IBM Software

IBM UrbanCode Deploy Lab Workbook

Machine Setup and Lab Exercises

IBM

Contents

1 Overview

The overall objective of this Lab Workbook is to introduce you to the basic concepts and tool usage of IBM UrbanCode Deploy.

This version of the workbook is based on version 6.1.1.4 of IBM UrbanCode Deploy. This lab book has been written for a machine running Microsoft Windows 7.

The modules in this workbook will guide you through the steps in understanding how to install and use IBM UrbanCode Deploy. In the exercises, you will be working with a sample application called JPetStore. It consists of three components: web content, database and JEE application. You will gain an understanding of how to install the IBM UrbanCode Deploy tool, configure and deploy the *JPetStore* application to two sample environments, Systems Integration Test (SIT) and User Acceptance Test (UAT). There are six lab modules in this workbook and each are described below:

Machine Setup: This lab walks the reader through installing all the software necessary to support UrbanCode Deploy and the JPetStore web application.

Install Urbancode Deploy Server and Agents: This lab's focus is the role of the Deploy Administrator. In this lab you will install the IBM UrbanCode Deploy server and agents, verify the communications between the server and agents, and finally create the sample application's (*JPetStore*) databases.

Creating Urbancode Components, Applications and Environments: In this lab you will use UrbanCode Deploy to create representation model for the *JPetStore* application, a sample enterprise Java application that you will deploy in following modules. In this module you import the components of the *JPetStore* application into the UrbanCode Deploy server. You will then define an Application and Components for *JPetStore*. Finally, you will define a set of named Environments for the application.

Defining Computer Resources and Groups: In this lab you will build the representation model for the computing topology in our example Data Center.

In the remaining exercises you will create processes on the Components and Application, and test a partial deployment against one of your environments. You'll then move on to deploy additional versions of the application and create a snapshot. You will then apply that snapshot by promoting the snapshot from the sample SIT environment to the UAT environment.

2 Machine Setup

When you have completed this chapter, you will have a completed machine that is ready for the IBM UrbanCode Deploy server and client software installation.

2.1 Windows 7 machine requirements

The following settings and packages were used in the prepared virtual machine environment for this workbook. This is the general minimum configuration which is assumed has been prepared before continuing with the setup of the IBM UrbanCode Deploy environment contained in this workbook.

Resource	Notes
MS Windows 7 64-bit	Other versions should work but this version was used for all screenshots and specific commands contained in the included instructions
MS Windows .NET Framework	Some elements in this demo (e.g, MySQL) require the MS .NET framework be installed.
VMWare Tools	If you are using a virtual machine and are using VMWare Workstation to manage your virtual machine, install the latest version of VMWare tools – see the VMWare documentation for instructions. This will enable features like higher resolution graphics, and the ability to move files to/from the virtual machine by using copy/paste.
Mozilla Firefox 24 or Greater OR another supported browser: • Microsoft Internet Explorer version 9 or later • Apple Safari 8 or later • Google Chrome 39 or later	This lab is written for use with Firefox. If you use another browser, some of the graphics or instructions may vary slightly. See this page for all current system requirements, including supported browsers: http://www-03.ibm.com/software/products/en/ucdep
2 CPUs	Can be reduced at runtime if you don't have enough physical resources
8 GB RAM	Can be reduced at runtime if you don't have enough physical resources
30 GB disk storage	Can be reduced during initial image creation depending upon usage scenario. Creating the disk image as a single file can help improve performance. If this same image will be used to also install IBM UrbanCode Release it is recommended to start with 15 GB disk storage unless a separate USB storage device is used to hold the installation media.

2.2 Download the required software

The following software distributions are required and must be downloaded separately.

TIP – If you download them from a browser in your virtual machine, you can save the steps (and time) needed to copy them into the virtual machine from your host.

Package	Release	Download Location
MySQL Community Edition	5.6x or later	http://dev.mysql.com/downloads/mysql/
Java Developer Kit (JDK) and Runtime Environment (JRE) ***Note – the full JDK is required to install and run UrbanCode Deploy. Just a JRE is not sufficient.* *32-bit JDK and JRE are sufficient.*	7.0x or later	http://www.oracle.com/technetwork/java/javase/downloads/index.html System Environment Variables must be set after installation…. **JAVA_HOME** - (set to location of the JDK base installation directory… e.g. "**C:\Program Files (x86)\Java\jdk1.7.0_67**") **JRE_HOME** – (set to the location of the JRE base installation directory… e.g. "**C:\Program Files (x86)\Java\jre7**")
Mozilla Firefox	24 or later	https://www.mozilla.org/firefox
Apache Tomcat	7.0x or later	http://Tomcat.apache.org Download zip file for tomcat from the "Core" section for the operating system you are using. This workbook is written for Windows 7.
IBM UrbanCode Deploy Installation Package	6.1.x	https://developer.ibm.com/urbancode/products/urbancode-deploy/ For purposes of consistency, rename the file for this package to **IBM_UCD_Server_Install_Package.zip**

Package	Release	Download Location
IBM UrbanCode Deploy Plug-Ins	various	https://developer.ibm.com/urbancode/plugins/ Locate and download the following plugins using the search facility in the web page: [Mandatory – these plugins are required] • **DBUpgrader-x.xxxx.zip** • **Tomcat-x.xxxx.zip** [Optional – download these plugins to make a more robust-looking demonstration] • **IIS-AdminScripts-x.xxxx.zip** • **IIS-AppCmd-x.xxxx.zip** • **IIS-MSDeploy-x.xxxx.zip** • **MCWASPlugin.zip** • **WebSphere Liberty-x.xxxx.zip** • **WebSphereMessageBroker.zip** • **WebSphereMQ-x.xxxx.zip**
Artifacts.zip (contains the sample JPetStore Application)	1.0	Release and Deploy Learning Circle on developerWorks: https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=860ff390-6cab-4f95-ab37-66d2ca7521b4#fullpageWidgetId=Wfd5260395f07_47f9_aa1a_6eec1f65a7e0&file=2f4ca265-480e-4efd-ae63-57f815c87b95

2.3 Add Entries to the Hosts File

In order to demonstrate UrbanCode capabilities to deploy to multiple machines and environments with different topologies, we will simulate multiple machines by creating multiple server name entries in your machine's **hosts** file.

The hosts file on Windows machines is located in:

```
C:\Windows\System32\drivers\etc
```

You will need administrator privilege to edit this file on Windows. Add the following lines to the end of your hosts file:

```
127.0.0.1 ucd.deploy-server.com
127.0.0.1 ucd.uat-db-server.com
127.0.0.1 ucd.uat-app-server.com
127.0.0.1 ucd.sit-server.com
```

2.4 Install Mozilla Firefox

This lab is written using the Mozilla Firefox browser as the main interface. Microsoft Internet Explorer, Google Chrome and Apple Safari are also supported… but the pictures in this document reflect the use of Mozilla Firefox. For full details on supported browsers, see this page (click the tab named **Prerequisites**): << UrbanCode Deploy System Requirements >>

Download and run the install program for Firefox from the URL identified in the earlier section named "Download the Required Software". Follow the prompts, but do not to install any extra items that are sometimes offered with open software (like 3rd party search engines, tool bars, adware, etc.)

2.5 Install the Java JDK

UrbanCode deploy requires the full Java Developers Kit (JDK), not just the Java Runtime Environment (JRE).

We recommend installing the 32-bit version of the JDK, even if you are using a 64-bit operating system.

As of the writing of this book, Firefox only distributes a 32-bit version, so a 64-bit JRE won't work with it. Microsoft Internet Explorer runs in 32-bit mode by default, so you must change some options in order to make it run in 64-bit mode.

For these reasons we suggest using a 32-bit JDK and JRE.

Download and run the installation program for the Java Developers Kit from the URL identified in the earlier section named "Download the Required Software". Follow the prompts, but do not to install any extra items that are sometimes offered with open software (like 3rd party search engines, tool bars, adware, etc.)

After the installation is complete you must set two system environment variables to the base directory where you installed Java. The **examples** below assume you installed JDK 1.7.0.67 in the default location. You need to alter the values to match the actual location of the JDK on your system:

```
JAVA_HOME = "C:\Program Files (x86)\Java\jdk1.7.0_67"
```

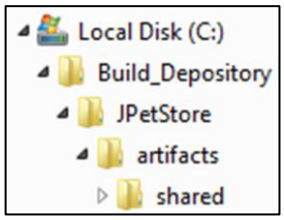
```
JRE_HOME ="C:\Program Files (x86)\Java\jre7"
```

2.6 Prepare the JPetStore Artifacts

Create a new folder on the **C:** drive - -

```
C:\Build_Depository\JPetStore
```

Extract the contents of the artifacts.zip file into this new folder. The new folder structure should look like this:



This will represent a place where the development build system deposits compiled artifacts (war files, jar files, etc.) for us to use. UrbanCode Deploy will pull from this location and import new versions into the CodeStation technology (a code repository that is included with UrbanCode Deploy).

2.7 Install MySQL Community Edition

2.7.1 Overview

This lab uses the JPetStore JEE reference application as the subject of the demonstration. This particular build of the JPetStore application is configured to use MySQL. UrbanCode Deploy will support, but does not require that you use MySQL. For simplicity sake, we download and install MySQL for the JPetStore application. (UrbanCode Deploy, itself, will use Derby as its data store for this lab.)

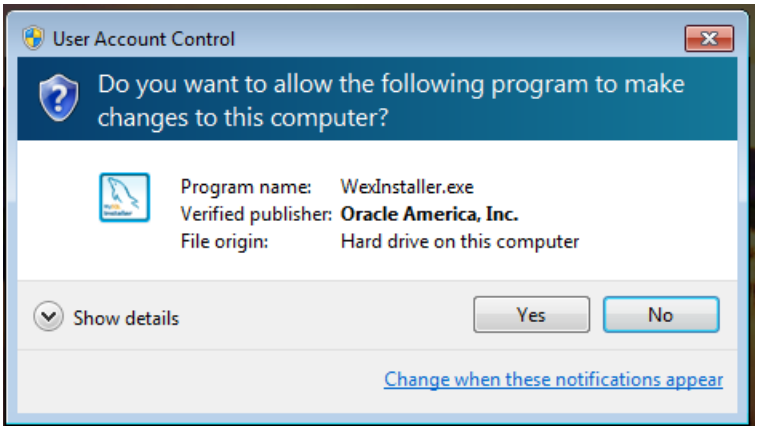
By default, MySQL will use port 3306 for communication. Make sure any firewalls will allow communication on this port.

**NOTE - - MySQL requires the Microsoft .NET Framework. Please install the latest version of the .NET Framework before continuing.

2.7.2 Run the MySQL Setup Program

Download and run the install program for MySQL from the URL identified in the earlier section named “Download the Required Software”.

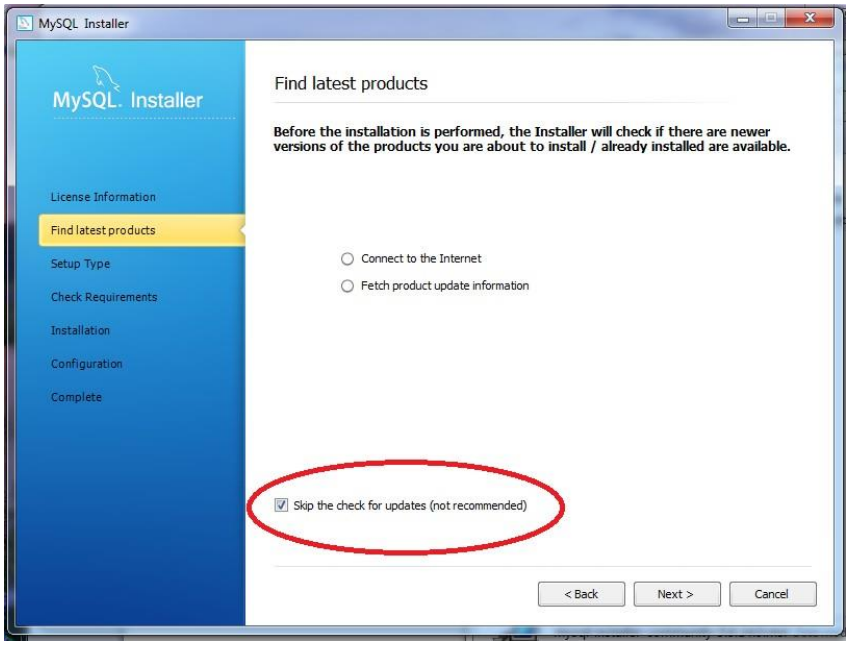
Answer “Yes” to any User Account Control dialogs asking if you want to allow the program to make changes on the computer.



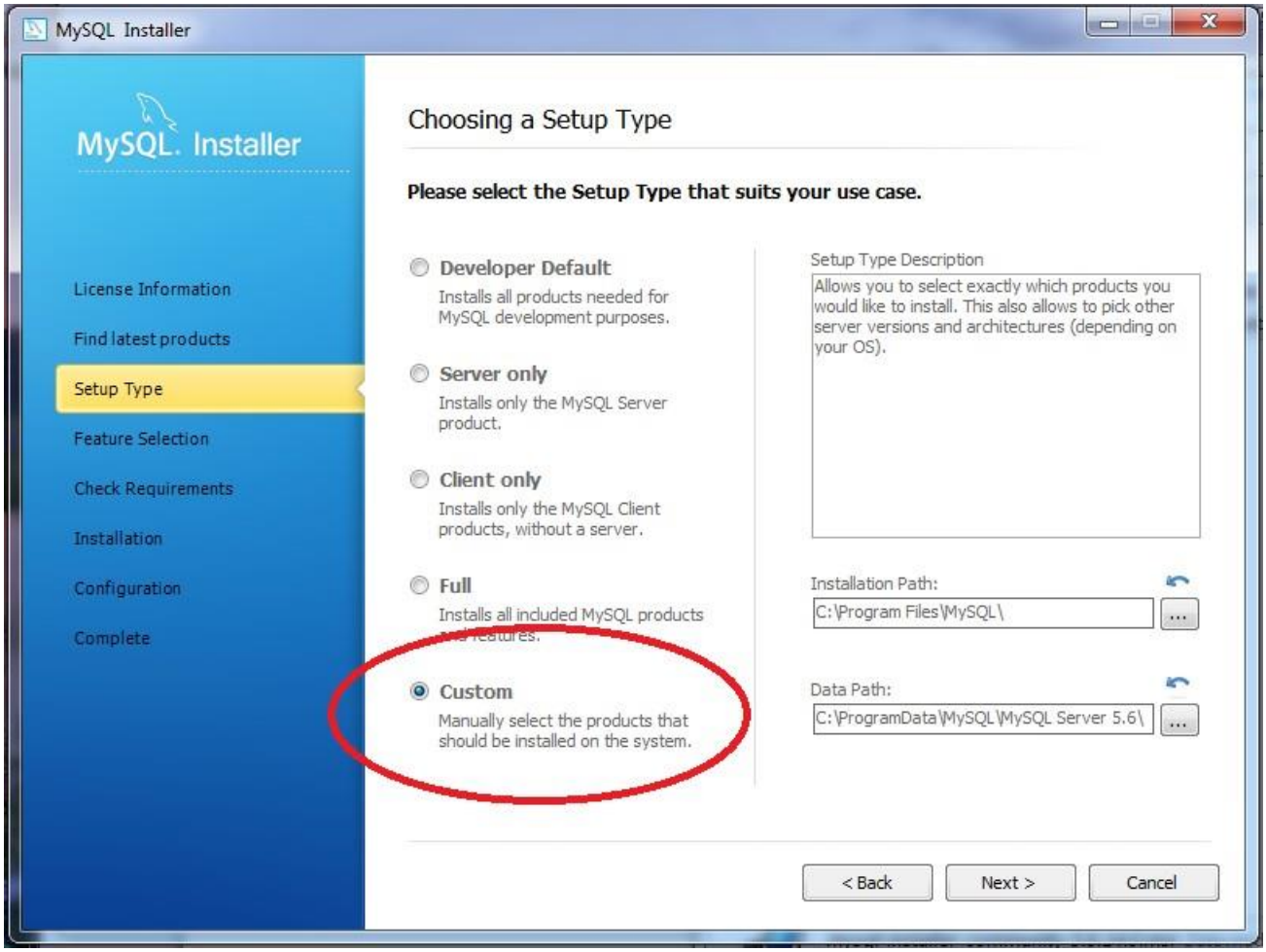
On the initial screen select **Install MySQL Products**.

On the **License Agreement** screen, check the box to accept the license terms and click the **Next** button.

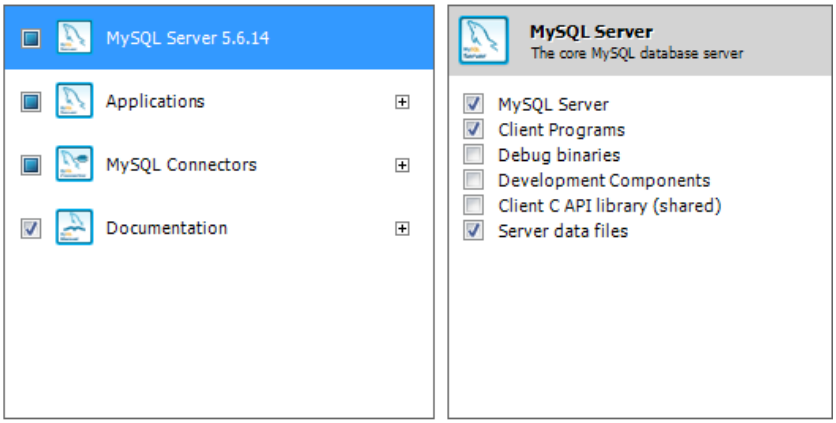
On the **Find Latest Products** screen, check the box to **skip the update check** and click the **Next** button. (NOTE – this is simply for expediency. If you do want to check for updates, no errors should be introduced, but some of your subsequent steps in the MySQL installation process may differ).

