbanCode Deploy application interface. The top navigation bar includes 'Dashboard', 'Components', 'Applications' (which is selected), 'Configuration', 'Processes', 'Resources', 'Calendar', 'Work Items', 'Reports', and 'Settings'. The main content area is titled 'Environment Comparison: SIT Environment and UAT Environment'. Below this, there are three tabs: 'Versions', 'Files' (which is selected), and 'Configuration'. The 'Files' tab displays a 'File Difference Report for JPetStore-APP'. It shows a comparison between 'Environment: SIT Environment - Version: 1.0 (Last Modified)' and 'Environment: UAT Environment - Version: 1.0 (Last Modified)'. A message indicates 'No file differences found.' Below this, there are sections for 'File Difference Report for JPetStore-WEB' and 'File Difference Report for JPetStore-DB'. The 'JPetStore-DB' section shows a single record named 'upgrade\_sql\_1.xml' with a timestamp of '5/30/2012, 11:54 AM' and '5/30/2012, 11:58 AM'. The 'Actions' column for this record has a 'Compare' button. To the right of the main interface, a modal window titled 'File Differences' compares 'Version 1.1' and 'Version 1.0'. The 'Version 1.1' section contains XML code for a database schema and data. The 'Version 1.0' section also contains XML code, showing the changes made from version 1.0 to 1.1.

[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

### File differences are tracked between the environments

When there is a problem in production or during test, traceability makes it easier to understand what changes are going in that could have caused the problem.

## The History tab show the process requests for a specific environment

The screenshot shows the IBM UrbanCode Deploy interface. The top navigation bar includes links for Dashboard, Components, Applications (which is selected), Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below the navigation is a breadcrumb trail: Home > Applications > JPetStore > Environments > Environment: SIT Environment. The main content area is titled "Environment: SIT Environment for JPetStore". Underneath, there are tabs for Resources, History (which is active and highlighted with a yellow box), Calendar, Configuration, and Changes. A table lists three process requests:

Process	Snapshot	Scheduled For	Status	Actions
Deploy JPetStore		1/29/2017, 10:17 AM	Success	<a href="#">View Request</a>
Deploy JPetStore		1/29/2017, 9:06 AM	Success	<a href="#">View Request</a>
Deploy JPetStore		1/22/2017, 10:12 PM	Success	<a href="#">View Request</a>

Below the table, it says "3 records - Refresh Print". To the right, there is a "Rows" dropdown set to "10". A callout box with a black border and white text points to the "View Request" link in the first row's Actions column, with the text "Opens process request" inside.

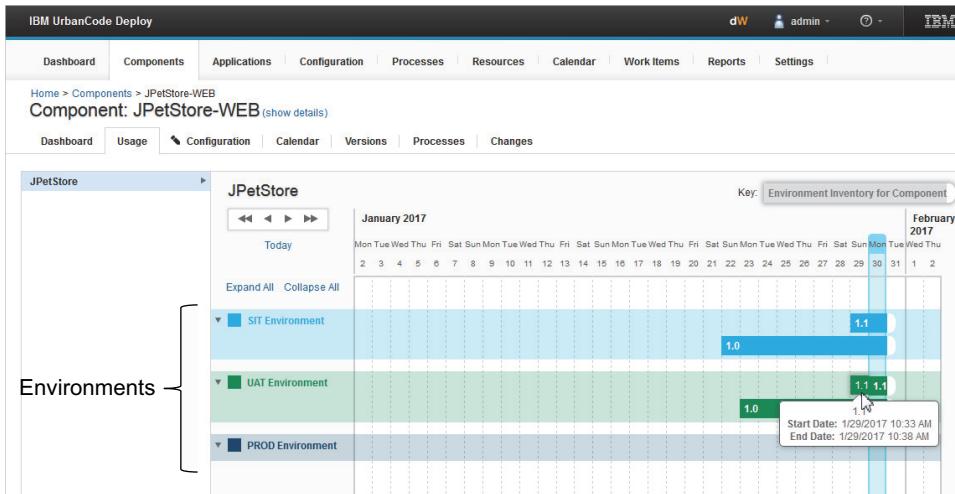
[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

*The History tab show the process requests for a specific environment*

You can view the progress of the process by going to the **History** tab and opening the process request.

## The Usage tab shows the environment inventory for each component



Shows for how long a component was deployed

Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

*The Usage tab shows the environment inventory for each component*

You can track the environment inventory for each component by clicking the component and then going to the **Usage** tab. IBM UrbanCode Deploy tracks both the version that is currently in an environment and also what was in various environments over time. This feature is helpful comparing flows through environments and identifying what changed.

## Deployment reports contain historical information about deployments

The screenshot shows the IBM UrbanCode Deploy interface. On the left, a sidebar titled 'Deployment' lists options: Deployment Count, Deployment Average Duration, Deployment Total Duration, Deployment Detail (which is selected and highlighted in blue), Security, and My Reports (0). The main area is titled 'Deployment > Deployment Detail'. It features a 'Date Range' section with dropdowns for 'Current Month' (set to 1/1/2017) and '1/31/2017'. Below this are filters for 'Application' (JPetStore), 'Environment' (UAT Environment), 'User' (Any), 'Status' (Success), and 'Plugin' (Any). There are 'Run' and 'Save' buttons. To the right is a table titled 'Selected Columns' with columns: Application, Environment, Date, User, Status, Duration, and Actions. The table contains four rows of deployment data for JPetStore in UAT Environment. At the bottom are navigation buttons for page 1 of 1 and a 'Rows' dropdown set to 10.

Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

### Deployment reports contain historical information about deployments

You can filter data, print and save reports, and save search criteria with deployment reports. You can obtain information for specific reporting periods, such as the number of deployments, average deployment duration, and total duration. In this example, the Deployment Detail report shows all deployments for the specified applications, environments, and users for a certain time period.

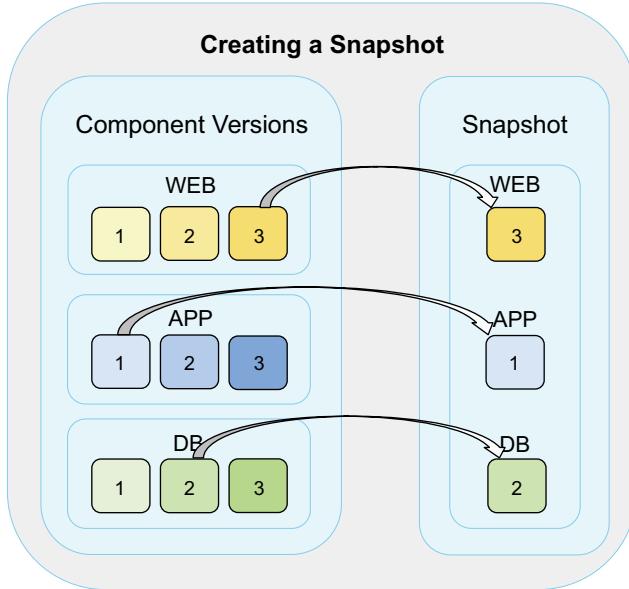
- **Deployment count:** Provides information about the number of deployments that ran during a specified reporting period.
- **Deployment average duration:** Provides average deployment times for applications that ran during a specified reporting period.
- **Deployment total duration:** Provides total deployment times for applications that ran during a specified reporting period.
- **Deployment detail:** Shows all deployments for the specified applications, environments, and users in the specified time period.