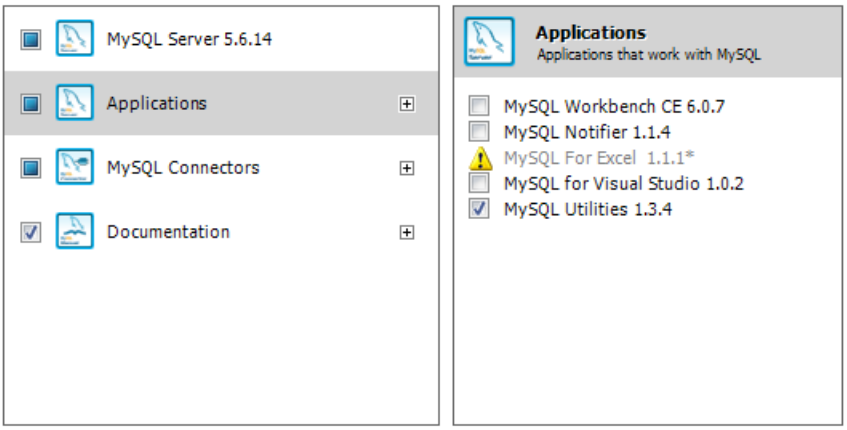


On the **Choosing a Setup Type** screen, select the **Custom** radio button - accept the defaults for the Installation and Data paths and click the **Next** button.



In the following screens, note the category selected on the left, and the options selected on the right:

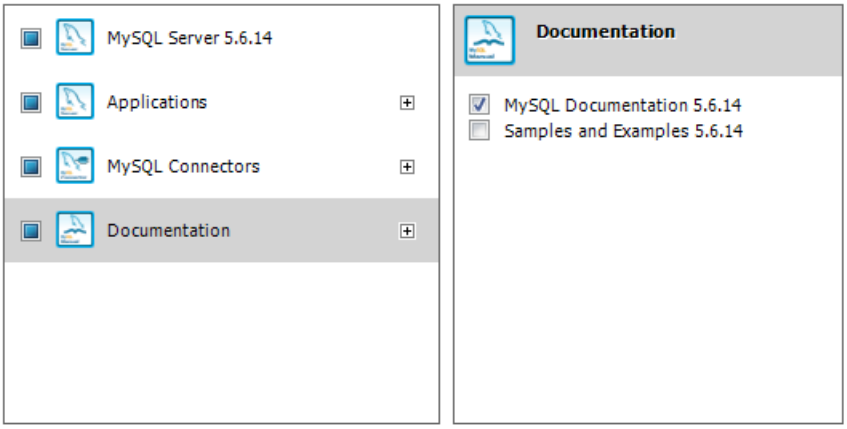




** NOTE – if Microsoft Excel is not installed in your machine, you will see the warning icon as depicted above. Excel components are not needed for this lab.

Accept the default selections for **MySQL Connectors**.

Under the **Documentation** option, **do not** install the Samples.

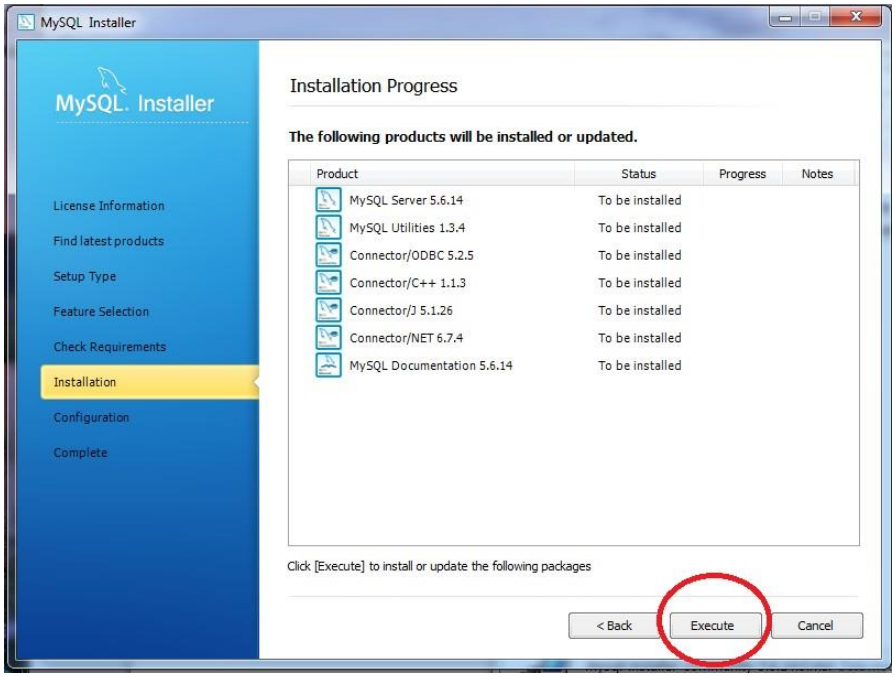


Click the **Next** Button.

On the **Check Requirements** screen, click the **Next** button.

NOTE – the MS .NET Framework must be installed before the MySQL installation will continue. If the installation program allows you to click the **Next button, then the requirement has been met.

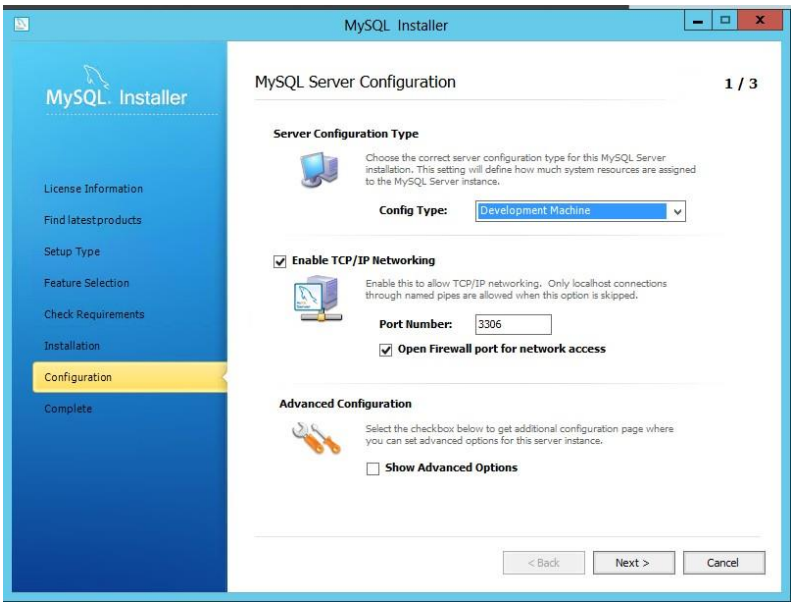
On the **Installation Progress** screen, click the **Execute** button.



This will take a few minutes to complete. After the installation completes, click the **Next** button to begin the configuration process.

On the **Configuration Overview** screen, click the **Next** button.
Several screens follow entitled **MySQL Server Configuration** – on the first, accept the defaults…

Server Configuration Type = **Development Machine**
Enable TCP/IP Networking = **Checked**
Port = **3306**
Open Firewall port for network access = **Checked**



Click the **Next** button.

On the next screen, set the **MySQL Root Password** = **password** (all lower case). There is no need to create any additional users at this time.

Click the **Next** button.

On the **Windows Service Details** screen, keep the defaults (allow to install as a service with the standard system account) and click the **Next** button.

On the **Configuration Overview** screen, click the **Next** button and then the **Finish** button to complete the installation.

2.7.3 Create the JPetStore Application Database

The JPetStore application requires a database. Create the empty database and verify that the database user can access it.

From the **Windows Start** menu, select

All Programs → MySQL → MySQL Server 5.6 → MySQL 5.6 Command Line Client

Enter the password (which is **password**, all lower case).

The system will respond with a **mysql>** prompt.

Type this command to create the database for the System Integration Test (SIT) environment:

```
create database jpetstore_sit;
```

You should receive the response –

```
Query OK, 1 row affected
```

Type this command to create the database for the User Acceptance Test (UAT) environment:

```
create database jpetstore_uat;
```

You should again receive the response –

```
Query OK, 1 row affected
```

Type this command to create the primary user:

```
create user 'jpetstore'@'localhost' identified by 'jppwd';
```

You should receive the response –

```
Query OK, 0 rows affected
```

Type these commands to grant privileges to the user for the two new databases:

```
grant all privileges on jpetstore_sit.* to 'jpetstore'@'localhost';
grant all privileges on jpetstore_uat.* to 'jpetstore'@'localhost';
```

After each command you will receive a response –

```
Query OK, 0 rows affected
```

Type `quit` to exit the MySQL program and close the command window.

2.7.4 Verify Database Access for the jpetstore User

	*In order for the following commands to work, you must add the location of the file mysql.exe to the PATH system environment variable. By default that location is:* `c:\Program Files\MySQL\MySQL Server 5.6\bin\` *(Note: – the reference to "5.6" may change dependent on the version of MySQL you have installed.)*

Open a command window and start the **MySQL** command interpreter as the **jpetstore** user by typing:

```
mysql -u jpetstore -pjppwd
```

Type the following command to test access:

```
show databases;
```

If the user has access, you should see the names of both the databases you just created (`jpetstore_sit` and `jpetstore_uat`).

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| jpetstore_sit      |
| jpetstore_uat      |
| test               |
+--------------------+
4 rows in set (0.11 sec)
```

Type `quit` to exit the **MySQL** command interpreter. Close the command window.

2.8 Setup Tomcat Servers

2.8.1 Overview

In order to provide an environment that shows off the capabilities of UrbanCode deploy, we will create TWO separate instances of the Tomcat server – one to represent System Integration Testing (SIT), and one representing User

Acceptance Testing (UAT). You could use this same technique to create even more Tomcat instances to exemplify other environments (e.g., Production), but this lab stops after only these two.

HTTP traffic will be directed between the two environments by setting up each server to listen on a different port – so browser requests must be modified to include the particular port as well as the host name.

Also, UrbanCode Deploy itself uses an embedded Tomcat instance that we will configure to listen on port 8080. The UrbanCode Deploy installation script takes care of installing this instance of Tomcat.

The following table enumerates all ports that need to be available for the processes involved in this demo (make sure they are not blocked by any firewalls in your machine).

Server:	Listens on Port(s):
MySQL	3306
UrbanCode Deploy Server	8080 8443 7918 (for JMS)
SIT Tomcat Instance	8005 8085 8445
UAT Tomcat Instance	8006 8086 8446

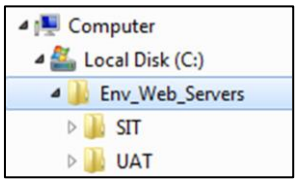
2.8.2 Create Locations to Hold Tomcat Servers

Create a folder on the C drive named **`Env_Web_Servers`**

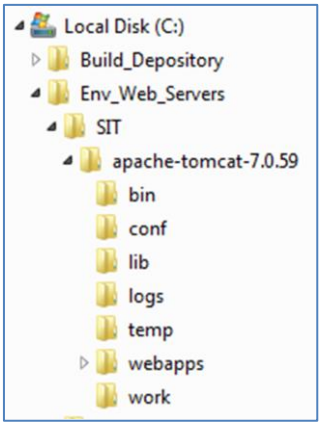
Create two sibling folders in the **`Env_Web_Servers`** folder you just made:

- **`SIT`**
- **`UAT`**

Your new folders should look like this:



Extract the zip file with Tomcat into the new **`SIT`** directory. Your **`SIT`** folder should now look like this:



2.8.3 Add user and role entries to the Tomcat users file in the SIT location

Navigate to the following directory, which contains the Tomcat SIT configuration files:

```
c:\Env_Web_Servers\SIT\apache-tomcat-xxxx\conf
```

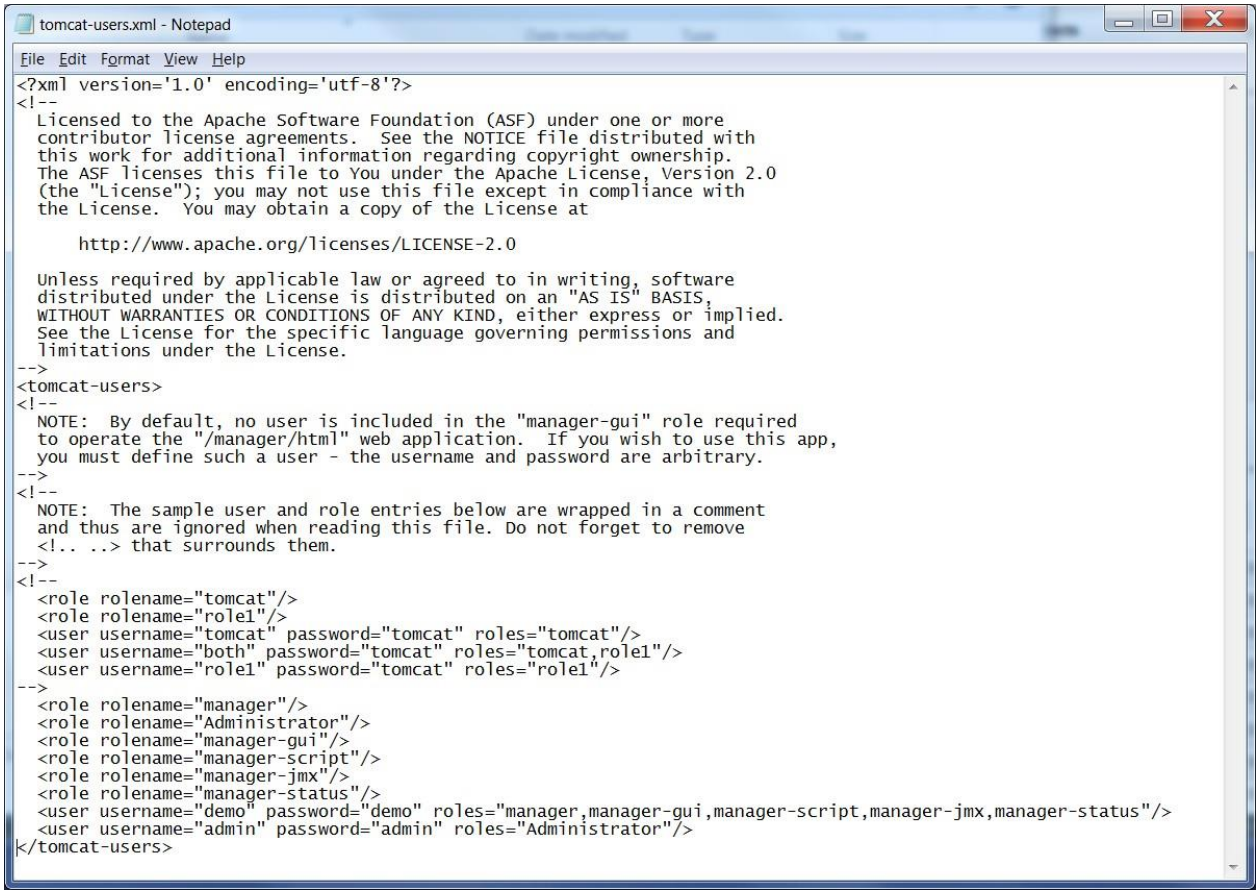
	*The reference to "xxxx" in the path above represents the version of Tomcat that you installed. Substitute the correct reference for the version number you actually downloaded to make the path correct.*

Edit the file named `tomcat-users.xml`

Add the following lines AFTER the last comment block, but BEFORE `</tomcat-users>` tag. (Comments are demarked by a block of text enclosed by `<!--` at the beginning, and `-->` at the end.)

```
<role rolename="manager"/>
<role rolename="Administrator"/>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="demo" password="demo" roles="manager,manager-gui,manager-
script,manager-jmx,manager-status"/>
<user username="admin" password="admin" roles="Administrator"/>
```

Your file should now look like this:



Save the file and close the editor.

2.8.4 **Configure SIT Tomcat Ports**

Edit the file named **`server.xml`** found in the same directory where we just found **`tomcat-users.xml`** (**`C:\Env_Web_Servers\SIT\apache-tomcat-xxxx\conf\`**)

Perform a search and replace of all occurrences based on the following:

- Replace **`8080`** with **`8085`**
- Replace **`8443`** with **`8445`**

Save the file and close the editor.

2.8.5 **Verify Access to the SIT Tomcat Server**

	*The system environment variables JAVA_HOME and JRE_HOME must be set to the location of your JDK and JRE, respectively. Otherwise you can start the Tomcat server. See the prerequisites at the beginning of this document.*

Navigate to the folder:

`C:\Env_Web_Servers\SIT\apache-tomcat-xxxx\bin`

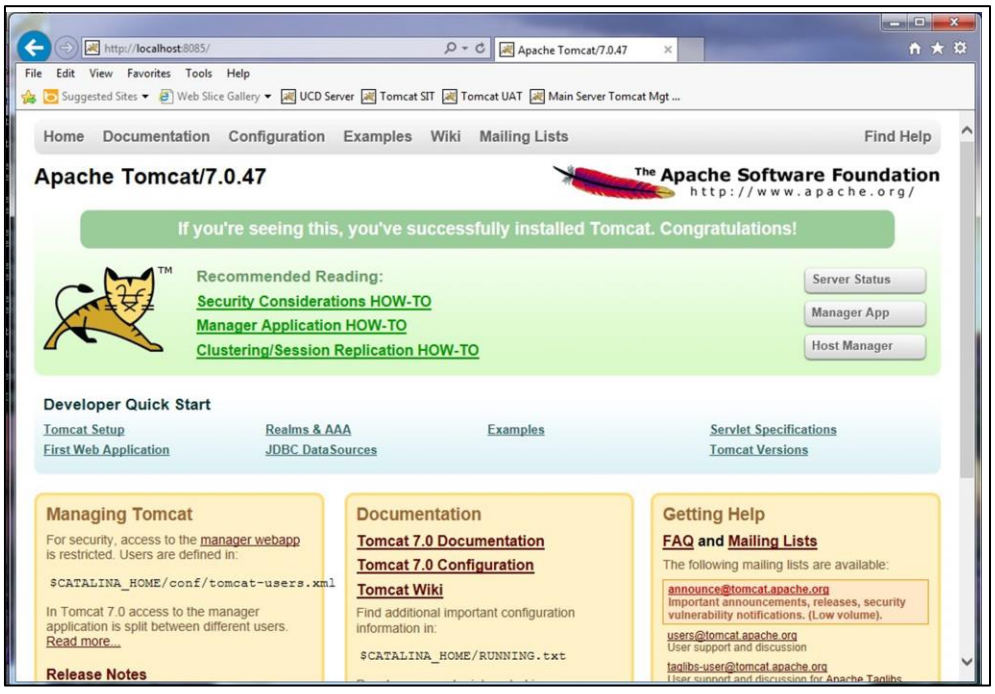
Double-click the file `startup.bat`.

A command window will appear and display status. In a few seconds you should see text that says "`INFO: Server startup in` " followed by the startup time in milliseconds. This indicates the Tomcat server has started without errors.

	*The command window must remain running in order to work with the Tomcat instance. Minimize the windows, but do not close them. (If you do close them, simply rerun the* `startup.bat` *file again.)*

Open a browser and navigate to the URL: `http://ucd.sit-server.com:8085`

This should display the Tomcat web console.



2.8.6 **Configure Tomcat for UAT Environment**

In the previous section you updated two files in the SIT configuration:

- `tomcat-users.xml`
- `server.xml`

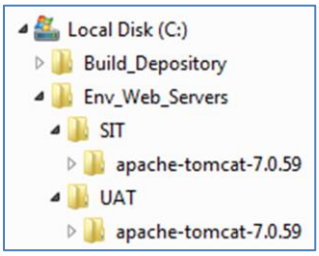
These files reside in this location:

```
c:\Env_Web_Servers\SIT\apache-tomcat-xxxx\conf
```

Now we want to create a companion web instance for the UAT configuration.

Copy the entire apache-tomcat directory from the **SIT** folder and paste a duplicate in the **UAT** directory

Your **SIT** and **UAT** directories should each now have an apache-tomcat folder below them.



Now navigate to the following folder in the **UAT** configuration:

```
c:\Env_Web_Servers\UAT\apache-tomcat-xxxx\conf
```

Edit the file named **server.xml**. Perform a search and replace of all occurrences of the values indicated below:

- Replace **8005** with **8006**
- Replace **8085** with **8086**
- Replace **8445** with **8446**

Save the file and close the editor.

2.8.7 Verify Access to the UAT Tomcat Server

	*The system environment variables **JAVA_HOME** and **JRE_HOME** must be set to the location of your JDK and JRE, respectively, before you can start the Tomcat server. See the prerequisites at the beginning of this document.*

Navigate to the folder:

```
C:\Env_Web_Servers\UAT\apache-tomcat-xxxx\bin
```

Double-click the file **startup.bat**.

A command window will appear and display status. In a few seconds you should see text that says "**INFO: Server startup in**" followed by the startup time in milliseconds. This indicates the Tomcat server has started without errors.

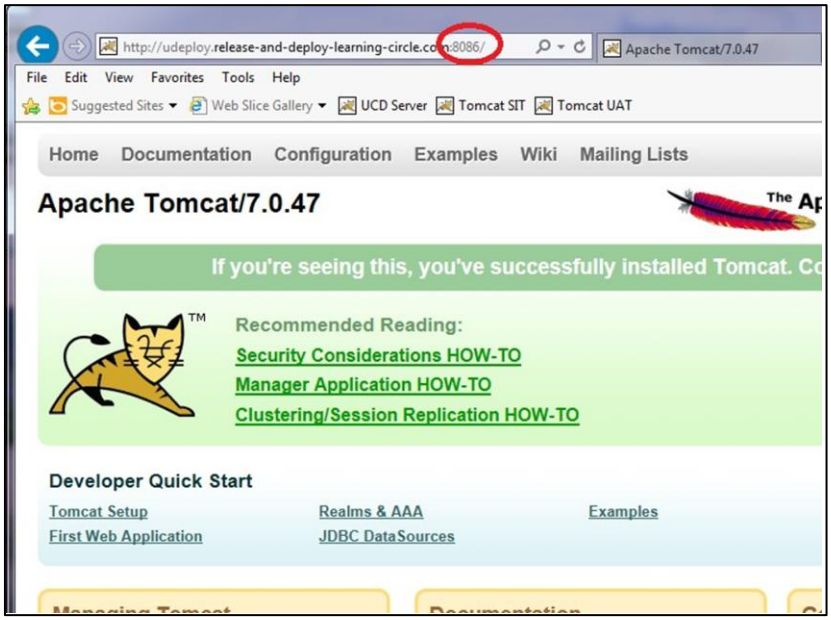
	*IMPORTANT* *The command window must remain running in order to work with the Tomcat instance. Minimize the window, but do not close it. (If you do close it, simply rerun the* **`startup.bat`** *file again.)*

Open a browser and navigate to the URL:

`http://ucd.uat-app-server.com:8086`

This should display the Tomcat web console.

Verify you can get to the default Tomcat web console, and ensure the correct port is being used to distinguish this **`UAT`** environment:



2.9 Summary

You have now set up the environment necessary to support the rest of the labs in this workbook. You have:

- Downloaded all required software and artifacts
- Modified your hosts file to support a demo that will look like multiple servers
- Installed Mozilla Firefox
- Installed a Java Developers Kit
- Extracted the components of JPetStore to a location that represents the location where developers deposit built artifacts
- Installed and configured MySQL to prepare for use as the JPetStore database.
- Set up 2 separate instances of Apache Tomcat to represent servers in 2 different environments (System Integration Test, and User Acceptance Test)

Your machine is now ready to begin installing UrbanCode Deploy and run the JPetStore sample application.

3 Install UrbanCode Deploy Server and Agents

In this lab we will install and configure the IBM UrbanCode Deploy server and agents. When completed we will have a single system running with the main UrbanCode Deploy server, agent, SIT and UAT environments.

3.1 Install IBM UrbanCode Deploy Server

Create the following directory on the hard drive:

```
C:\i
```

	*The UrbanCode Deploy setup utility runs from a batch file, not a Windows installer. We will unzip all the files from the downloaded zip archive into this directory and run the setup routine from there.* *Because we will be working in a command window, it is expedient to use a directory right on the root with a very short name.*

Locate the file you downloaded for UrbanCode Deploy – it should have come in a single zip file, and you should have renamed it to `IBM_UCD_Server_Package.zip`. See the note in Section 2.2.

Right-click on the zip file and select Extract All…

In the resulting dialog (Extract Compressed (Zipped) Folders, locate the field named "Files will be extracted to this folder" …… set this value to `C:\i`

Click the "Extract" button and wait for the files to be processed. This will extract all files into a new sub- directory under `C:\i` named `ibm-ucd-install`.

Open the Windows 7 Start menu and type `cmd` in the **'Search Programs and Files'** area.

IMPORTANT!!! IN THE NEXT STEP YOU MUST RUN THE COMMAND PROMPT AS ADMINISTRATOR OR SOME PARTS OF THE INSTALLATION WILL NOT CONFIGURE CORRECTLY!!!

In the search results, locate **cmd.exe**… right-click on it and select **"Run as Administrator"**... confirm any dialogs that appear.

In the new command window, change directories to `C:\i\ibm-ucd-install\`

	*The system environment variables **JAVA_HOME** and **JRE_HOME** must be set to the location of your JDK and JRE, respectively, before you can install or run the server.* *See the prerequisites at the beginning of this document.*

Start the installation by executing the batch file

```
install-server.bat
```

Respond to the prompts as indicated below. (Note: It is ok if the installation program reports that it can't find tools.jar.)

Enter **F** to display the full license agreement.

Enter **Y** to accept the license agreement.

UrbanCode Deploy Server installation directory: **C:\IBM\ibm-ucd\server** (NOT default)

If it prompts that the specified directory doesn't exist, respond: **Yes (default)**

Home directory of JRE/JDK used to run the server: **C:\Program Files (x86)\Java\jdk1.7.0_67 (default)**
- NOTE: Your value here will be different… confirm it is set to your JRE or JDK (JRE is sufficient in this particular value.)

Host name to access the Web UI: **ucd.deploy-server.com** (NOT default)

Web UI to always use secure connections using SSL: **Yes (default)**

Port on which Web UI listens for secure HTTPS requests: **8443 (default)**

Port on which Web UI redirects unsecured HTTP requests: **8080 (default)**

Initial password for the admin user: **admin** (NOT default)
- *the system will ask you to confirm password in the next prompt*

Port for agent communication: **7918 (default)**

Require mutual authentication: **No (default)**

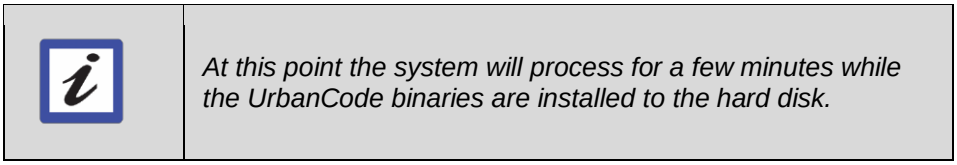
Port and hostname of Rational License Key Server: **<none> (default)**

Create database schema: **Yes (default)**

Database type to use: **derby (default)**

Database username: **ibm_ucd (default)**

Database password: **password (default)**



At this point the system will process for a few minutes while the UrbanCode binaries are installed to the hard disk.

Install Server as a Windows service? **Y** (NOT default)

Service name: **ibm-urbancode-deploy (default)**

User account to run the new UrbanCode service: **.\localsystem (default)**

Start UrbanCode service automatically? `Y` (NOT default)

Press the **Enter/Return** key when prompted to exit the installation script. Wait a few minutes to allow the server to start before continuing.

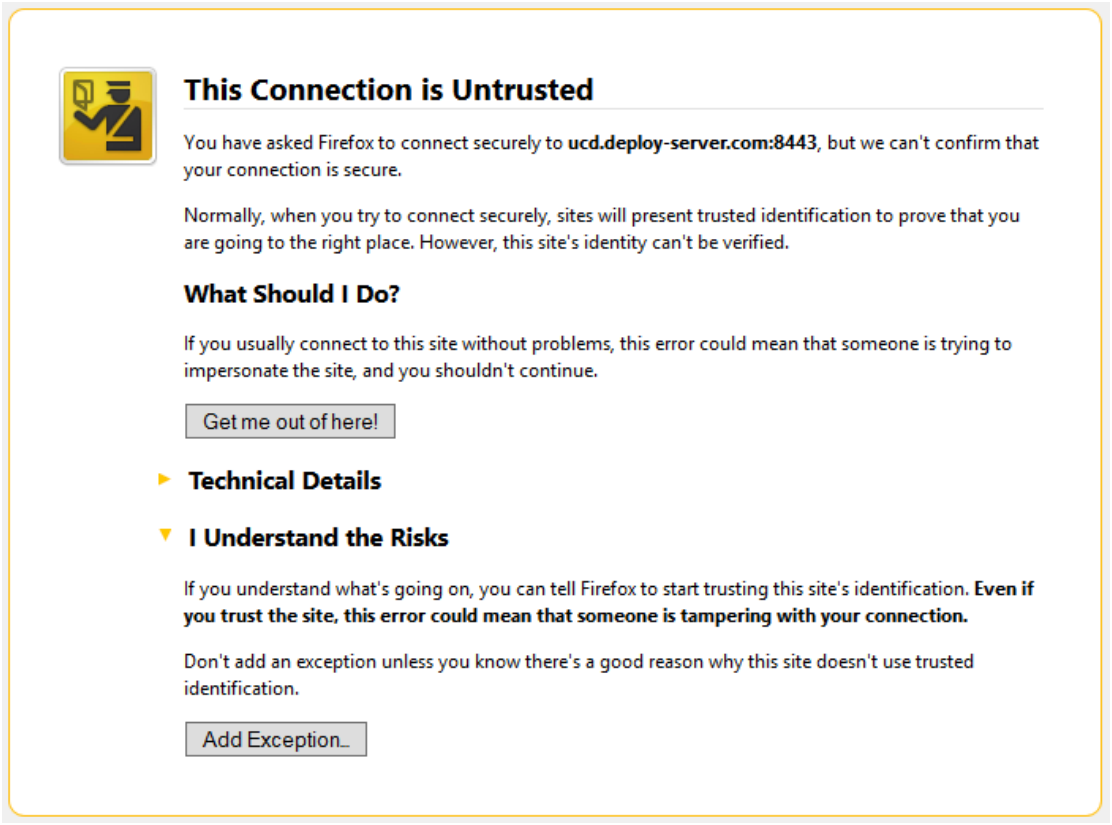
Open a browser window and enter the URL:

```
http://ucd.deploy-server.com:8080
```

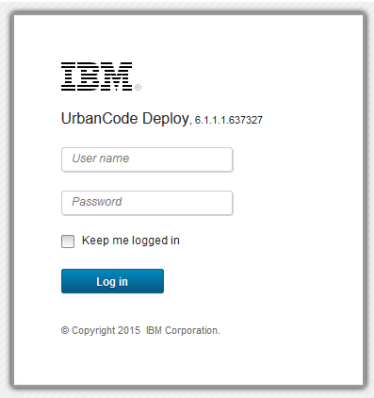
TIP – you may want to bookmark this URL for future reference.

If you are challenged with security warning about the connection, add an exception (this is just accessing the local server).

Below is the window Firefox shows:



If you see the login prompt for UrbanCode Deploy, then the server is running.

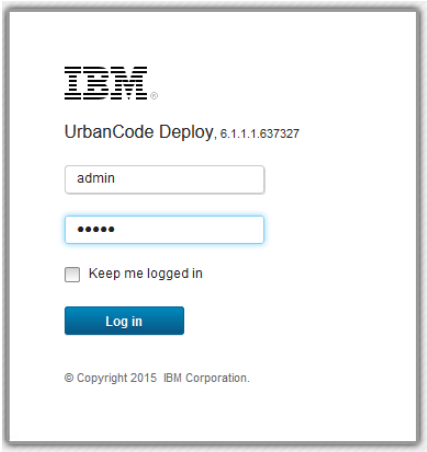


3.2 Install the IBM UrbanCode Deploy agent

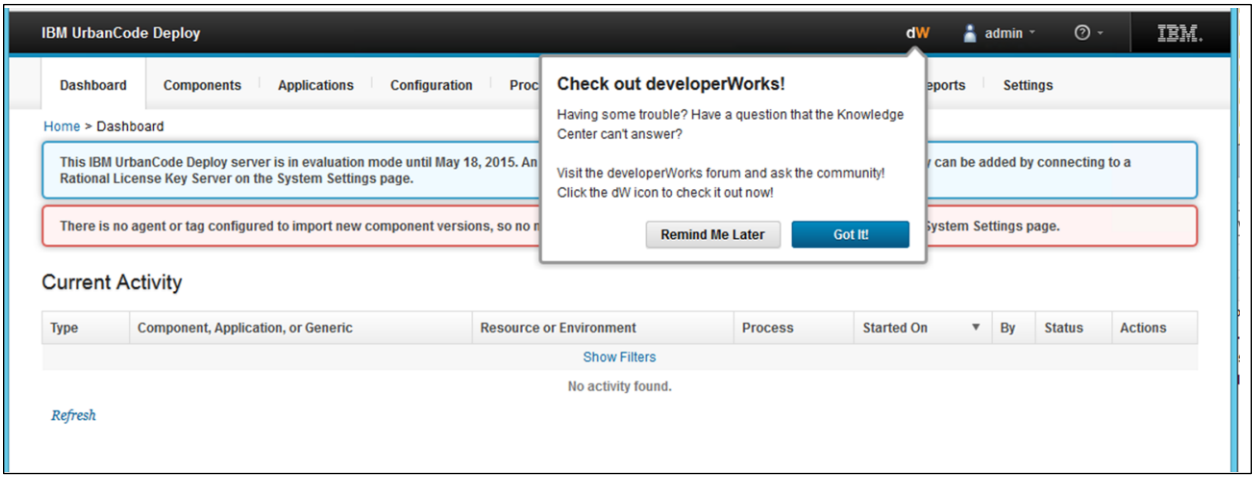
UrbanCode Deploy uses software "agents" to perform work on remote servers. For our lab, we will only use the one machine (the same one we've just installed the UrbanCode Deploy server onto), but in a real-world installation, this one UrbanCode server would manage the applications on hundreds or thousands of remote machines. We simulate that in our lab by making clever use of domain names to make it *appear* as if there are multiple machines involved.

The agent software is available via the installed UrbanCode server. Since we have installed the server, we can now access the installation bits for the agent.

Log in using **`admin`** for both the **user name** and **password**.



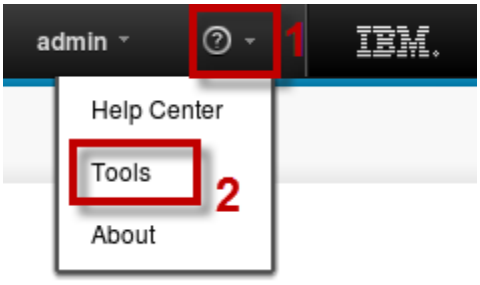
Dismiss the advertisement to "Check out developerWorks" (click "Got it").



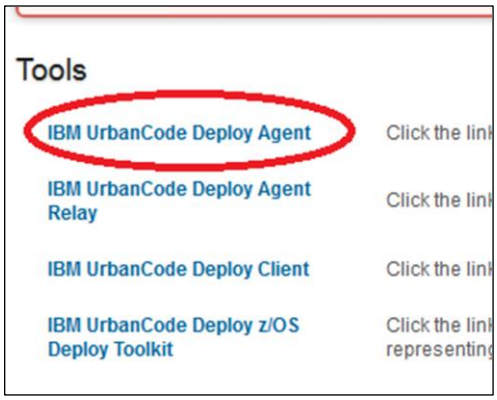
Ignore the warning banners in blue and red for now.

Click the IBM UrbanCode Deploy **Help** menu button (upper right of the browser window).

Click **Tools**



Click the link for **IBM UrbanCode Deploy Agent** to initiate a file download



*The browser by default will download the file into the current OS user's **Downloads** folder.*

Log out of UrbanCode Deploy and close the browser.

Navigate to the location where the agent was downloaded (see info box above).

Right-click on the zip file and select **Extract All…**

In the resulting dialog (Extract Compressed (Zipped) Folders, locate the field named "Files will be extracted to this folder" …… set this value to `C:\i`

Click the "Extract" button and wait for the files to be processed. This will extract all files into a new folder under `C:\i` named `ibm-ucd-agent-install`.

Open the Windows 7 Start menu and type `cmd` in the **'Search Programs and Files'** area.

IMPORTANT!!! IN THE NEXT STEP YOU MUST RUN THE COMMAND PROMPT AS ADMINISTRATOR OR SOME PARTS OF THE INSTALLATION WILL NOT CONFIGURE CORRECTLY!!!

In the search results, locate **cmd.exe**… right-click on it and select **"Run as Administrator"**... confirm any dialogs that appear.

In the new command window, change directories to `C:\i\bm-ucd-agent-install`

*The system environment variables **JAVA_HOME** and **JRE_HOME** must be set to the location of your JDK and JRE, respectively, before you can install or run the agent.*

See the prerequisites at the beginning of this document.

Start the installation by executing the batch file

`install-agent.bat`

Respond to the prompts as indicated below. (Note: It is ok if the installation program reports that it can't find tools.jar.)

Agent install directory: `C:\IBM\ibm-ucd\agent` (NOT default)

If it prompts you to create the installation directory: `Y (default)`

Home directory of JRE/JDK used to run the agent: `/usr/lib/jvm/ java-1.7.0-openjdk-1.7.0.25.x86_64/jre (default)`

- NOTE: Your value here will be different than the one shown above… confirm it is set to your JRE or JDK (JRE is sufficient in this particular value.)

Will the agent connect to an agent relay instead of directly to the server? `N (default)`

Hostname of UrbanCode Deploy server: `ucd.deploy-server.com` (NOT default)

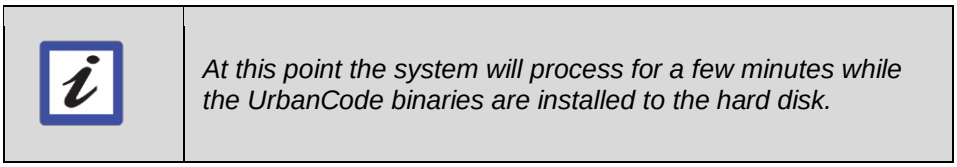
Enter the agent communication port for the server: **7918 (default)**

Configure a failover server connection? **N (default)**

Does the server agent communication use mutual authentication with SSL? **N (default)**

Enter the name for this agent: **localhost** (NOT default)

Enter teams for this agent: **None (default)**



Install Agent as a Windows service? **Y** (NOT default)

Service name: **ibm-urbancode-deploy-agent (default)**

User account to run the new UrbanCode service: **.\localsystem (default)**

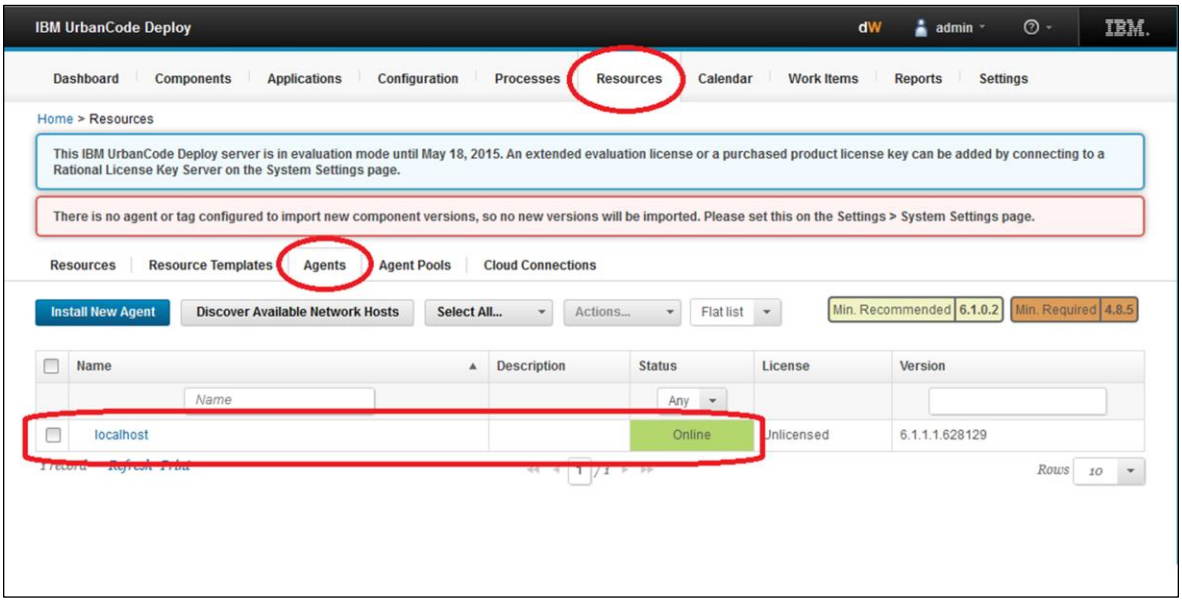
Start UrbanCode service automatically? **Y** (NOT default)

Press **Enter** when prompted to exit the installation script. Wait a few minutes to allow the agent to start before continuing.

3.3 Verify the IBM UrbanCode Deploy agent and server are communicating

Return to the UrbanCode Deploy browser window (or log in again if you have closed it - - username **admin** password **admin**.)

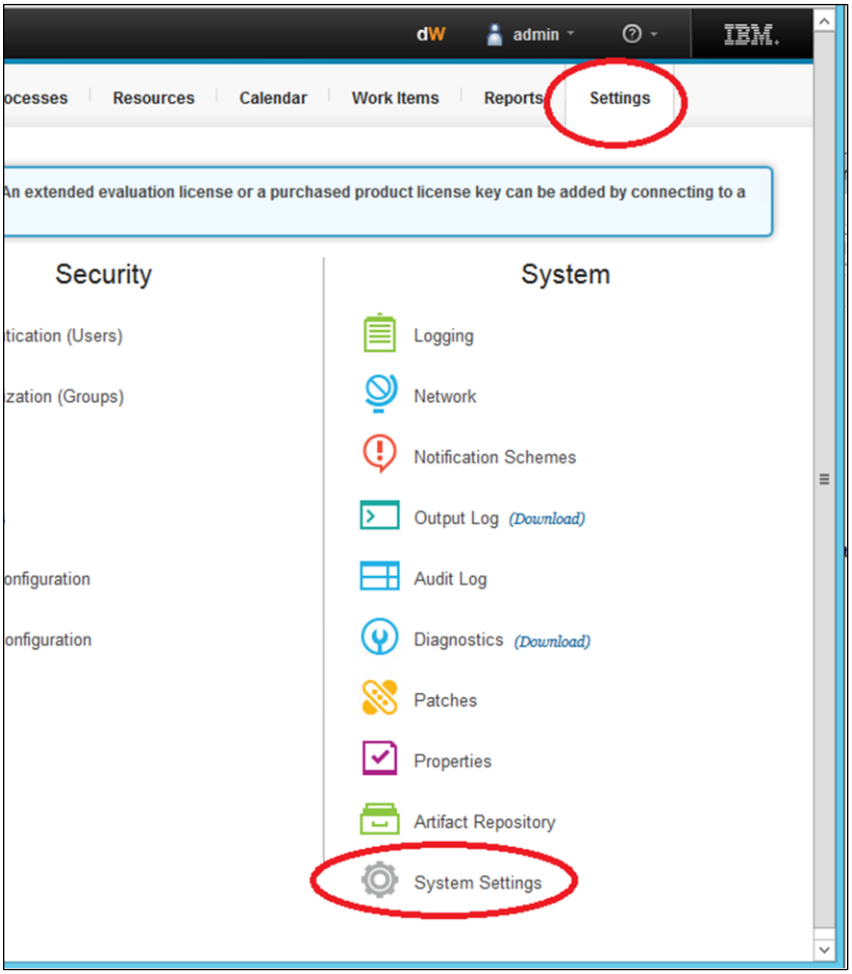
Click the Resources tab on the dashboard. On the Resources page click the **Agents** tab.
You should see a resource named **localhost**, and the status value should be **Online**.



3.4 Assign Agent to Import Component Versions

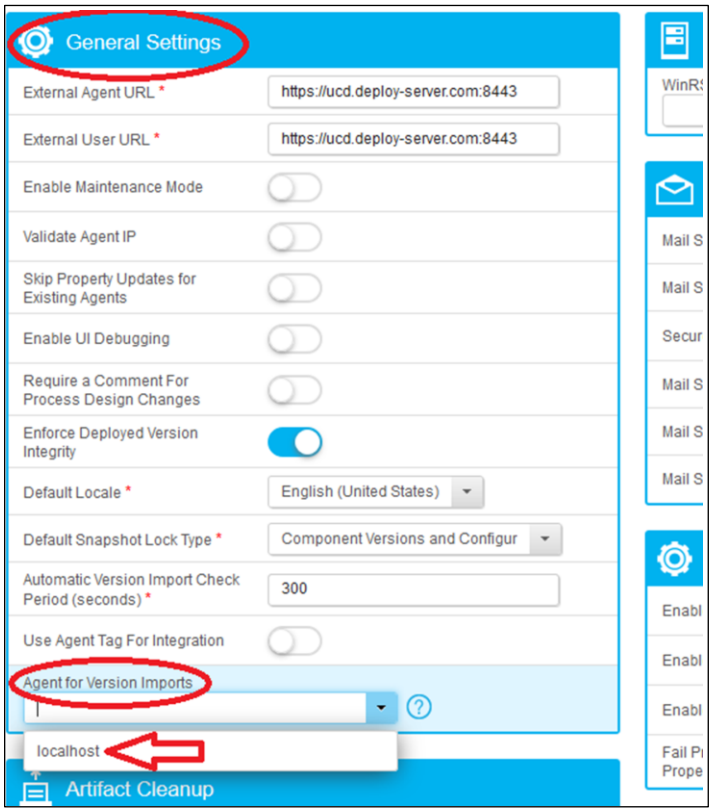
The banner in red warns that no agent has yet been assigned to work to import versions of artifacts that UrbanCode will deploy. We can rid ourselves of this banner by assigning this new agent.

In the top row of tabs, click **Settings**, and in the far right column, click **System Settings**.



Look in the **General Settings** section (left hand column) and scroll down until you see the field named **Agent for Version Imports**.

Set the value of that field to **localhost** using the drop-down box, then click the **Save** button. (You may have to scroll down further to see the button.)



This should dismiss the red banner from UrbanCode Deploy.

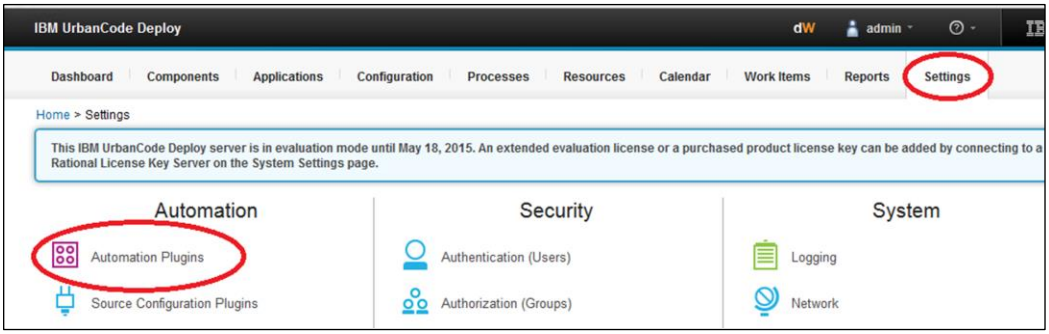
3.5 Install UrbanCode Deploy Automation Plugins

UrbanCode deploy implements a core set of functions that ship with the server, and provides an extensible framework built around “plugin” functionality. Plugins are self-contained extension units that you can apply to UrbanCode Deploy to provide specific integration with 3rd party technology.

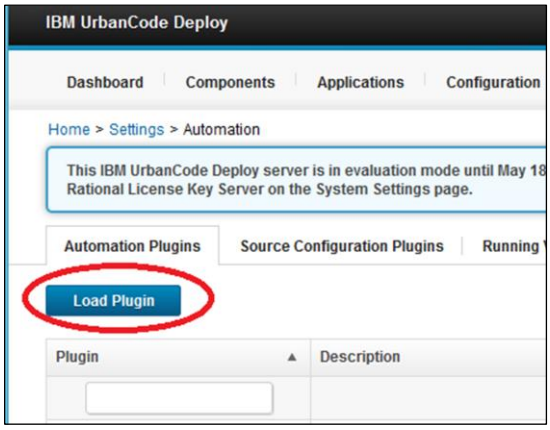
Only 2 plugins are required for this lab – DBUpgrader and Tomcat. The rest listed in Section 2.2 are optional, but make for a good presentation to illustrate the extended capabilities of UrbanCode deploy.

Make a new folder in `C:\i` named `plugins`. Copy the plugin zip files for **DBUpgrader** and **Tomcat** (and any others you downloaded) into this folder. (See Section 2.2 about downloading the required software.)

Return to the UrbanCode Deploy browser. In the top row of tabs, click **Settings**, and in the far left column, click **Automation Plugins**.

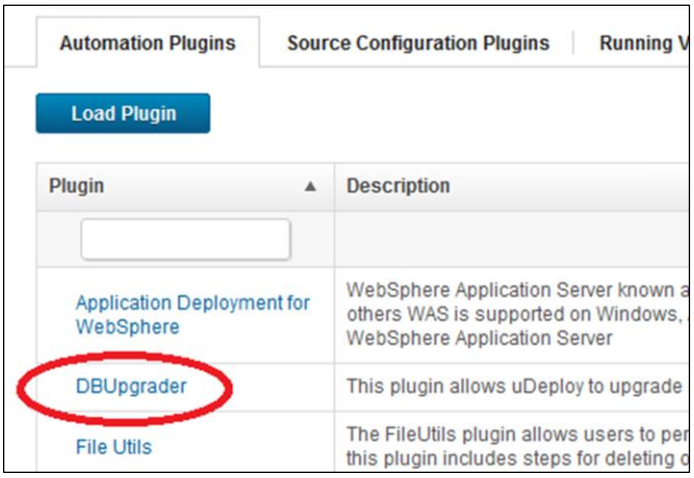


Click the **Load Plugin** button.



Click the **Browse…** button in the dialog and navigate to the `C:\i\plugins` folder – select the **DBUpgrader** plugin file, click the **Open** button, then the **Submit** button to load it.

You should then see `DBUpgrader` listed in the web page.



Repeat these steps for the **Tomcat** plugin, and any other plugins you wish to load into UrbanCode Deploy.

3.6 Remove unneeded files

Log out of UrbanCode Deploy, close down any running programs and delete the `C:\i` folder and all its contents from your system.

This will recover the disk space for all the temporary files we used so far.

3.7 Optional shutdown and backup

At this point in time, you can create a snapshot of the virtual machine. If you do so, shut down the virtual machines first.

3.8 Summary

The UrbanCode Deploy server and agent have been deployed. We've set up Tomcat with 2 instances to represent a System Integration Test (SIT) environment and a User Acceptance Test (UAT) environment (separated by different ports). Empty databases for the JPetStore application have also been created.

Note on License Keys – blue banner during evaluation period.