## Topics

- Deploying to an environment
- Examining application environment inventory

▶ Deploying applications with snapshots

## A snapshot is collection of specific versions of components and processes



Unit 4: Deploying applications to target environments

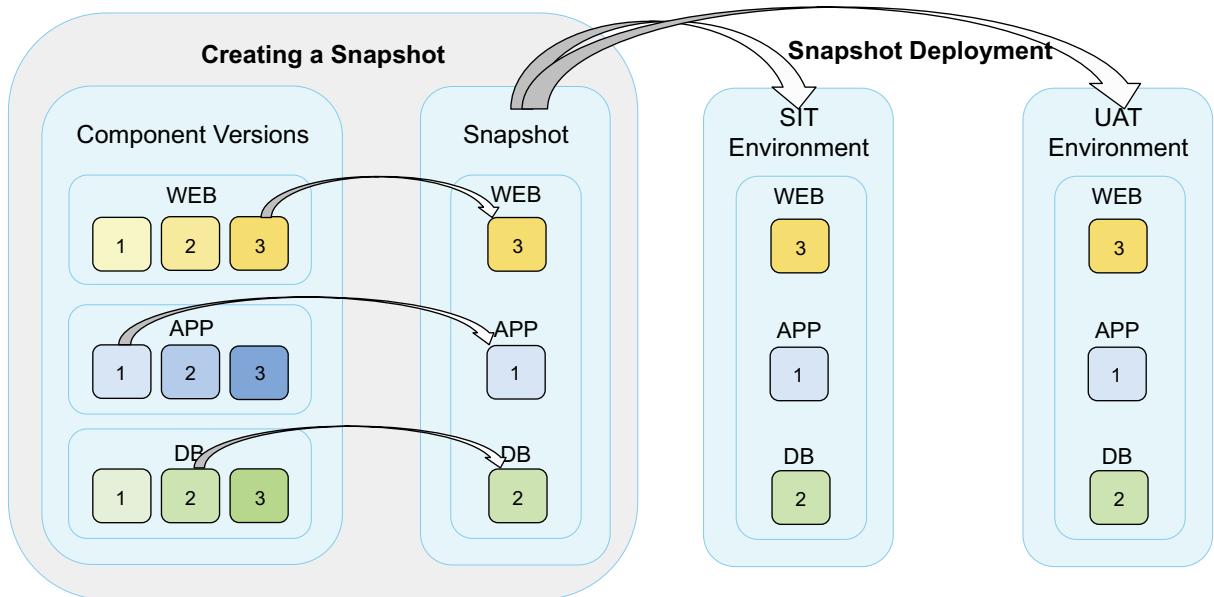
© Copyright IBM Corporation 2017

A snapshot is collection of specific versions of components and processes

Snapshots are models that you create before you deploy the application. A snapshot specifies the exact version for each component in the application. When a snapshot is created, information is gathered about the application, including the component versions, for an environment.

Typically, the snapshot is generated in an environment that has no approval gates. This kind of environment is called an uncontrolled environment.

## Snapshots ensure consistent deployments from one environment to another



Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

*Snapshots ensure consistent deployments from one environment to another*

You use snapshots to ensure the same exact deployment happens from one environment to another. If you have several components, selecting a version for each one, each time, could introduce errors. A snapshot is a single deployable item that you can push from environment to environment. After all the appropriate stages and approvals for a snapshot are complete, the snapshot is pushed to production.

## A snapshot captures a picture of the application's current state

The screenshot shows two main windows of the IBM UrbanCode Deploy application.

**Main Application Window:** Shows the 'Applications' tab selected for the 'JPetStore' application. It lists environments: 'SIT Environment' (Snapshot: None, Compliant: 4/4) and 'Create Snapshot Environment' (Snapshot: None, Compliant: 3/3). A yellow arrow points from the 'Create Snapshot Environment' row to a callout box.

**Create Snapshot Dialog:** A modal window titled 'Create Snapshot' is open. It has fields for 'Name' (set to 'SIT Promote') and 'Description'. There are 'Save' and 'Cancel' buttons at the bottom. A yellow arrow points from this dialog to the second application window.

**Second Application Window:** Shows the 'Situations' tab for the 'SIT Promote' snapshot. It displays component versions for 'JPetStore-APP', 'JPetStore-DB', and 'JPetStore-WEB'. Each component has three version entries: '1.0 X', '1.1 X', and '1.0 X'. A yellow arrow points from the 'Create Snapshot' dialog to this window.

Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

*A snapshot captures a picture of the application's current state*

To create a snapshot, you choose the component versions and processes to include in the snapshot.

## Previewing deployments for impact analysis

The screenshot shows the IBM UrbanCode Deploy interface. On the left, under 'Environments', the 'UAT Environment' is selected and highlighted in green. A yellow arrow points from this selection to a 'Preview Deployment' dialog box in the center. This dialog box has 'Process \* Deploy\_JPetStore' selected. Below it are 'OK' and 'Cancel' buttons. To the right of the dialog, the 'Deployment Preview' section is expanded, showing two tables: 'Changes Per Resource' and 'Changes Per Component'. Both tables list changes for 'JPetStore-DB' and 'JPetStore-WEB' components.

Resource/Component	Process	Version	Change Type
JPetStore-DB	Deploy DB Component	1.1	+ Active
JPetStore-WEB	Deploy web component	1.1	+ Active

Component/Resource	Process	Version	Change Type
JPetStore-DB	Deploy DB Component	1.1	+ Active
JPetStore-WEB	Deploy web component	1.1	+ Active

[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

### Previewing deployments for impact analysis

You can preview an application snapshot before you deploy it to see the upcoming changes in resources and properties.

## Deploying a snapshot deploys the components in the snapshot

The screenshot shows the IBM UrbanCode Deploy application interface. At the top, there's a navigation bar with links like Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below that, it says "Home > Applications > JPetStore Application: JPetStore (show details)". Under the "Components" tab, there's a sub-menu with Environments, History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. A "Create Environment" button is visible. On the left, there's a list of environments: SIT Environment (Snapshot: SIT Promote) and UAT Environment (Snapshot: None). A "Request Process" button is also present. A modal dialog box titled "Run Process on UAT Environment" is open in the center. It contains fields for "Process" (set to "Deploy.JPetStore"), "Snapshot" (set to "SIT Promote"), and other options like "Only Changed Versions" and "Schedule Deployment?". At the bottom of the dialog are "Submit" and "Cancel" buttons.