IBM UrbanCode Deploy installs with a temporary 60 day license. During this period a blue banner appears on every screen until the administrator installs and configures the Rational License Key Server for use with a valid key. The remaining lab illustrations have been rendered after UrbanCode Deploy has been properly licensed. Although your own system will be displaying this blue banner throughout your lab exercises, all other aspects and functions of UrbanCode Deploy should follow this document.

Installation and configuration of the Rational License Key Server is outside the scope of this document. For more information on this topic, please refer to this link:

https://www-01.ibm.com/software/rational/support/licensing/

4 Creating UrbanCode Components, Applications and Environments

4.1 Overview

JPetStore is a sample enterprise Java application that you will deploy in the following labs. UrbanCode Deploy allows an administrator to model their computing infrastructure and applications in terms of discrete elements. This exercise will introduce you to those elements as you encode them into the product. Those elements include:

- Application Projects
- Components
- Environments
- Processes

In this lab you will:

- Create the JPetStore application project
- Break down JPetStore into individual components and load each component
- Define how the components are assembled to create the deployable application
- Define two named environments to represent infrastructure used by two main groups in the software development lifecycle:

    1. System Integration Test (SIT)
    2. User Acceptance Test (UAT)

You will define the JPetStore application project in terms of three components.

JEE Application Files – JPetStore Application Component. WAR file that contains the main pieces of the JPetStore application.

Database Scripts - Scripts and supporting files to create the JPetStore database. Subsequent versions of this component contain scripts to modify the database schema.

Web Files - Static web content for JPetStore, such as web page image files.

When you have completed this lab, you will have an application defined with its three components.

4.2 Create the JPetStore JEE Component

Components are the basic building block of the application you model in UrbanCode Deploy. Over time, a component changes (as the developers improve it). UrbanCode Deploy can keep track of, and deploy any version of any component anytime the situation demands.

When a developer builds an application, they create a unit that was built from multiple source files. That new unit is typically given a version number of its own. CAUTION – Please keep in mind that 'version' used in the context of UrbanCode Deploy is not referring to source code – but a version of a compiled artifact that has been built and dropped into a common area (file system, repository or asset management system) where it waits to be deployed onto a runtime platform. This is where UrbanCode picks up in the lifecycle of the software product.

4.2.1 Create the JEE Component

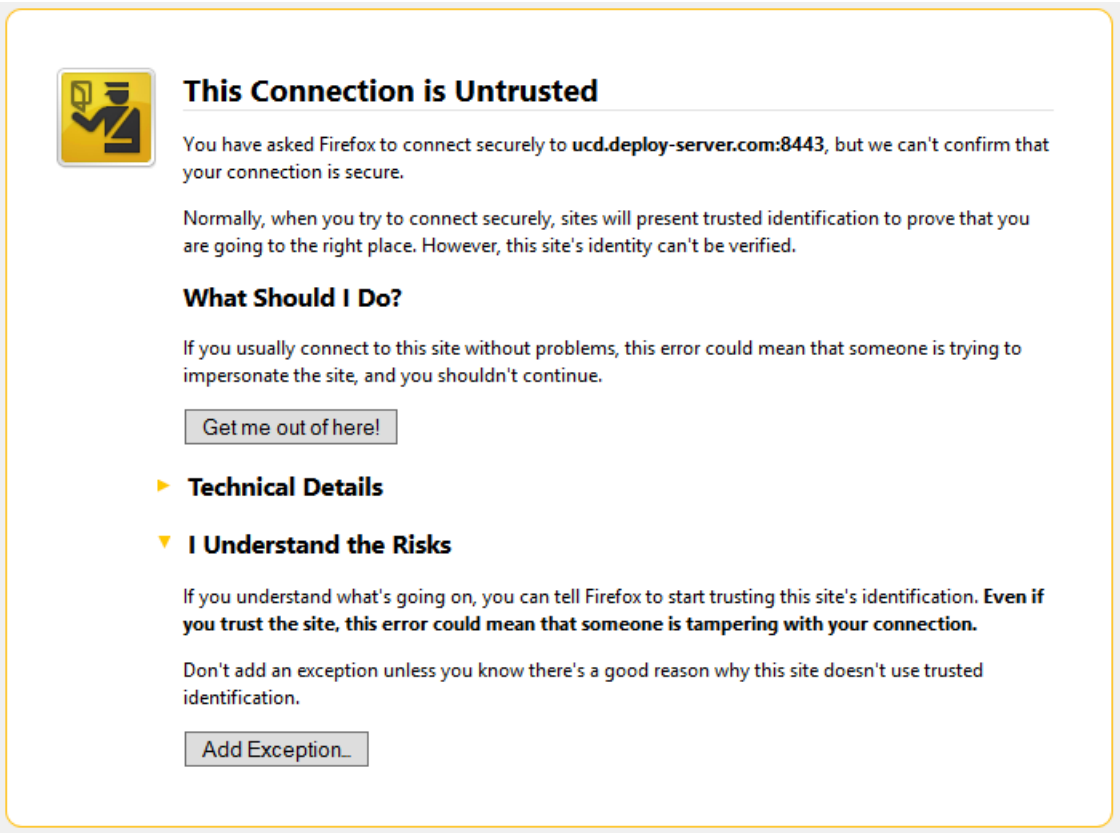
If it is not already running, launch a browser and log into UrbanCode Deploy.

Open a browser window and enter the URL:

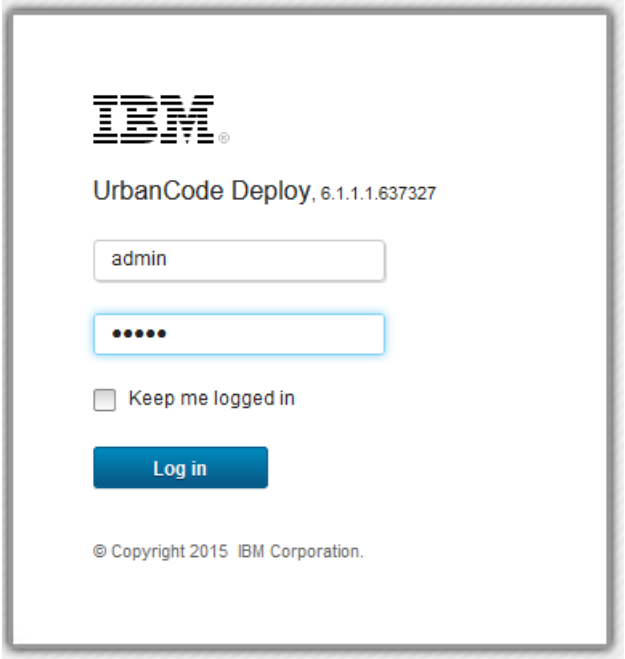
`http://ucd.deploy-server.com:8080`

TIP – you may want to bookmark this site for future reference.

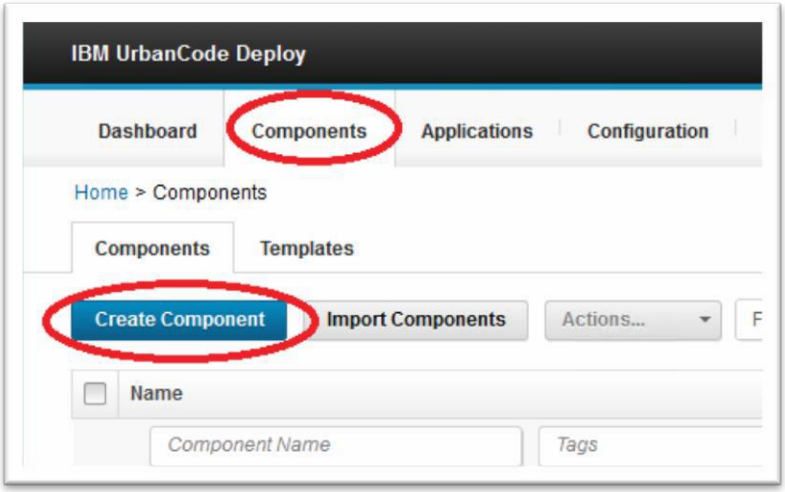
If you are challenged with security warning about the connection, add an exception (this is just accessing the local server). Below is the window Firefox shows:



Log in using `admin` for both the **user name** and **password**.

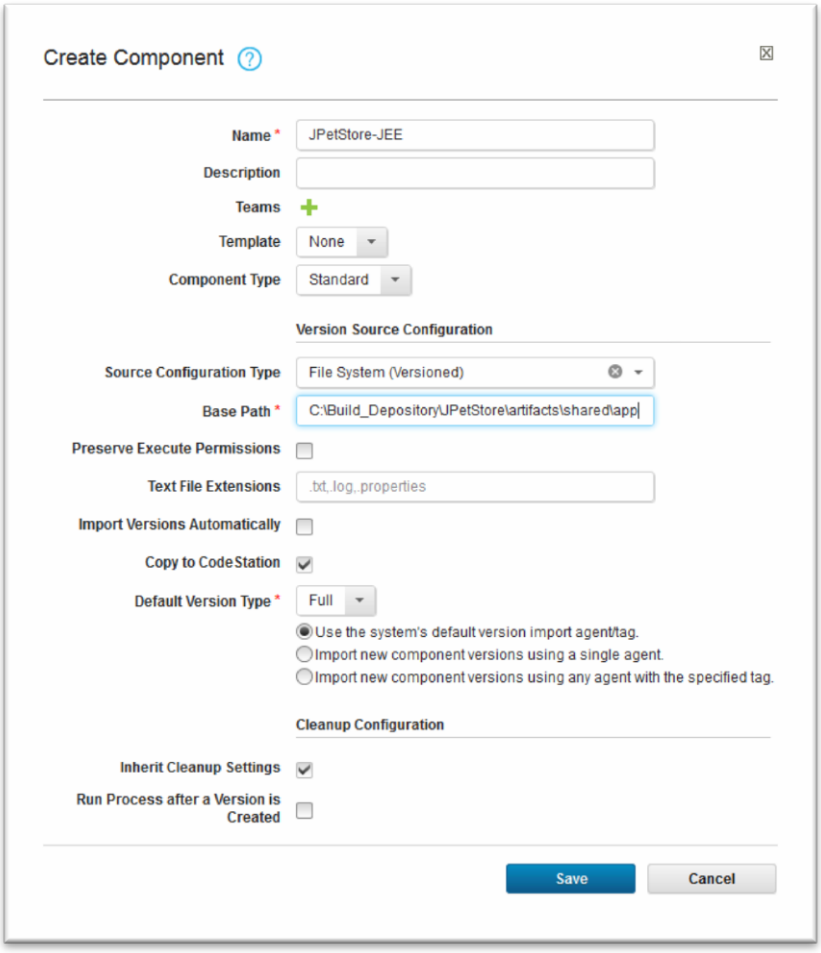


Click on the **Components** tab then on the **Create Component** button.



Name the new component `JPetStore-JEE`. Set other values as shown below:

Source Config Type = `File System (Versioned)`
Base Path = `C:\Build_Depository\JPetStore\artifacts\shared\app`



Click the **Save** button at the bottom of the pop-up (NOTE – you may have **to scroll down** to see the **Save** button.)

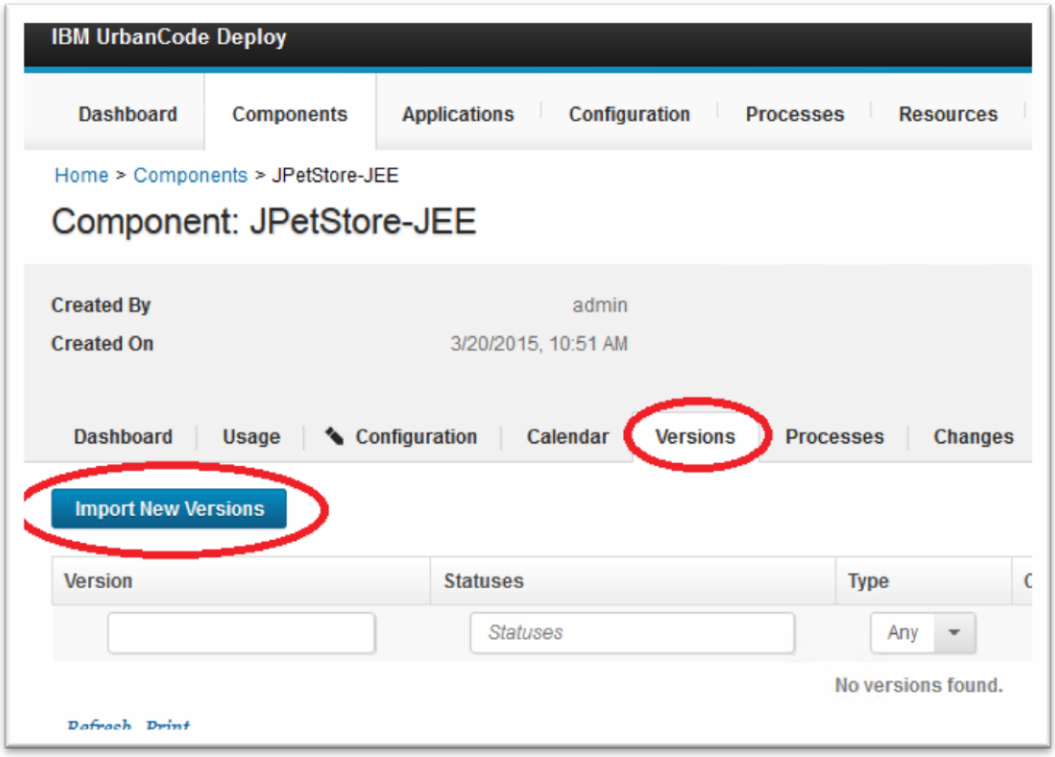
This creates the component and immediately opens the **Dashboard** tab of the component (lower row of tabs in UrbanCode Deploy). To see the values you just entered, click on the **Configuration** tab. Here you could update or change any of the original settings you entered.

By using the **File System (Versioned)** setting, you are declaring that each folder under the **Base Path** folder is actually a named version of the component. Each of the versioned folders actually contains the files for that version. When a developer builds another version of the component and deposits it here, UrbanCode Deploy can sense this update and make the new version available for deployment (or even automatically deploy the new version if desired).

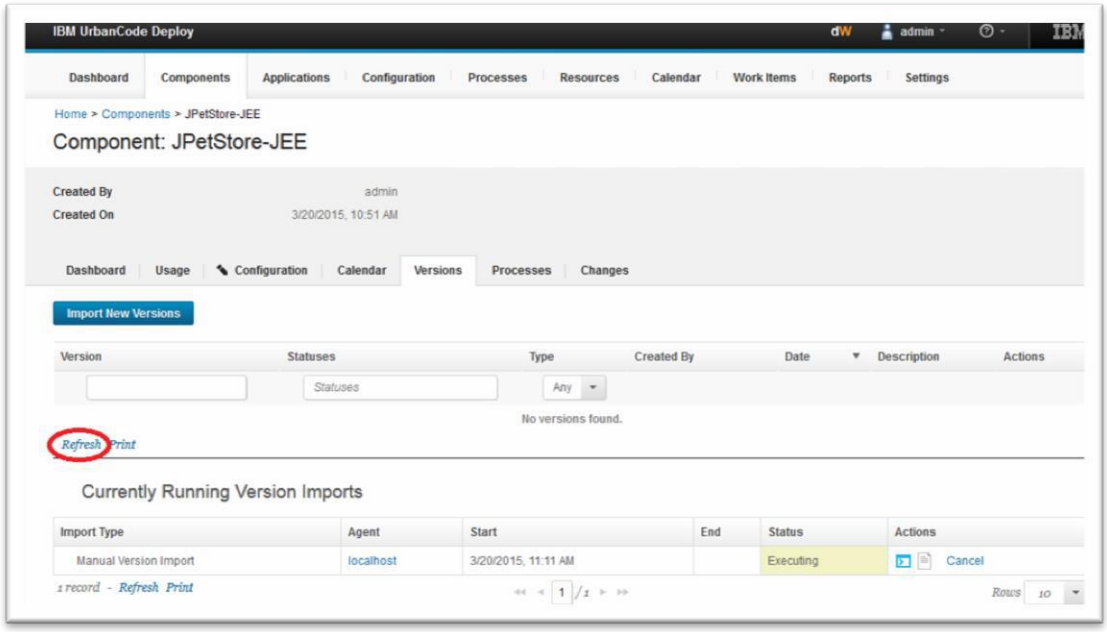
4.2.2 Import Version 1.0 of the JEE Component

Click on the **Versions** tab from the lower row of tabs.

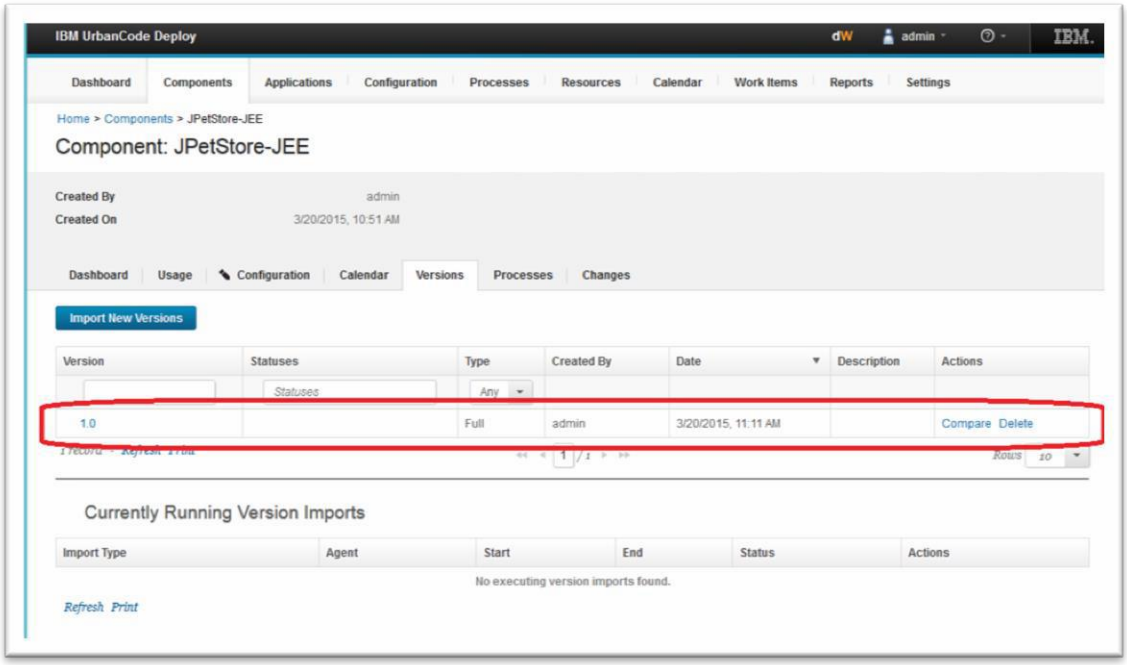
Click on the **Import New Versions** button to load any versions that are found in the file system. For the JEE component, the only version defined is 1.0.



Wait a moment and click the **Refresh** link *above* the section named **Component Request History**. (UrbanCode Deploy does not automatically update this screen after the version import.)



This should refresh the page and you should see a new line indicating the component file(s) have imported successfully.



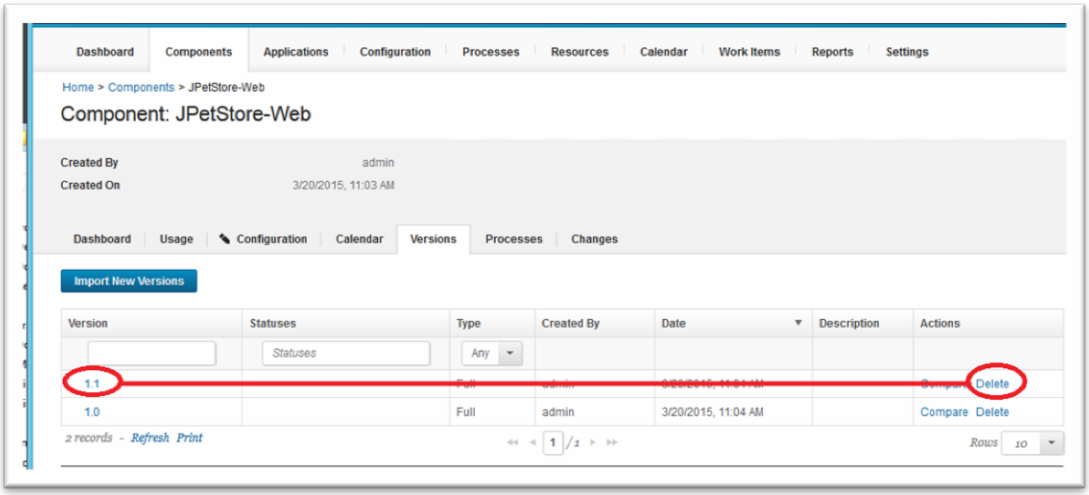
4.3 Create JPetStore WEB and DB Components

4.3.1 Create JPetStore-Web and import new versions.

Repeat the steps above to create and import another component called `JPetStore-WEB`. Set other values as shown below:

Source Config Type = `File System (Versioned)`
Base Path = `C:\Build_Depository\JPetStore\artifacts\shared\web`

When you import the versions, two versions are loaded 1.0 and 1.1. Initially, you only want to have version 1.0. Later you will add version 1.1 and update the deployment. For now, click on the **Delete** link to the far right of version 1.1 to delete that version.



This should leave only the row indicating Version 1.0 of this component. NOTE – this action does not remove the component files from the repository – it only removes the reference from UrbanCode Deploy. Later in the labs we will re-import this component to illustrate how UrbanCode Deploy can discover new versions that have been made available by developers.

4.3.2 Create and import JPetStore-DB

Repeat the steps one more time to create a new component named **`JPetStore-DB`**. Set other values as shown below:

Source Config Type = **`File System (Versioned)`**
Base Path = **`C:\Build_Depository\JPetStore\artifacts\shared\db`**

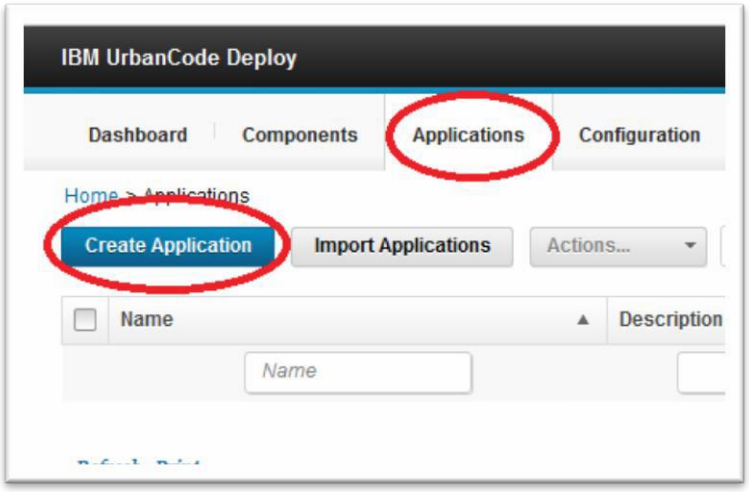
As with the previous component, delete all the imported versions except for version 1.0.

4.4 Create and Define the JPetStore Application

Now that you have defined the components for JPetStore, you can create an application that includes those components.

4.4.1 Create the JPetStore application

Click on the **Applications** tab from the *upper* row of tabs, then click on the **Create Application** button.
.

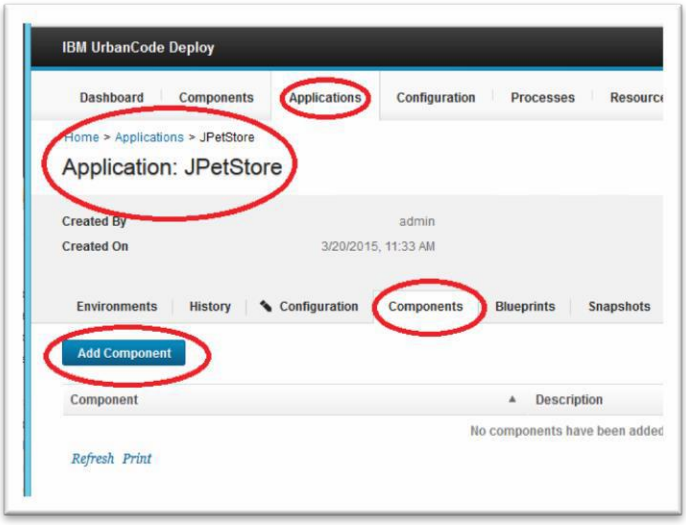


Enter `JPetStore` as the name of the application and then press the **Save** button.

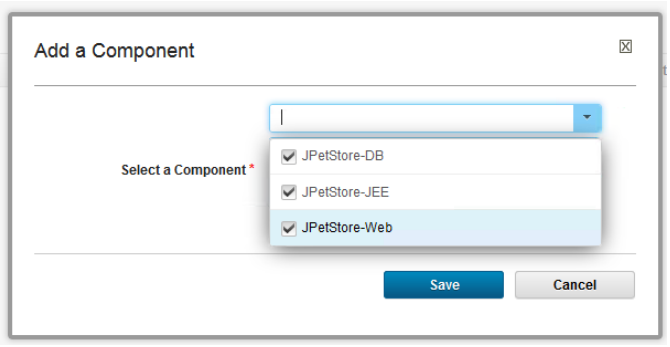
4.4.2 Add the components to JPetStore

UrbanCode Deploy should now be focused on the new application object.

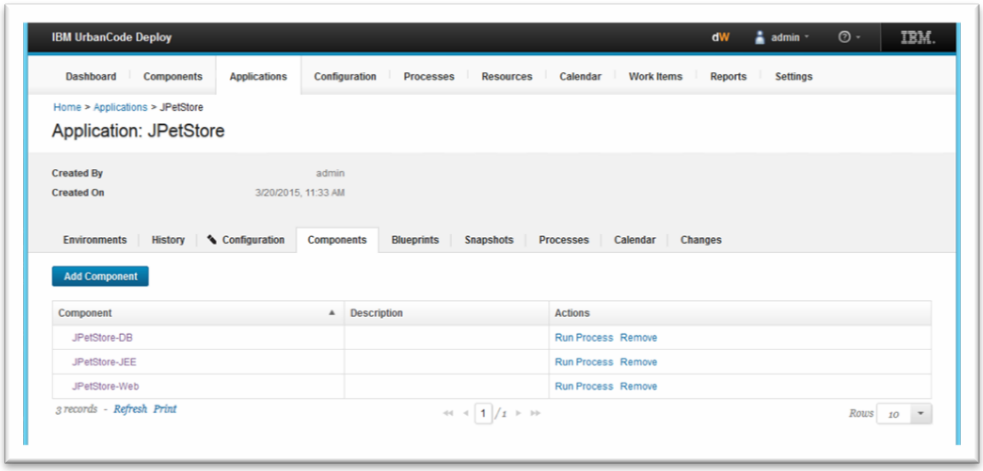
Select **Components** from the *lower* set of tabs and click the **Add Component** button as shown.



Use the dropdown box in the dialog and check all three components (`JPetStore-JEE`, `JPetStore-DB` and `JPetStore-Web`) then click the **Save** button.



Your screen should now look like the picture below:



4.5 Create Environments for the JPetStore Application

So far we have created **components** and the **application** to which they belong in the UrbanCode representation model.

Next we will represent the different computing **environments** into which we will be deploying the JPetStore application.

The UrbanCode Deploy environments represent the groups of target machines (resources) onto which it will deploy the components of the JPetStore application. As part of the software development lifecycle, application pieces start in the computing environments of the development group, then move through several levels of testing, and finally are delivered to the operations group for deployment into production servers. UrbanCode Deploy can model the computing environments of each of those groups (who will certainly have different computing systems built to suite their own specific needs). Each of those groups of computing environments are represented in UrbanCode Deploy as "**Environment**" objects.

An UrbanCode Deploy architect will generally create separate Environment objects to represent each of the stages of a customer's software development life cycle (i.e., Development, System Integration Testing, User Acceptance Testing, Performance Testing, Production, etc.). UrbanCode Deploy can keep track of which versions of the pieces of an application that are deployed into each of those environments, and provides nice features to allow administrators to

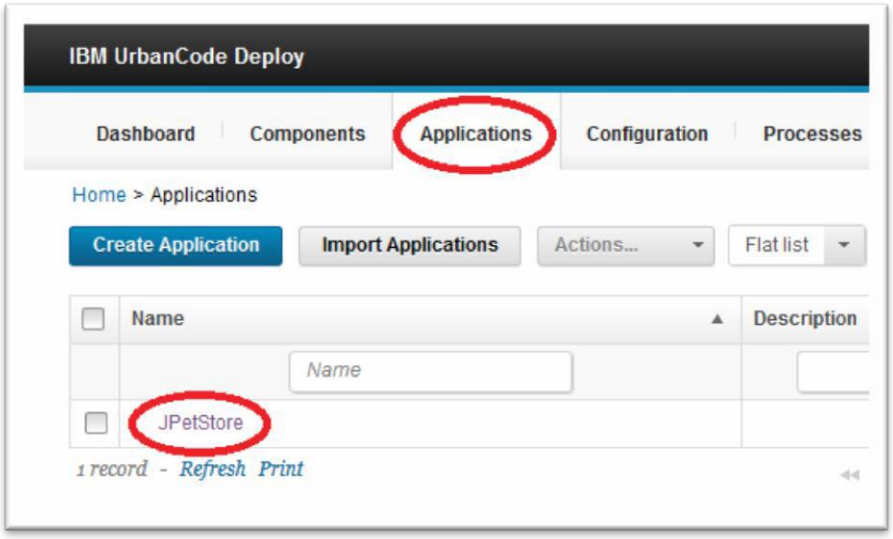
move the pieces through the development stages as a unit (snapshots… but more on that later).

For this lab, we will create only two environments to illustrate the value of UrbanCode Deploy – representing System Integration Test (SIT) and User Acceptance Test (UAT).

Modern web applications are comprised of different components that are usually deployed onto separate computer systems. For instance, Database Servers are usually run on their own dedicated systems in all environments except maybe a developer's test environment. Since we are using only one physical machine for our lab, we have taken steps to simulate an environment that is comprised of multiple machines. This way the UrbanCode Deploy representation model will look more like a real-world computing topology with multiple computer systems. We have laid the groundwork for this in the earlier sections of this document.

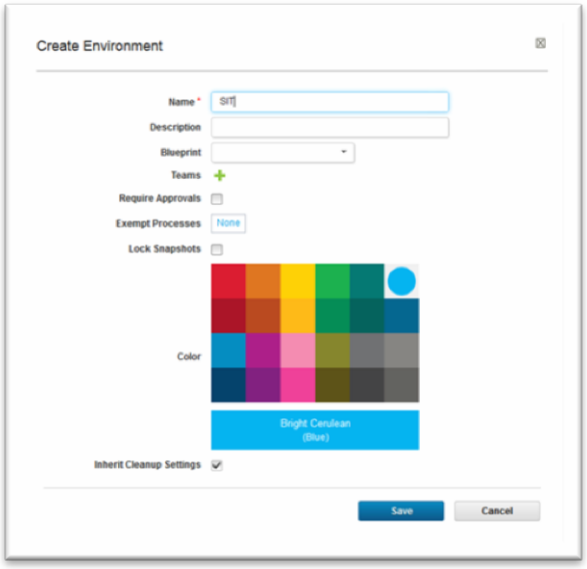
4.5.1 Create SIT and UAT Environments

Click on the **Applications** tab (upper row of tabs) to see the list of defined applications. Click on the link for the `JPetStore` application.



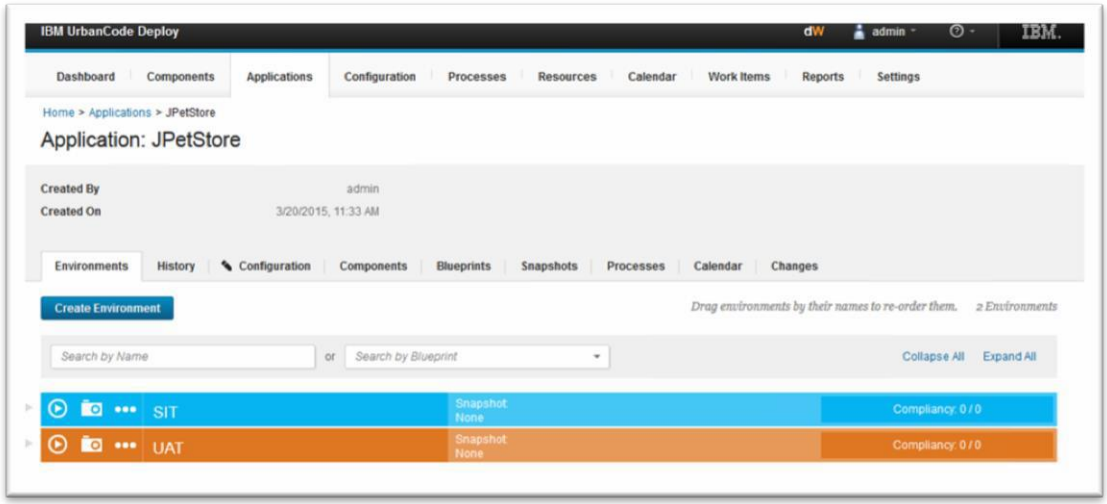
In the lower row of tabs, click the **Environments** tab, then click the **Create Environment** button. Name the new environment `SIT`.

Set the Color to `Bright Cerulean (Blue)`. Click the **Save** button.



Repeat the previous steps in this section to create another environment named **UAT**. Set the **Color** to **Cinnamon (Orange)**.

Verify this task by clicking on the **Applications** tab to see the list of defined applications and then click on **JPetStore** to open it. You should see the **SIT** and **UAT** Environments as shown below:



4.6 Summary

You have defined an application in UrbanCode Deploy called **JPetStore** and loaded version 1.0 of the application's three components.

You have defined two environments associated with this application, named **SIT** and **UAT**.

5 Defining Computer Resources and Groups

5.1 Overview

In the following lab section, we will review the simple topology of the networks in our data center. This particular Data Center houses the machines that support:

- System Integration Test (SIT)
- User Acceptance Test (UAT)

The application, JPetStore, is a simple 3-Tier web application that consists of:

- Static Web Content
- Simple Business Logic
- Database Storage

Each of the pieces of the application are developed and managed by different groups.

Each of them is on a separate development schedule (and will produce different versions at different times), but all are designed to work together in order to function as a single application.

All the pieces must progress through the same software development lifecycle, which includes the two testing environments we've mentioned.

System Integration Test

The **System Integration Test** phase of development is managed by developers. The purpose of this phase is simply to confirm that all the pieces of the application will run, and no changes made to any single piece will prevent the application from operating as-designed.

This group runs all the pieces of the JPetStore application on a single machine (for administrative simplicity).

User Acceptance Test

The **User Acceptance Test** phase of development is managed by the QA group. The purpose of this phase is to test the pieces of the application for purposes of verifying that the application is satisfying the needs of the end users.

This group wants to run the application in a computing environment that more closely represents the production environment. The pieces of the JPetStore application are installed on two separate machines (one for the JEE and Web pieces, and a second dedicated machine for the DB piece).

For our lab, we have only one physical machine to use. But in the section for Machine Setup, we took steps to be able to mimic a multi-machine environment (by using different port numbers, and making several entries in the machine's hosts file.) In the next several labs we will build a model of resources in UrbanCode Deploy that will look more like a real-world environment (using multiple machines and different topologies between groups/environments). One major advantage of UrbanCode deploy is its ability to be able to distinguish between different topologies, and make adjustments at deployment time to automatically accommodate those differences.

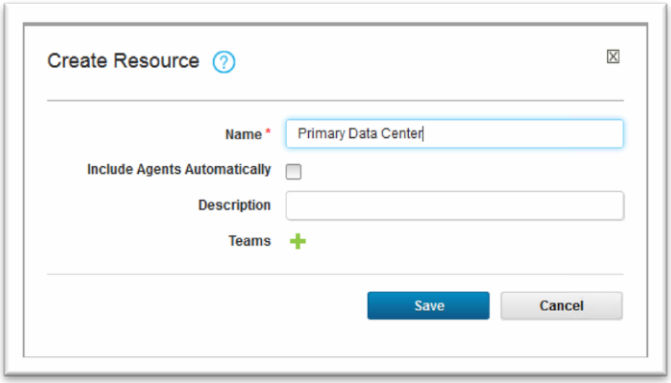
5.2 Defining Compute Resources in UrbanCode

5.2.1 Define a Resource Hierarchy to reflect the Data Center Topology

Select **Resources** from the upper row of tabs.

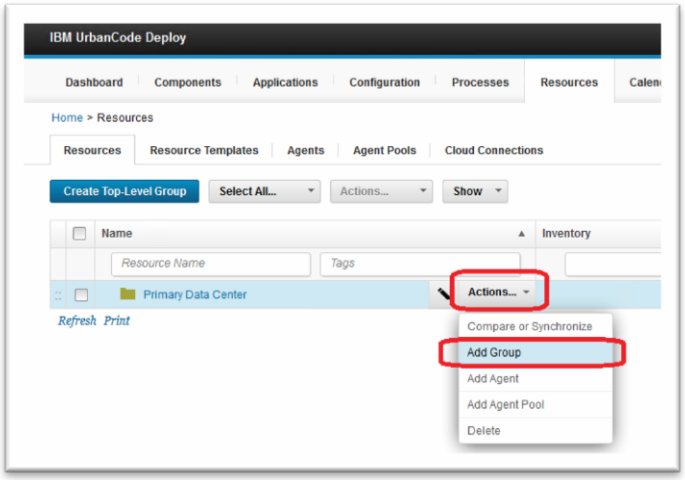
Make sure the lower row of tabs also has the **Resource Tree** tab selected.

Click the **Create Top-Level Group** button. Name the group `Primary Data Center`. Click the **Save** button.

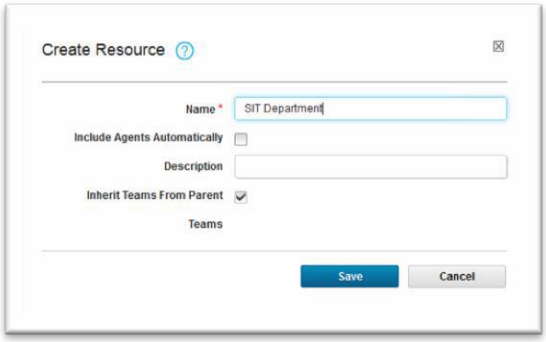


Find the new group in the list, and hover your mouse over it which will produce the **Actions…** context menu.

Click the **Actions…** context menu then select **Add Group** from the drop down list.



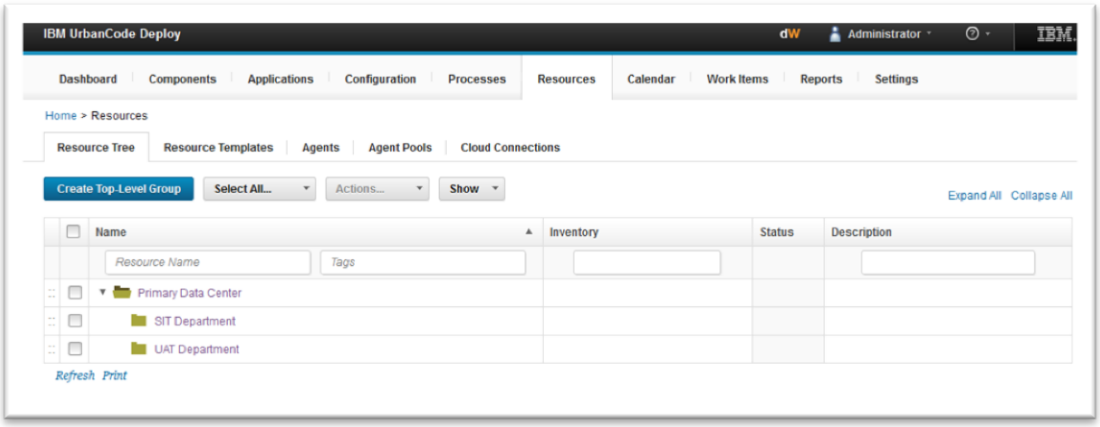
Name the new group `SIT Department`.



A new folder should appear as a child of the **`Primary Data Center`** folder in the browser.

Repeat these steps to add another child folder to the **`Primary Data Center`** named **`UAT Department`**.

Your screen should now look like this:



5.2.2 Add Computing Resources to Each Group

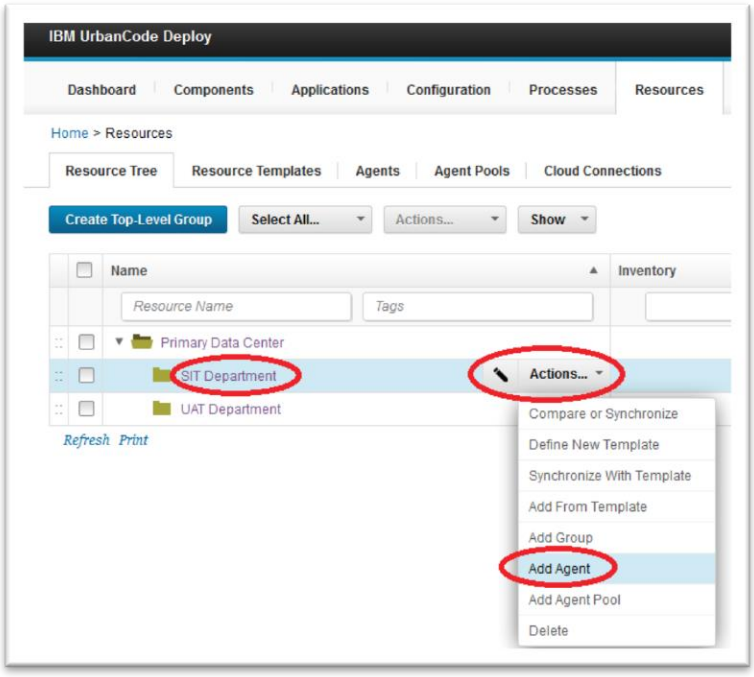
Now we will add the machines (agents) to our model.

Every machine with which UrbanCode Deploy interacts must have the Agent software running. When the Agent software is installed, we identified the UrbanCode Deploy server (hostname or IP address) to which it will communicate. This way, when the Agent software runs it calls back to the server to make itself known… and will appear in UrbanCode Deploy as a member of a list of agents available to do work for us.

In the earlier section, we described our topology as having three separate machines (one in the SIT group, and two in the UAT group). We will now create the representations for these machines (agents) in our model.

Hover your mouse over the entry for the **`SIT Department`** and the **Actions…** context menu will appear on that row, a little further to the right.

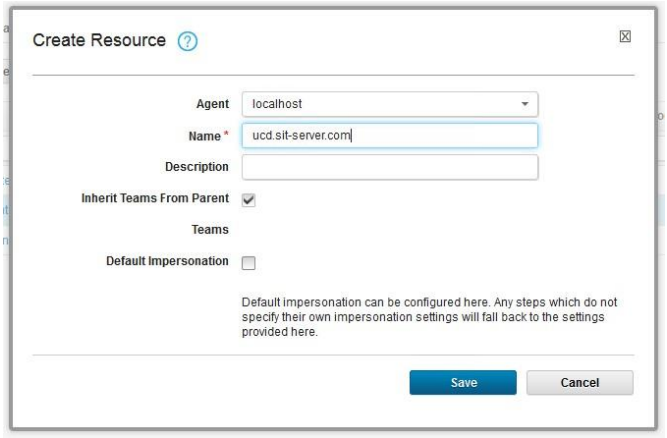
Click the **Actions…** context menu and select **Add Agent**.



In the **Agent** field, set the drop down box to `localhost` (our only selection, because we only have one actual Agent installed).

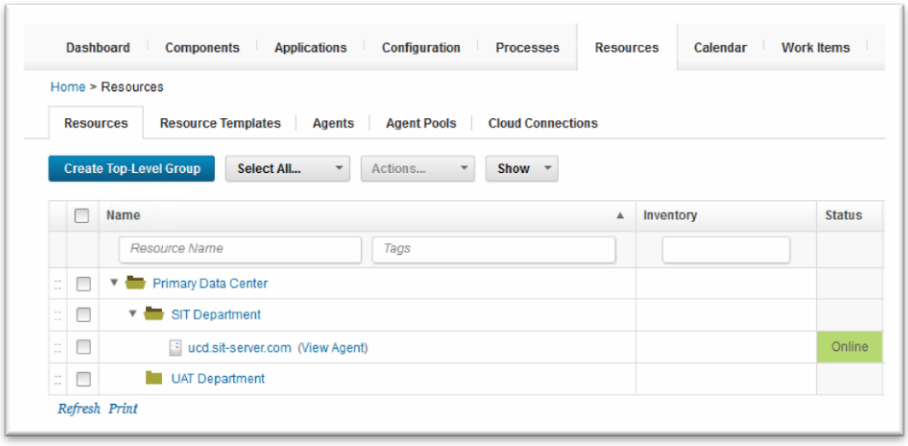
Change the **Name** field to:

```
ucd.sit-server.com
```



Click the **Save** button.

Your screen should look like the picture below:



Now hover your mouse over the **`UAT Department`**, and use the **Actions…** context menu to **Add Agent** to *that* group.

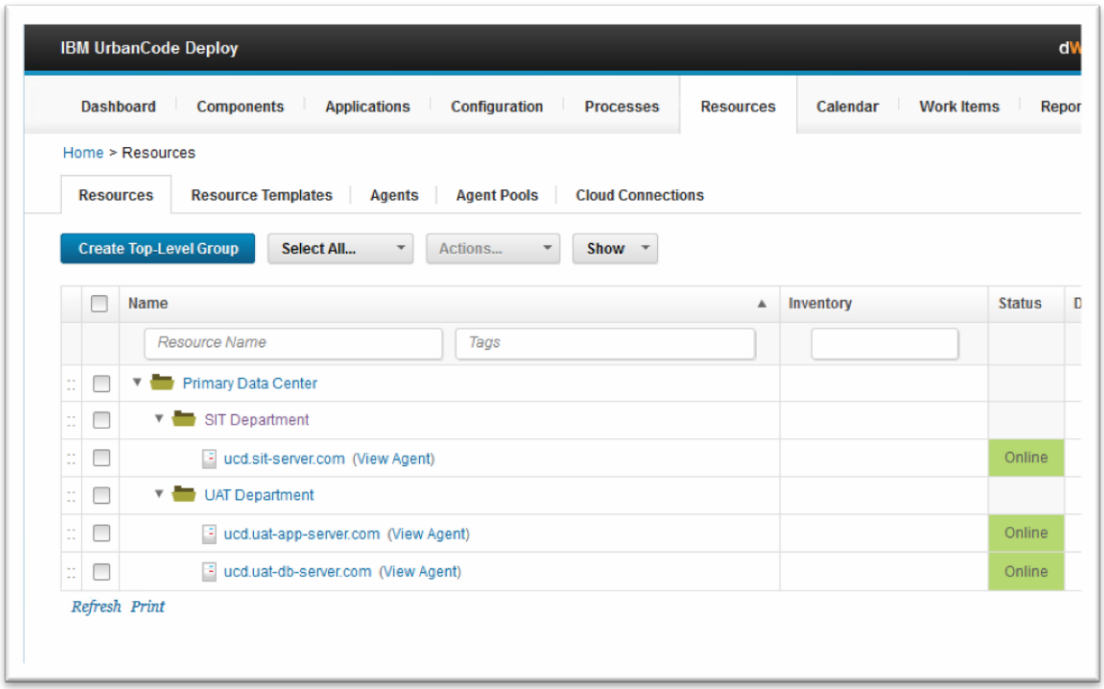
Again, set the **Agent** dropdown to **`localhost`**. Change the **Name** field to:

`ucd.uat-app-server.com`

Add <u>one more</u> Agent to the **`UAT Department`**… The **Agent** dropdown must again be **`localhost`**. Change the **Name** field to:

`ucd.uat-db-server.com`

Your screen should now look like the picture below:



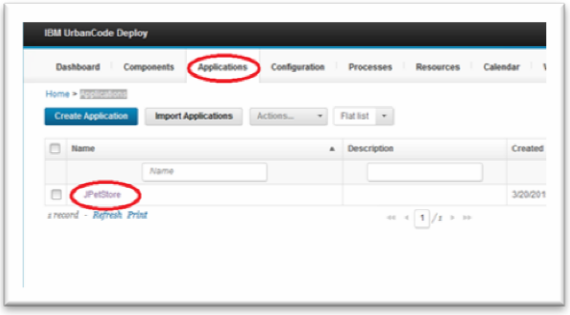
5.3 Associating Environments with Resources and Components

Now we've described our Data Center resources. Next we tell UrbanCode Deploy which of those resources are assigned to each of the environments (SIT and UAT)

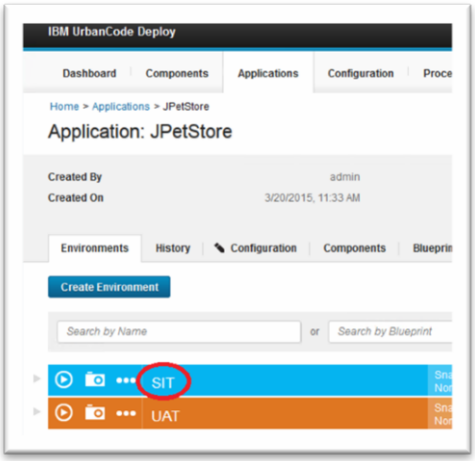
5.3.1 Identify the resources for the SIT environment

Click the **Applications** tab in the top row of tabs.

Click on the link for the `JPetStore` application. This will display the two environments we created earlier (SIT and UAT).



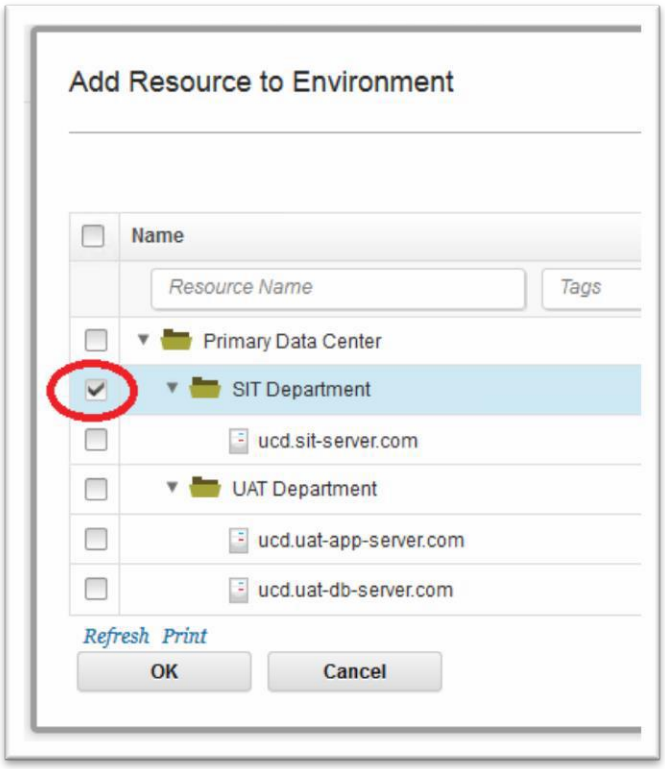
Click on the `SIT` environment (click on the text for `SIT` as shown below).



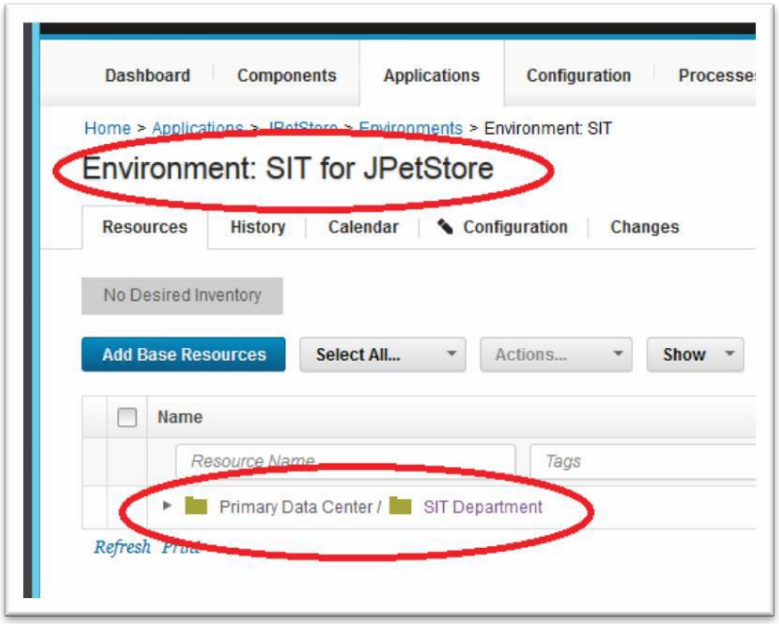
From the **Resources** tab in the lower row of tabs click on the `Add Base Resources` button.

	*Note that some tabs on the upper and lower rows are named the same. Context is important… take note when you are asked to select a tab if it is on the upper or lower row!*

In the resulting list, click the check box next to the folder named SIT Department. This will include all the children of that object as well (in this case, that means the server named `ucd.sit-server.com`). In this way we tell UrbanCode Deploy which resources (machines) are associated with the SIT department.

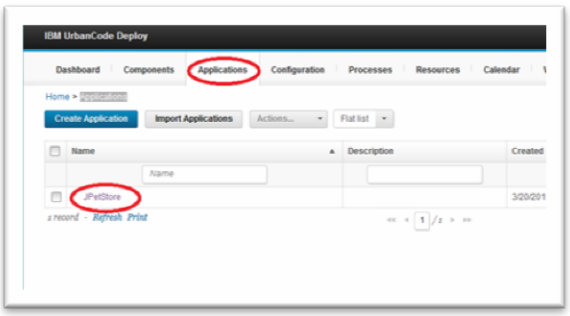


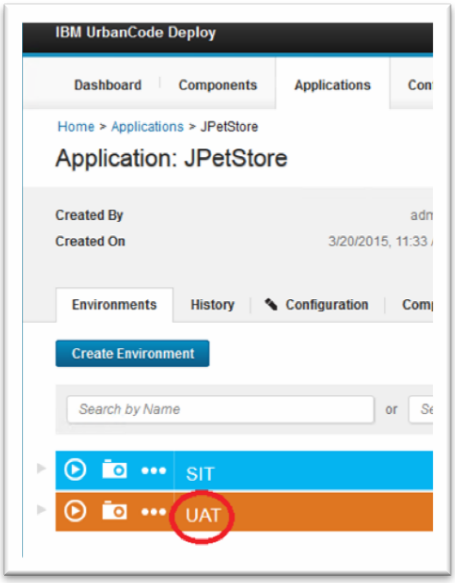
Click the **OK** button. The environment (SIT) should now display the folders we just indicated.

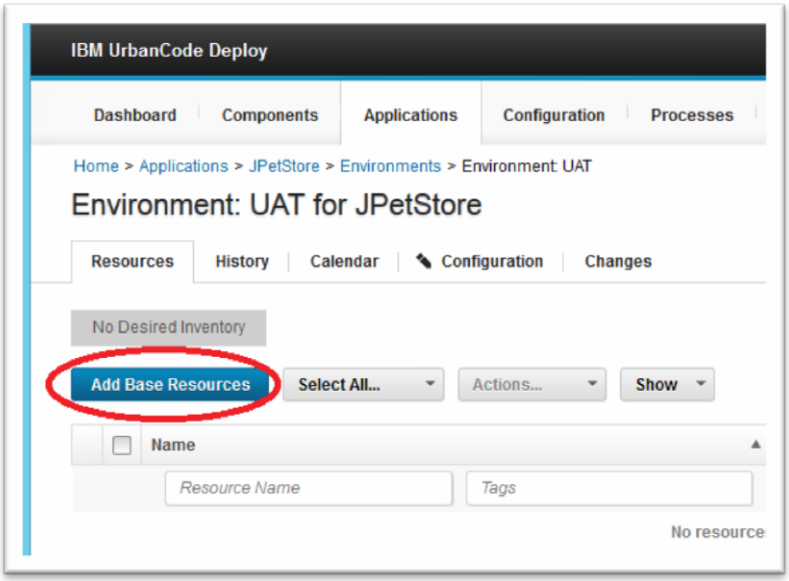


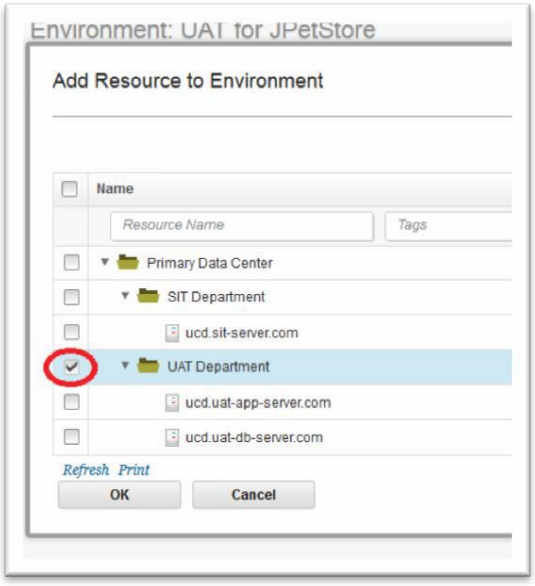
5.3.2 **Identify resources for the UAT environment**

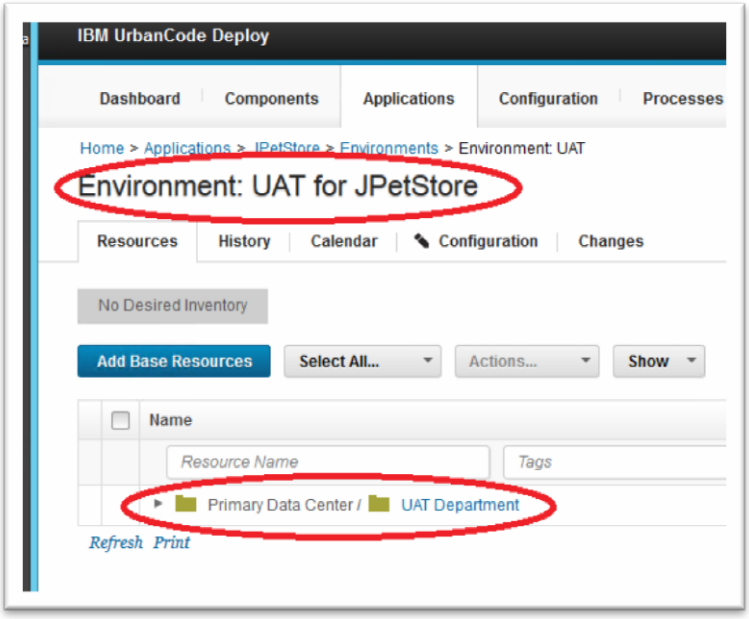
Repeat these steps in the previous section to add the **`UAT Department`** resource group to the **`UAT`** environment.




IBM UrbanCode Deploy
Dashboard
Components
Applications
Home > Applications > JPetStore
Application: JPetStore
Created By
Created On
Environments
History
Configuration
Create Environment
Search by Name
SIT
UAT


IBM UrbanCode Deploy
Dashboard
Components
Applications
Configuration
Processes
Home > Applications > JPetStore > Environments > Environment: UAT
Environment: UAT for JPetStore
Resources
History
Calendar
Configuration
Changes