Unit 4: Deploying applications to target environments

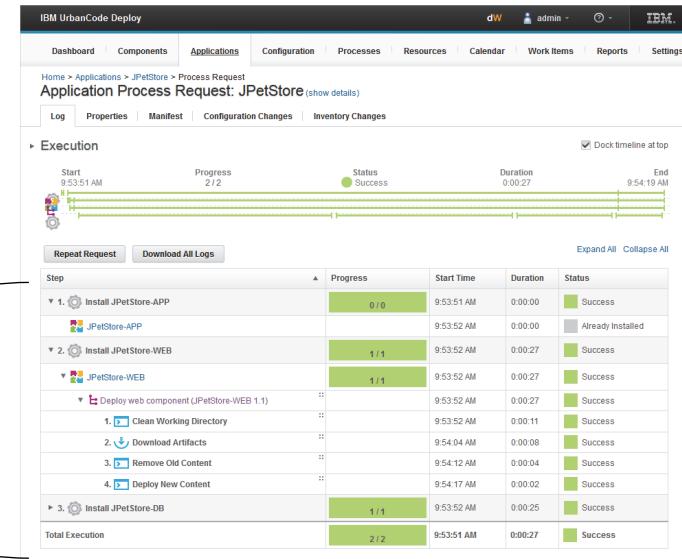
© Copyright IBM Corporation 2017

Deploying a snapshot deploys the components in the snapshot

When you have a snapshot, you select both the environment to deploy to and the process to use for the deployment.

## Application and component processes are the same

Same steps from the application and component processes



[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

*Application and component processes are the same*

The deployment looks the same as the original deployment each time the snapshot is used.

## The inventory is updated after running the snapshot

Snapshot created from components and processes in the SIT environment

Snapshot deployed to UAT environment

SIT Environment			
Component	Version	Date Deployed	Compliance
JPetStore-WEB	1.1 (+ 1 more)	1/29/2017, 10:17 AM	Compliant (2/2) <a href="#">View Request</a>
JPetStore-DB	1.1 ( <a href="#">View Details</a> )	1/29/2017, 10:17 AM	Compliant (1/1) <a href="#">View Request</a>
JPetStore-APP	1.0 ( <a href="#">View Details</a> )	1/29/2017, 9:06 AM	Compliant (1/1) <a href="#">View Request</a>

UAT Environment			
Component	Version	Date Deployed	Compliance
JPetStore-WEB	1.1 (+ 1 more)	1/30/2017, 9:53 AM	Compliant (2/2) <a href="#">View Request</a>
JPetStore-DB	1.1 ( <a href="#">View Details</a> )	1/30/2017, 9:53 AM	Compliant (1/1) <a href="#">View Request</a>
JPetStore-APP	1.0 ( <a href="#">View Details</a> )	1/23/2017, 10:28 AM	Compliant (1/1) <a href="#">View Request</a>

[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

*The inventory is updated after running the snapshot*

The inventory is immediately updated and can be viewed alongside of other environments in the application.

## Environments can be configured to lock snapshot versions or configuration

The screenshot shows two side-by-side configurations in the IBM UrbanCode Deploy application.

**Left Configuration:** This section is titled "Locks component versions". It shows a table of components and their versions. A yellow box highlights the "Lock Versions" button. A callout box points to this button with the text "Locks component versions".

Component	Versions
JPetStore-APP	1.0 [X] Add...
JPetStore-DB	1.1 [X] Add...
JPetStore-WEB	1.0 [X] 1.1 [X] Add...

**Right Configuration:** This section is titled "Locks configuration". It shows a table of items and their current status. A yellow box highlights the "Lock Configuration" button. A callout box points to this button with the text "Locks configuration".

Name	Version	Commit	Actions
Applications			
Resources			
Components			
System			

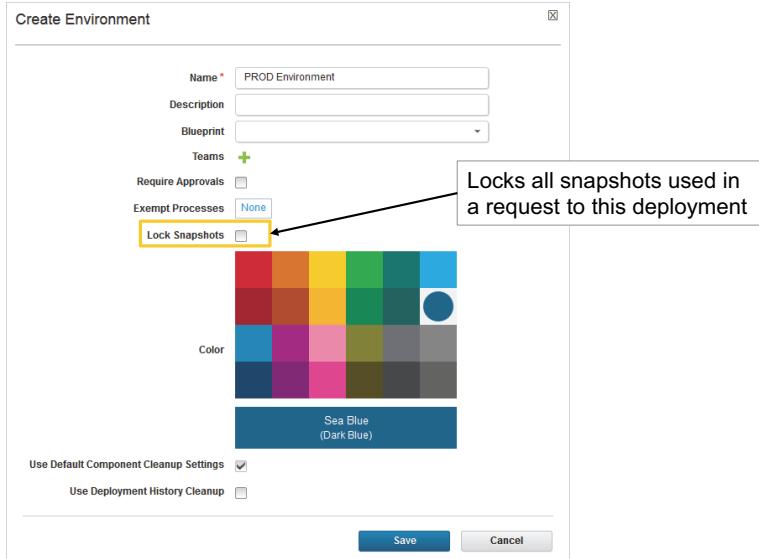
Unit 4: Deploying applications to target environments

© Copyright IBM Corporation 2017

Environments can be configured to lock snapshot versions or configuration

Environments can be configured to specifically lock either snapshot versions or the snapshot configuration, or both, when snapshots are deployed to the environment.

## You can lock a snapshot at environment creation



[Unit 4: Deploying applications to target environments](#)

© Copyright IBM Corporation 2017

*You can lock a snapshot at environment creation*

By default, the lock setting for both component versions and snapshot configurations are determined by the option selected in the Lock Snapshots field on the Create Environment window. All snapshots used in a request to this deployment will be locked to prevent further changes.

## Unit summary

- A deployment is the promotion of an application from one environment to the next
- You run a deployment by running an application process in a target environment
- You can compare two environments to see differences in the versions that they contain
- Snapshots are models that you create before you deploy the application
- A snapshot specifies the exact version for each component in the application

## Exercises: Deploying applications to target environments



- Run an application process to deploy components
- Deploy specific component versions to update the application
- Create a snapshot and deploy it to a new environment



# **Unit 5 Setting up server security, approvals, and quality gates**

IBM Training



## **Setting up server security, approvals, and quality gates**

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

IBM UrbanCode Deploy uses a flexible team-based and role-based security model that maps to your organizational structure. In this unit, you learn how to define and configure roles, set up approvals and notifications, and use quality statuses and gates.

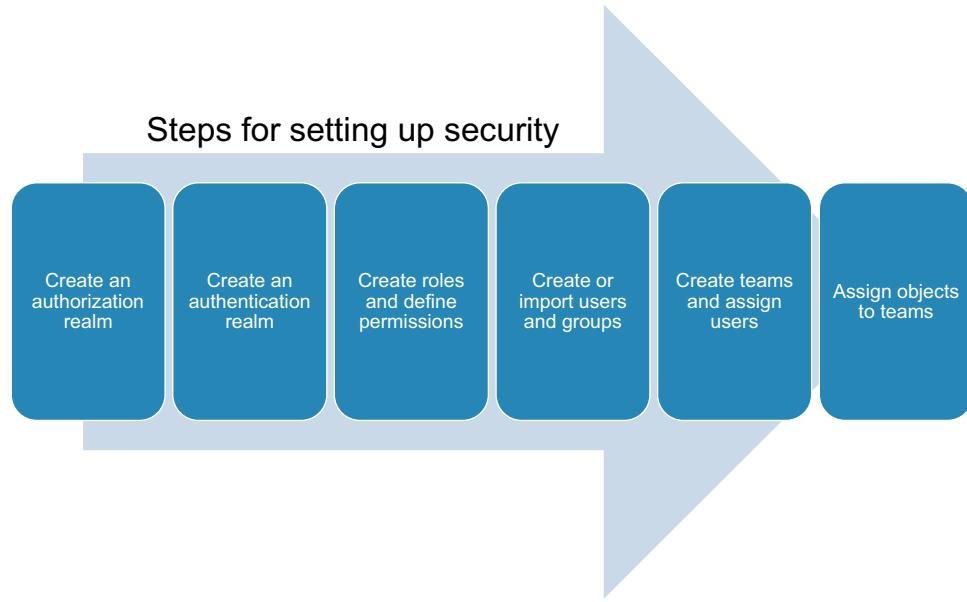
## Objectives

- Describe the significance of a system team
- Create authorization realms, user groups, and teams
- Create and define roles and permissions
- Set up notifications and approvals
- Implement statuses and gates to ensure testing quality

## Topics

- ▶ Configuring server security
  - Setting up notifications
  - Setting up an approval process
  - Implementing statuses and gates to improve quality

**The security system for the server consists of an authentication realm, authorization realm, roles, and teams**



[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

*The security system for the server consists of an authentication realm, authorization realm, roles, and teams*

From a high level, the security system for the server consists of an authorization realm, authentication realm, roles, and teams.

The authentication realm verifies the identity of the user or system that is trying to log on to the IBM UrbanCode Deploy server.

The authorization realm manages user groups. Roles manage permissions. Teams bring together users with roles and specify which objects the team can access.

**You can access the security system from any of these links on the Settings tab**

The screenshot shows the IBM UrbanCode Deploy interface. At the top, there's a navigation bar with links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Settings link is highlighted. Below the navigation bar, the main content area has three columns: Automation, Security, and System. The Security column is highlighted with a yellow box around its title and the first five items: API Keys And Certificates, Authentication (Users), Authorization (Groups), Teams, and Tokens. The Automation column contains links for Automation Plugins, Source Configuration Plugins, Running Version Imports, Locks, Blueprint Designer Integrations, Post Processing Scripts, and Statuses. The System column contains links for Logging, Network, Notification Schemes, Output Log (Download), Audit Log, Diagnostics (Download), Patches, Properties, CodeStation, and System Settings.

---

Unit 5: Setting up server security, approvals, and quality gates

© Copyright IBM Corporation 2017

**You can access the security system from any of these links on the Settings tab****You access the security features from the **Settings** tab.**

## Authorization realms manage user groups

The screenshot shows the 'Authorization Realms' section of the IBM UrbanCode Deploy interface. On the left, there's a sidebar with 'Authorization Realms' and 'Groups' options. The main area shows a table with one record: 'Internal Security' with a description of 'Internal database storage.' Below the table are buttons for 'Refresh' and 'Print'. A modal window titled 'Create Authorization Realm' is open, prompting for 'Name' (set to 'New Realm'), 'Description', and 'Type'. The 'Type' dropdown is currently set to 'LDAP or Active Directory', which is highlighted with a yellow box and has a mouse cursor pointing at it. Other options shown are 'Internal Storage' and 'SSO'. At the bottom of the modal are 'Save' and 'Cancel' buttons.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

### Authorization realms manage user groups

Authorization realms verify the identity of a user or system attempting to login. There are three available types for the server, internal storage authorization, an LDAP authorization realm, and an SSO authorization realm:

- **Internal storage authorization:** Does not retrieve users from any external source. Instead, you add users to internal storage authorization realms manually.
- **LDAP authorization realm (Lightweight Directory Access Protocol):** LDAP is a widely used protocol for accessing distributed directory information over IP networks. It uses an external LDAP server for authorization.
- **SSO authorization realm (single sign-on):** Uses an external server for authorization and allows a user to sign on with one set of credentials for multiple applications.

## A role is a set of permissions

The screenshot shows the 'Role Configuration' tab in the 'Security' section of the IBM UrbanCode Deploy interface. A 'Developer' role is selected. A callout points from the word 'Role' to the 'Developer' button. Another callout points from the word 'Objects' to the list of objects on the left, which includes Agent, Agent Pool, Agent Relay, Application, Application Template, Cloud Connection, Component, Component Template, Environment, Environment Template, Process, Resource, Resource Template, Server Configuration, and Web UI. A third callout points from the word 'Permissions' to the table titled 'Permissions Granted to Role Members'. The table lists two records: Standard Environment and QA Environment. For Standard Environment, 'Execute on Environments' is checked, while 'View Environments', 'Create', and 'Edit' are unchecked. For QA Environment, 'View Environments' is checked, while 'Execute on Environments', 'Create', and 'Edit' are unchecked.

Type	Execute on Environments	View Environments	Create	Edit
Standard Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Create	<input type="checkbox"/> Edit
QA Environment	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> Create	<input type="checkbox"/> Edit

An authentication realm is a source of information about users

Authentication realms manage users and determine user identity within authorization realms for the server. Users can be created manually or imported from external systems, such as LDAP.

## Groups identify users with the same role

The screenshot shows the 'Groups' page in the IBM UrbanCode Deploy interface. The main table lists two groups: 'Administrators' and 'Developers'. The 'Developers' group is selected, and a callout box highlights the 'Developers' tab in the breadcrumb navigation and the list of members. Two members, 'Ben (BenFranklin)' and 'Ken', are listed with 'Remove' actions. The interface includes standard navigation buttons like Refresh and Print.

Group	Authorization Realm	Actions
Administrators	Internal Security	
Developers	Internal Security	

Groups / Developers

User	Actions
Ben (BenFranklin)	
Ken	

Group name: Developers  
Group members: Ben (BenFranklin), Ken

A role is a set of permissions

Permissions define what can be done; roles define who can do it. Separate permissions are available for each type of object on the server, including components, applications, and environments.

## An authentication realm is a source of information about users

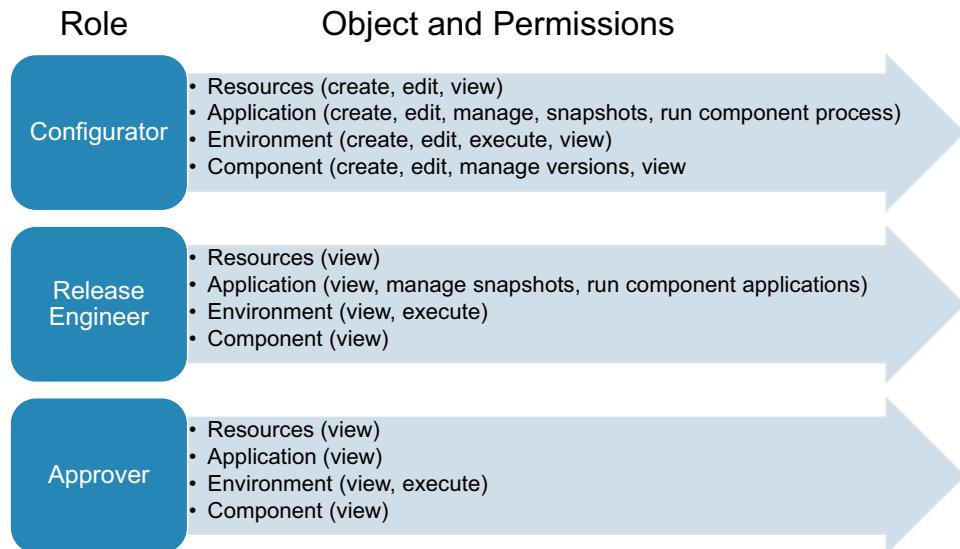
The screenshot shows the 'Internal Security' section of the 'Users' management page. A yellow box highlights the 'Internal Security' link in the left sidebar and the table of users below. The table has columns for User, Name, Email, and Actions. The 'Actions' column contains links for Edit, Reset Password, and Remove. The table shows four records: BenFranklin, Jenn, Ken, and admin.

User	Name	Email	Actions
BenFranklin	Ben		Edit Reset Password Remove
Jenn	Jenn		Edit Reset Password Remove
Ken	Ken		Edit Reset Password Remove
admin			Edit Reset Password

*Groups identify users with the same role*

Groups are containers that grant permissions to multiple users; members automatically share a group's permissions.

## Keep the roles simple but ensure sufficiency for performing work



[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

*Keep the roles simple but ensure sufficiency for performing work*

When defining the roles for your team, start by keeping the roles simple, but make them sufficient to perform the appropriate work.

## Users are granted permissions by being assigned to teams

The screenshot shows the 'Teams' tab selected in the navigation bar. The main panel displays the 'JPetStore Team' configuration, including its name and role members. A callout box highlights that users assigned to the 'Developer' role will inherit its permissions. A separate dialog box shows a list of users for selection, with one user checked.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

*Groups grant permissions to multiple users*

You can grant a group of users the same role and permissions by dragging the group to the role.

## Groups grant permissions to multiple users

The screenshot shows the 'JPetStore Team' configuration page. In the 'Role Members' section, the 'Administrators' group is selected. On the right, the 'Users & Groups' table shows the 'Administrators' group selected. A yellow box highlights the 'Administrators' group in both sections.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

Users are granted permissions by being assigned to teams

You can also assign individuals to teams to give them the proper permissions.

Teams provide two important functions:

- First, a team provides the mechanism to assign roles to users. When a user is assigned a role, all permissions that are granted to the role are automatically granted to the user, but only for objects that the team has access to.
- Second, teams secure objects such as applications, components, and environments. When a team is attached to an object, only team members with the appropriate permissions can interact with the affected resource.

## Restrict permissions by assigning teams to objects

The screenshot shows the 'Team Object Mappings' section of the IBM UrbanCode Deploy interface. In the 'Team Object Mappings' table, there is one row for 'JPetStore' under the 'Standard Application' type. A modal dialog is overlaid on the page, containing a search bar with 'Enter text to filter...', a checkbox for 'Clear Filter', and a checked checkbox for 'JPetStore' which says '1 selected'. Buttons for 'Clear Selection', 'OK', and 'Reset' are also present.

Name	Types
JPetStore	Standard Application

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

### Restrict permissions by assigning teams to objects

You can assign an object, such as an application, to a team in the Team Object Mappings section.

## Security reports provide information about user roles and privileges

The screenshot shows the IBM UrbanCode Deploy interface with the title "IBM UrbanCode Deploy". The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports (which is selected), and Settings. The left sidebar has a "Reports" section with categories: Deployment, Security, User Security, Resource Security, Application Security (selected), Component Security, and Environment Security. The main content area is titled "Security > Application Security" and contains a table with the following data:

Application	Run Component Processes	Create	Delete	Edit Basic Settings	View	Manage Blueprints	Manage Components	Manage Environments
JPetStore	BenFranklin Ken Jenn admin	Jenn admin	Jenn admin	Jenn admin	BenFranklin Ken Jenn admin	Jenn admin	Jenn admin	Jenn admin

Below the table, it says "1 record" and "1 / 1 Rows 10".

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

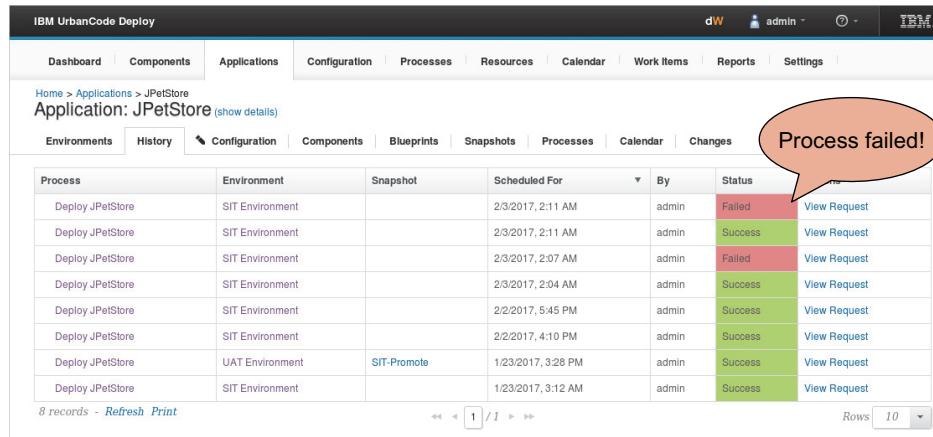
*Security reports provide information about user roles and privileges*

Security reports track information about roles and permissions. The application security report shows the user roles and permissions that are set for applications.

## Topics

- Configuring server security
- ▶ **Setting up notifications**
  - Setting up an approval process
  - Implementing statuses and gates to improve quality

## Notifications can be sent when user-defined trigger events occur



A screenshot of the IBM UrbanCode Deploy application interface. The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The Applications menu is currently selected. Below the navigation is a breadcrumb trail: Home > Applications > JPetStore. The main content area shows a table titled "Application: JPetStore (show details)". The table has columns for Process, Environment, Snapshot, Scheduled For, By, and Status. A single row in the table is highlighted with a red background in the "Status" column, which contains the text "Failed". A red speech bubble is overlaid on this row, containing the text "Process failed!". At the bottom of the table, there are links for Refresh and Print, and a page navigation section showing "1 / 1". To the right, there is a "Rows" dropdown set to "10".

Process	Environment	Snapshot	Scheduled For	By	Status
Deploy JPetStore	SIT Environment		2/3/2017, 2:11 AM	admin	Failed
Deploy JPetStore	SIT Environment		2/3/2017, 2:11 AM	admin	Success
Deploy JPetStore	SIT Environment		2/3/2017, 2:07 AM	admin	Failed
Deploy JPetStore	SIT Environment		2/3/2017, 2:04 AM	admin	Success
Deploy JPetStore	SIT Environment		2/2/2017, 5:45 PM	admin	Success
Deploy JPetStore	SIT Environment		2/2/2017, 4:10 PM	admin	Success
Deploy JPetStore	UAT Environment	SIT-Promote	1/23/2017, 3:28 PM	admin	Success
Deploy JPetStore	SIT Environment		1/23/2017, 3:12 AM	admin	Success

*Notifications can be sent when user-defined trigger events occur*

Notifications are emails that are sent whenever user-defined trigger events on the server occur, such as when a deployment finishes or an approval is required.

**IBM Training**

**IBM UrbanCode Deploy requires an external SMTP mail server to send notifications**

The screenshot shows the IBM UrbanCode Deploy interface with the 'Settings' tab selected. On the left, there's a navigation sidebar with sections like Automation, Security, and System. Under 'System', the 'System Settings' option is highlighted with a yellow box. The main content area displays various configuration options under 'General Settings', including fields for 'External Agent URL' (http://192.168.27.100:8080) and 'Internal User URL' (http://192.168.27.100:8080). Other settings include 'Enable Maintenance Mode', 'Validate Agent IP', 'Skip Property Updates for Existing Agents', 'Enable UI Debugging', 'Require a Commit for Process Design Changes', 'Create Default Children for Plugins', 'Enable Deployed Version Snapshot', 'Default Locale' (English (United States)), 'Default Snapshot Lock Type' (Component Versions and Configuration), and 'Automatic Version Import Check Period (seconds)' (300000). To the right, there are sections for 'WinRS Agent Install Settings' and 'Mail Server Settings'. The 'Mail Server Settings' section is highlighted with a yellow box and contains fields for 'Mail Server Host' (smtp.example.com), 'Mail Server Port' (25), 'Secure Mail Server Connection' (disabled), 'Mail Server Sender Address' (sender@example.com), 'Mail Server Username' (username), and 'Mail Server Password' (\*\*\*\*\*). Below these are sections for 'Security' with fields for 'Minimum Password Length' (0) and 'Require Complex Passwords' (disabled).

Unit 5: Setting up server security, approvals, and quality gates

© Copyright IBM Corporation 2017

*IBM UrbanCode Deploy requires an external SMTP mail server to send notifications*

Before you can send notifications, users must have email addresses attached to them on the server; email addresses are not automatically imported.

## Notifications are sent to appropriate parties based on scheme

The screenshot shows two views of the IBM UrbanCode Deploy interface. On the left, the 'Settings' pane is open, displaying categories like Automation, Security, and System. In the 'System' section, the 'Notification Schemes' item is highlighted with a yellow box. On the right, a larger window titled 'Create Notification Scheme' is displayed, showing a table with columns for 'Notification Scheme', 'Description', and 'Actions'. The table contains one record: 'Default Notification Scheme'.

*Notifications are sent to appropriate parties based on scheme*

To set up notifications, open the Notifications pane by clicking **Settings > Notification Schemes**.

## Notifications include triggering events and roles

The screenshot shows the 'Notification Scheme: Default Notification Scheme' page in the IBM UrbanCode Deploy web interface. On the left, there is a table titled 'Notification Entries' with columns: Type, Role, Template, and Actions. A message at the top of the table says, 'This scheme has no notification entries.' Below the table is a button labeled 'Add Notification Entry'. A callout arrow points from this button to a modal window titled 'Add Notification Entry'. This modal contains fields for 'Role' (set to 'Administrator'), 'Event Type' (set to 'Process Success'), 'Target' (set to 'Application'), 'Type' (set to 'Standard Application'), and 'Template Name' (set to 'ApplicationDeploymentSuccess'). At the bottom of the modal are 'Save' and 'Cancel' buttons. A yellow arrow points from the 'Save' button in the modal to the main 'Notification Entries' table on the right, indicating that the new entry has been successfully added. The table now shows one record: 'Process Success' under 'Type', 'Application / Administrator' under 'Role', 'ApplicationDeploymentSuccess' under 'Template', and 'Edit Delete' under 'Actions'.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

### Notifications include triggering events and roles

When you set up the notification, you determine which users receive notification by selecting both the triggering events and the role. The role is inherited from the security system.

## Notifications are sent to appropriate parties based on scheme

The screenshot shows the 'Basic Settings' tab for the 'JPetStore' application. The 'Notification Scheme' dropdown is highlighted with a yellow box and has a context menu open, showing 'None', 'Default Notification Scheme', and 'None' again.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

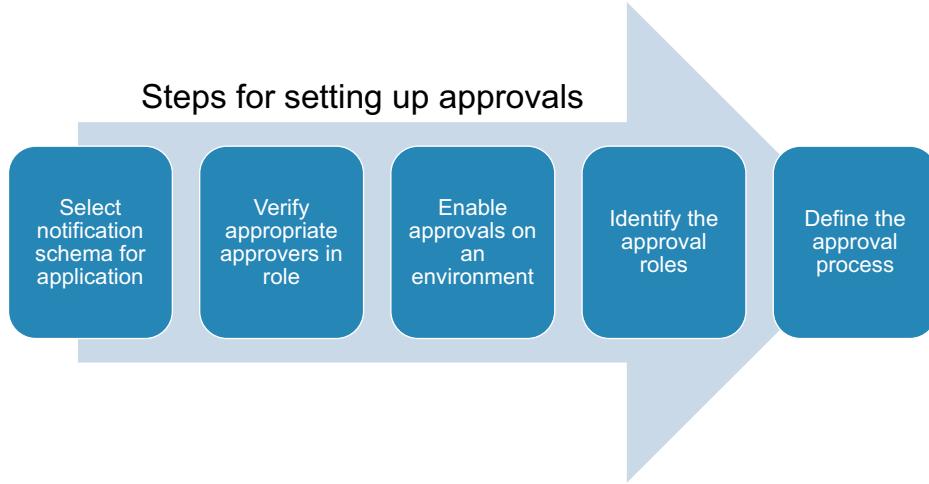
*Notifications are sent to appropriate parties based on scheme*

You can assign notification schemes to applications and environments at creation or in the **Configuration** tab.

## Topics

- Configuring server security
  - Setting up notifications
-  **Setting up an approval process**
- Implementing statuses and gates to improve quality

## Approvals provide more visibility into deployments for audit trails



*Approvals provide more visibility into deployments for audit trails*

An approval specifies the job that needs approval and the role of the approver. The main tasks for setting up an approval process include selecting the notification scheme, enabling approvals on an environment, and defining the approval process.

## IBM Training



## An approval process specifies the job that needs approval and the role of the approver

**Select to require approvals**

**Select approver and notification type**

Unit 5: Setting up server security, approvals, and quality gates

© Copyright IBM Corporation 2017

An approval process specifies the job that needs approval and the role of the approver

To ensure that components cannot be deployed to the environment without first being approved, select the **Require Approvals** check box. This option enforces an approval process before the deployment can be deployed to the environment. Approvals are usually attached to environments.

## IBM Training



**When a request for approval is made, the users with the corresponding role are notified through email**

The screenshot shows the IBM UrbanCode Deploy interface. In the main window, there is a table titled 'Approval Progress' with one row:

Task	Target	Required Role	Start Time	Status	Completed By	Actions
1. Environment Approval	PROD2 Environment	Administrator	1/31/2017, 5:21 PM	Open		<b>Respond</b>

An arrow points from the 'Respond' button in the main interface to the 'Approve' radio button in a modal window that has popped up. The modal window contains the following fields:

- Response:  Approve  Reject
- Comment: Ready for production
- Submit
- Cancel

In the second screenshot, the status of the task has changed to 'Complete'.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

*When a request for approval is made, the users with the corresponding role are notified through email*

The approver can approve or reject a deployment and also provide comments to the decision.

## The Deployment Details report tracks approval status

The screenshot shows two overlapping IBM UrbanCode Deploy application windows. The left window displays the 'Approval Progress' for a process request titled 'Application Process Request: JPetStore'. It lists one task: '1. Environment Approval' (Status: Failed) with 'Completed By: admin (1/31/2017, 5:33 PM)'. The right window shows the 'Deployment Details' report, which lists deployment events for the JPetStore application across various environments (PROD2, SIT, UAT). One specific deployment entry for PROD2 environment on 1/31/2017 at 5:32 PM is highlighted with a yellow box and labeled 'APPROVAL\_REJECTED' in the Status column.

Application	Environment	Date	User	Status
JPetStore	PROD2 Environment	1/31/2017, 5:29 PM	admin	SUCCESS
JPetStore	PROD2 Environment	1/31/2017, 5:32 PM	admin	APPROVAL_REJECTED
JPetStore	SIT Environment	1/22/2017, 10:12 PM	admin	SUCCESS
JPetStore	UAT Environment	1/23/2017, 10:28 AM	admin	SUCCESS
JPetStore	SIT Environment	1/26/2017, 10:17 AM	admin	SUCCESS
JPetStore	SIT Environment	1/26/2017, 9:56 AM	admin	SUCCESS
JPetStore	UAT Environment	1/26/2017, 10:38 AM	admin	SUCCESS
JPetStore	UAT Environment	1/30/2017, 9:53 AM	admin	SUCCESS
JPetStore	UAT Environment	1/28/2017, 10:33 AM	admin	SUCCESS

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

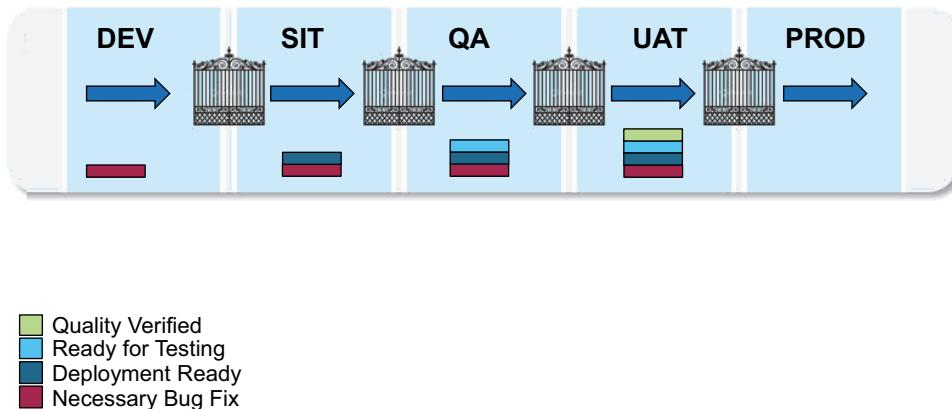
*The Deployment Details report tracks approval status*

You can see the status of the approvals by clicking **Reports > Deployment Details**.

## Topics

- Configuring server security
  - Setting up notifications
  - Setting up an approval process
-  [Implementing environment gates to improve quality](#)

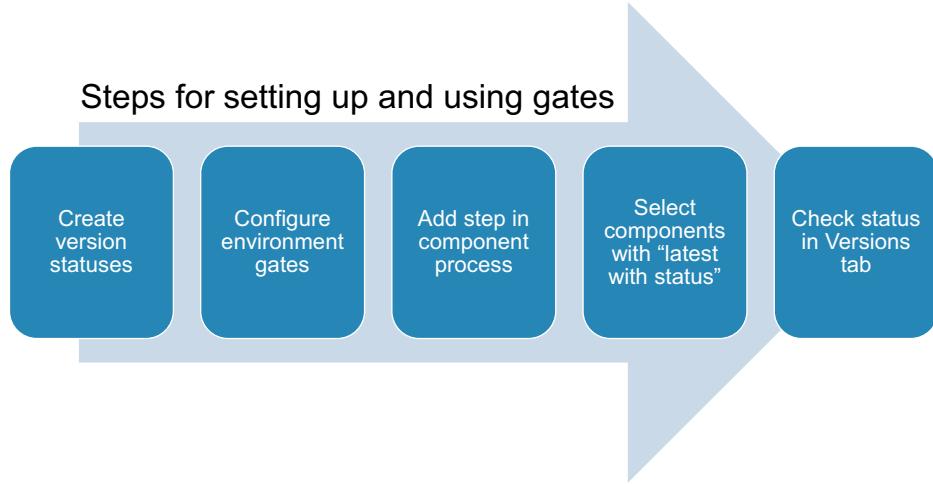
## Gates specify the conditions that must be met for deployment



Gates specify the conditions that must be met for deployment

An environment gate is a requirement that must be met before component versions can be deployed to an environment. A version status adds a tag to a version to indicate that it has met those requirements and can be deployed to the environment.

## Using statuses and gates requires initial configuration



[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

### Using statuses and gates requires initial configuration

When you set up gates, you first need to create the version statuses and configure the environment gates. Then, you add a step in the component process to check the version status. When you select the components for your deployment, choose **Latest with Status** before deployment. This choice deploys only those versions that have met the appropriate status.

## Version statuses identify when component versions meet criteria

The screenshot shows the IBM UrbanCode Deploy interface with the 'Statuses' tab selected. On the left, there's a sidebar with 'Automation' options like Automation Plugins, Source Configuration Plugins, Running Version Imports, Locks, Blueprint Designer Integrations, Post Processing Scripts, and Statuses (which is highlighted with a yellow box). The main content area shows two tables: 'Inventory Statuses' and 'Version Statuses'. The 'Version Statuses' table has three rows: Deployment Ready (blue), Necessary Bug Fix (red), and Ready for Testing (blue, currently selected). A modal dialog titled 'Edit Status' is open over the table, showing the 'Ready for Testing' status being edited. The dialog fields include Name (Ready for Testing), Description (Version is ready to be tested in UAT), Color (Bright Cerulean (Blue)), Unique (unchecked), Required Role (None), and Save/Cancel buttons.

[Unit 5: Setting up server security, approvals, and quality gates](#)

© Copyright IBM Corporation 2017

*Version statuses identify when component versions meet criteria*

On the **Statuses** tab, a table shows the currently configured version statuses. When you create a status, you can set the required role field as a specific role to restrict the ability to assign a status to a version.

## Environment gates check the version tags on component versions

The screenshot shows the IBM UrbanCode Deploy interface. In the top navigation bar, 'Configuration' is selected. Under 'Configuration', the 'Components' tab is active. On the left sidebar, 'Environment Gates' is selected. The main panel displays 'Environment Gates' settings for the 'JPetStore' application. It shows two environment rows: 'SIT Environment' and 'UAT Environment'. The 'UAT Environment' row has a status dropdown set to 'Ready for Testing'. A callout box points to this row with the text: 'Choose an environment that requires gates, and add conditions.'

[Unit 5: Setting up server security, approvals, and quality gates](#)

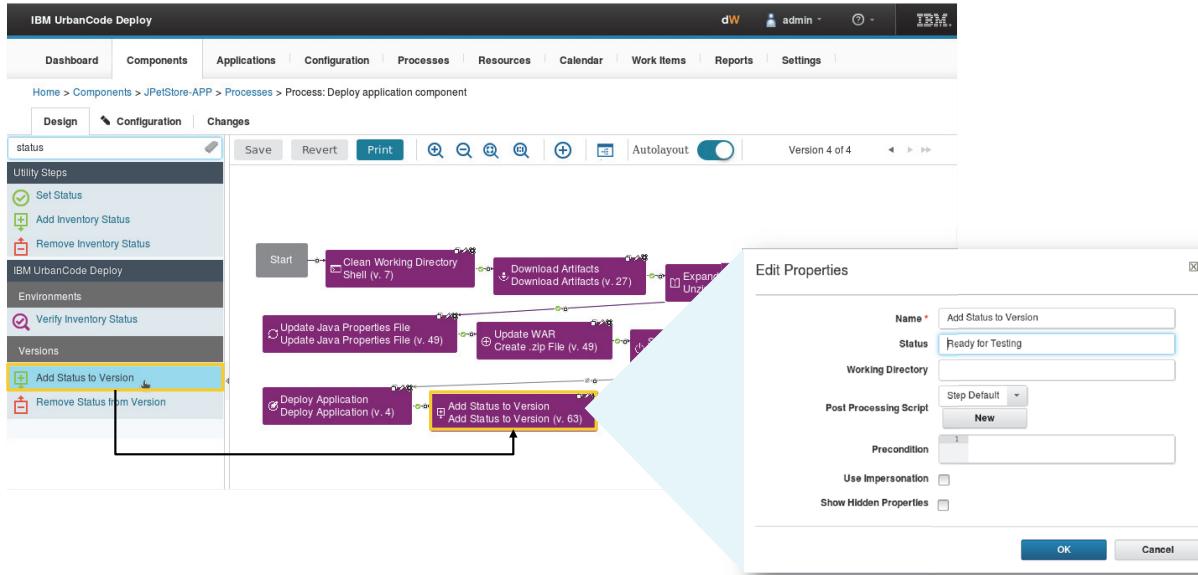
© Copyright IBM Corporation 2017

### Environment gates check the version tags on component versions

After you have your statuses configured, you can set up the application gates on your environments from the application's **Configuration** tab.

For each environment, you can set version statuses that the version must have before it can be deployed to that environment. In this example, only versions that have the status Ready for Testing are allowed to be deployed to the UAT environment.

## Statuses can be added to versions automatically from component processes



Unit 5: Setting up server security, approvals, and quality gates

© Copyright IBM Corporation 2017

Statuses can be added to versions automatically from component processes

You can add statuses to versions in component processes using the Add Status to Version step in the IBM UrbanCode Deploy Versions plug-in.

## Adding version statuses to component versions becomes part of the process

Execution				
Step	Progress	Start Time	Duration	Status
▼ 1. ⚙ Install JPetStore-APP	1 / 1	2:33:18 PM	0:00:23	Success
▼ JPetStore-APP	1 / 1	2:33:18 PM	0:00:23	Success
▼ Deploy application component (JPetStore-APP 1.0)	::	2:33:18 PM	0:00:23	Success
1. 🗃 Clean Working Directory	::	2:33:19 PM	0:00:02	Success
2. 📥 Download Artifacts	::	2:33:21 PM	0:00:04	Success
3. 📦 Expand WAR	::	2:33:25 PM	0:00:02	Success
4. 🛡 Update Java Properties File	::	2:33:27 PM	0:00:02	Success
5. 🛡 Update WAR	::	2:33:29 PM	0:00:02	Success
6. 🔍 Start Tomcat	::	2:33:32 PM	0:00:02	Success
7. 🛡 Undeploy Application	::	2:33:34 PM	0:00:02	Success
8. 🛡 Deploy Application	::	2:33:37 PM	0:00:02	Success
9. 📜 Add Status to Version	::	2:33:39 PM	0:00:02	Success
▶ 3. ⚙ Install JPetStore-DB	0 / 1	2:33:42 PM	0:00:03	Running
▶ 2. ⚙ Install JPetStore-WEB	0 / 1	2:33:42 PM	0:00:03	Running
Total Execution	1 / 3	2:33:18 PM	0:00:27	Running

Adding version statuses to component versions becomes part of the process

Now the component process includes adding a version status.

## Version statuses are tracked in each component

The screenshot shows the IBM UrbanCode Deploy interface. The top navigation bar includes links for Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The user is currently viewing the 'Components' section for 'JPetStore-DB'. A specific version, 'Version: 1.0 (show details)', is selected. Below this, there are tabs for Main, Configuration, and History. The 'Statuses' section contains a table with one row:

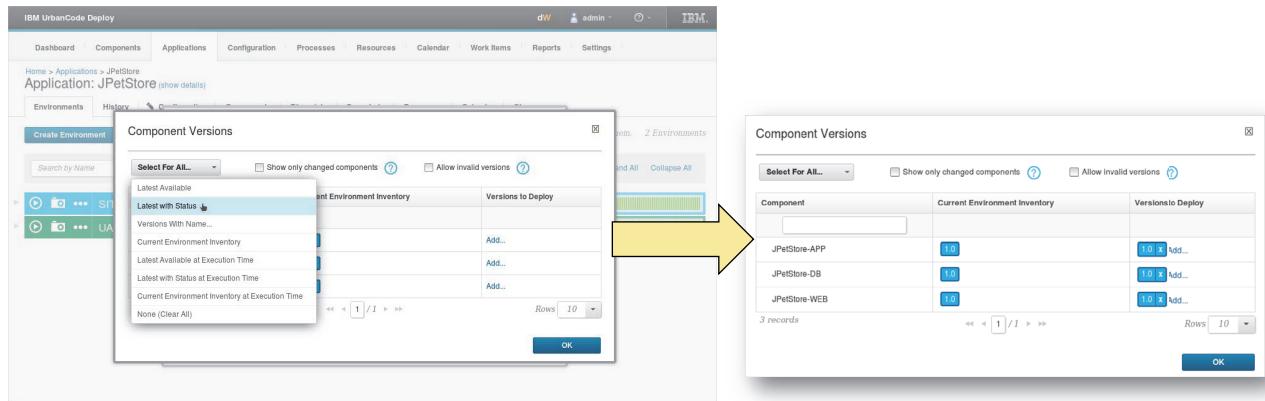
Status	Description	Created	By	Actions
Ready for Testing	Version is ready to be tested in UAT	2/1/2017, 2:34 PM		Remove

A callout box with the text 'Shows when a status was added' has an arrow pointing to the 'Created' column of the first table row. Below the statuses, the 'Artifacts' section lists two files: 'lib' (783.9 KB) and 'upgrade\_sql\_1.xml' (15.7 KB). There are buttons for 'Download All', 'Expand All', and 'Collapse All'.

### Version statuses are tracked in each component

In order to view a version status, navigate to the component containing the version and then to the component's **Versions** tab. Click the version. You see a table containing the version's statuses, and you can add or remove statuses from this interface.

## Select the latest components with a status



### Select the latest components with a status

When you select components for deployment, select “latest with status” to view the components that have passed the gate.

## Unit summary

- The security system for the server consists of an authentication realm, authorization realm, roles, and teams
- Permissions define what can be done, not who can do it; roles define who can do it
- Users are granted permissions by being assigned to teams
- Notifications are emails that are sent whenever user-defined trigger events on the server occur
- Approvals provide more visibility into deployments for audit trails
- An environment gate is a requirement that must be met before component versions can be deployed to an environment
- Version statuses identify when component versions meet criteria

## Exercises: Setting up server security



- Create a new user and role
- Create a team
- Associate the team to the existing application and environments



IBM Training



© Copyright IBM Corporation 2016. All Rights Reserved.