No Desired Inventory
Add Base Resources
Select All...
Actions...
Show
Name
Resource Name
Tags
No resource

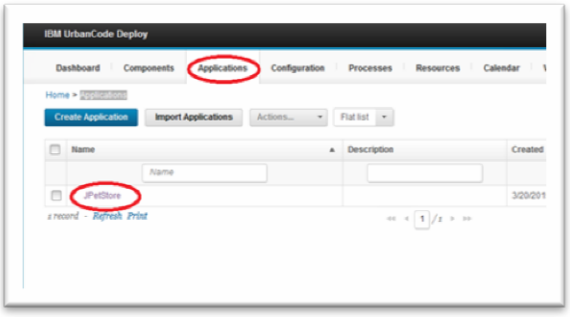




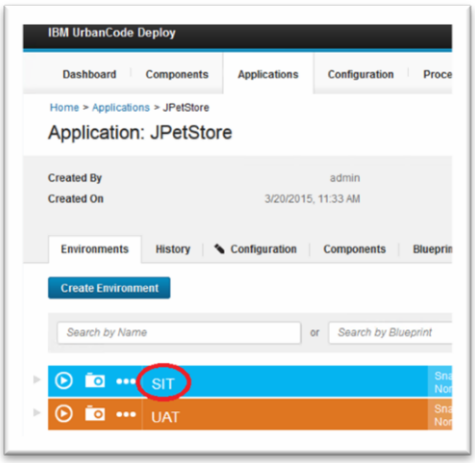
5.3.3 Assign the Application Components to be installed on the Resources in the SIT Environment

Click the **Applications** tab in the top row of tabs.

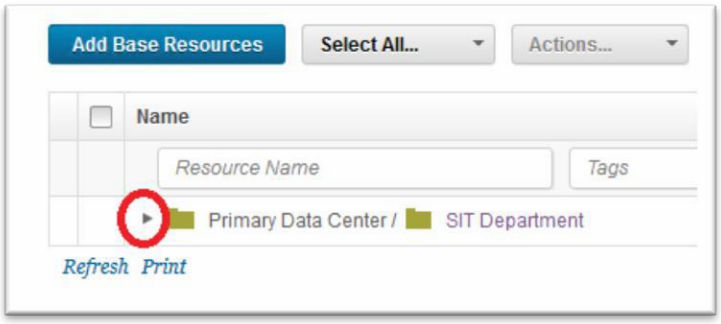
Click on the link for the **`JPetStore`** application. This will display the two environments we created earlier (**`SIT`** and **`UAT`**).



Click on the **`SIT`** environment (click the text for SIT as shown below).

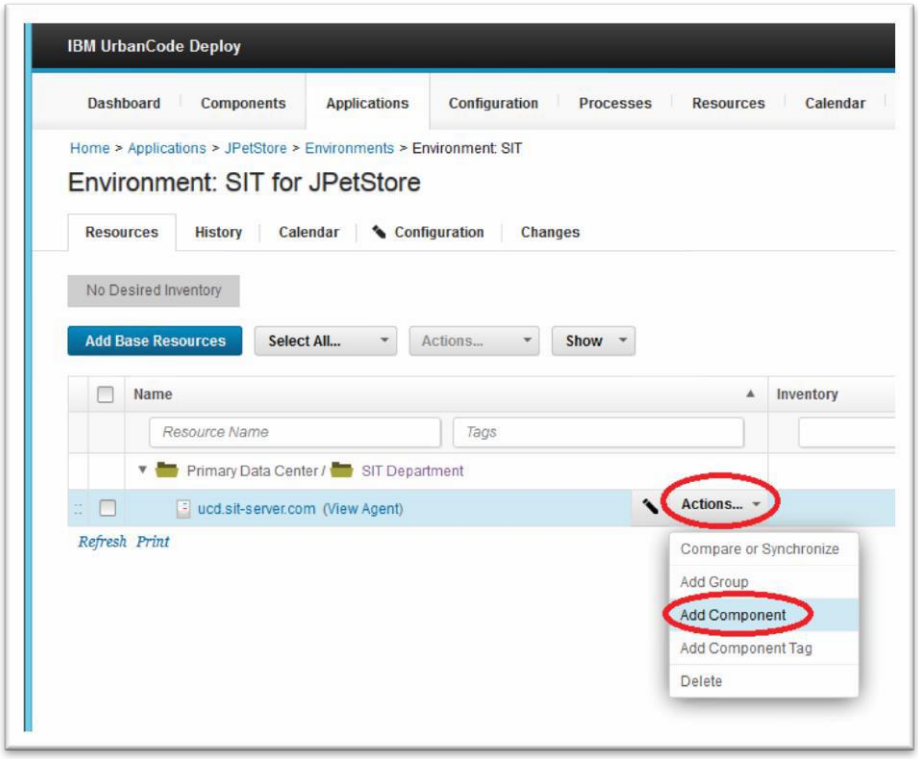


Click the 'twistie' control to expand the tree under the resource groups.

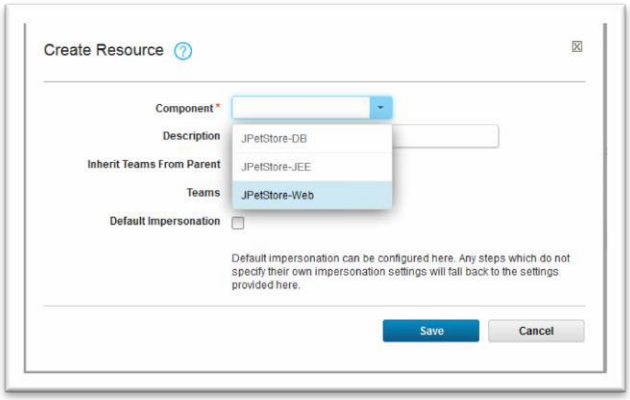


You should see the resource **`ucd.sit-server.com`** and the **Actions…** menu to the right. (If you do not see the **Actions…** menu, hover your mouse over the **`ucd.sit-server.com`** resource and it will appear.)

Click the **Actions…** menu and select **Add Component**.

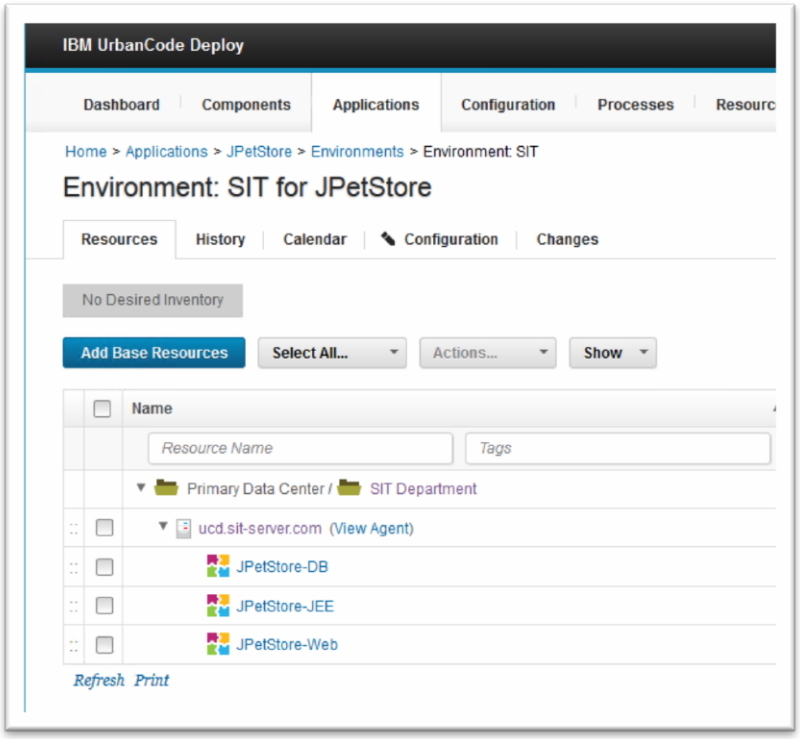


In the **Create Resource** dialog box, set the **Component** dropdown to `JPetStore-Web` and hit the **Save** button.



Repeat this last step twice more to add `JPetStore-JEE` and `JPetStore-DB` to the `ucd.sit-server.com` resource as well.

Your screen should now look like this:

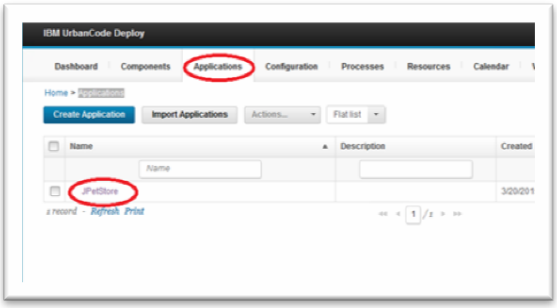


In this way, we've told UrbanCode Deploy that these three components (`JPetStore-DB`, `JPetStore-JEE` and `JPetStore-JEE`) will all be installed on the same machine, namely `ucd.sit-server.com`.

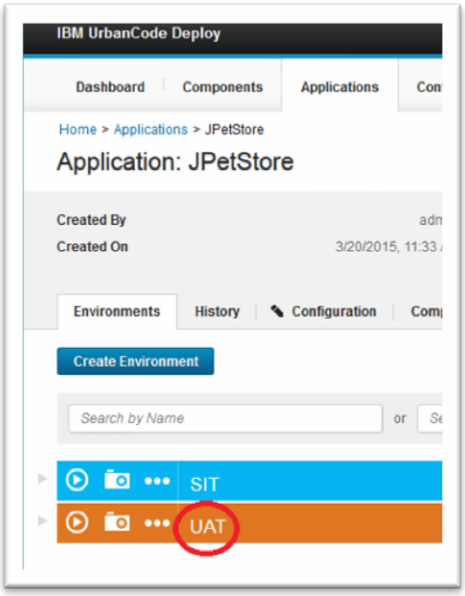
5.3.4 Assign the Application Components to be installed on the Resources in the UAT Environment

Click the **Applications** tab in the top row of tabs.

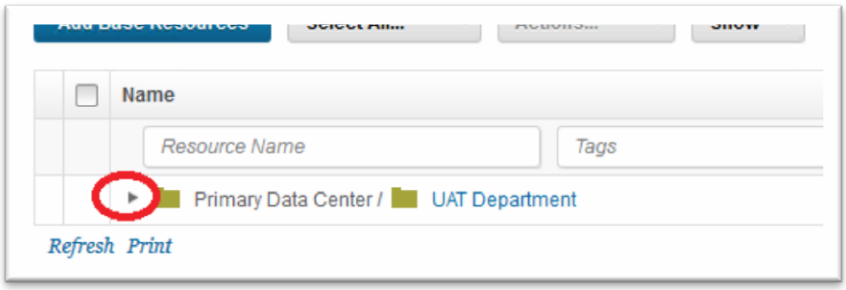
Click on the link for the `JPetStore` application. This will display the two environments we created earlier (`SIT` and `UAT`).



Click on the `UAT` environment (click the text for `UAT` as shown below).



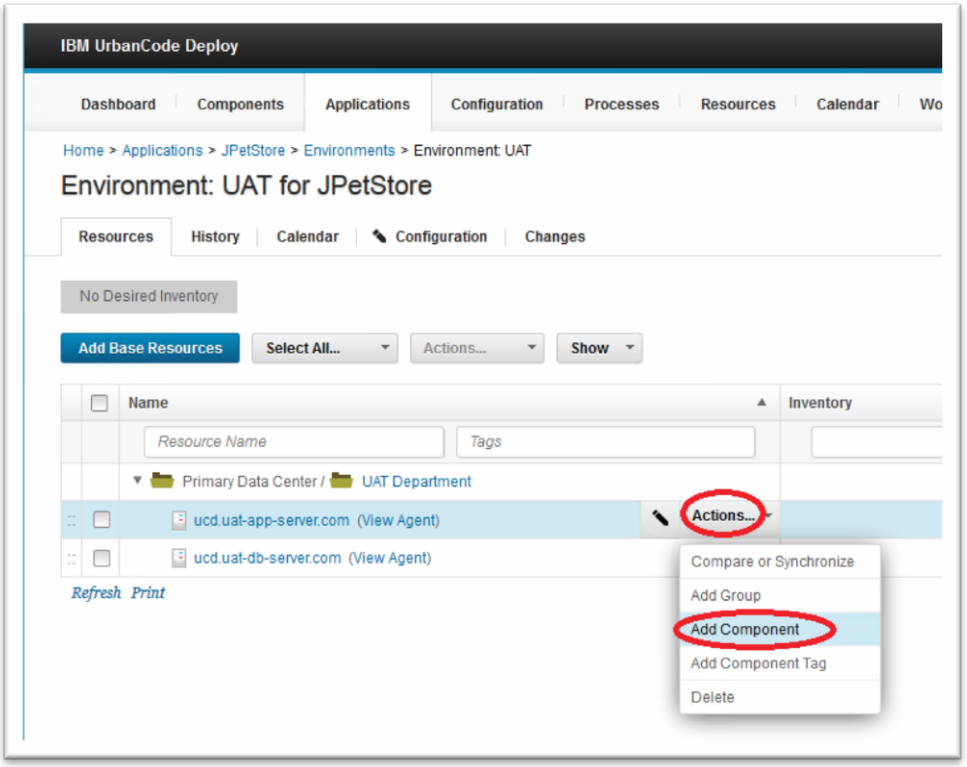
Click the 'twistie' control to expand the tree under the resource groups.



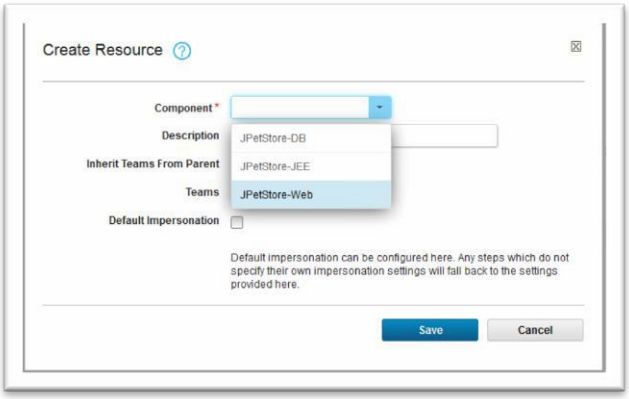
You should now see the two resources we assigned to the UAT Department:

- `ucd.uat-app-server.com`
- `ucd.uat-db-server.com`

Hover your mouse over ucd.uat-app-server.com and you should see the **Actions…** menu appear to the right. Click the **Actions…** menu and select **Add Component**.

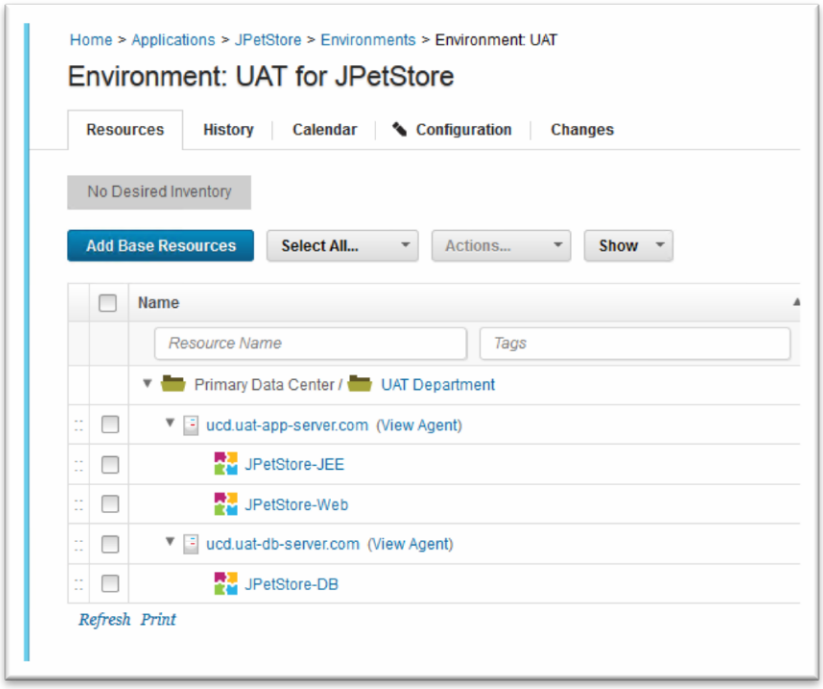


In the **Create Resource** dialog box, set the **Component** dropdown to **JPetStore-Web** and hit the **Save** button.



Repeat this last step once more to add **`JPetStore-JEE`** to the **`ucd.uat-app-server.com`** resource as well. (Do **NOT** add **`JPetStore-DB`** to this resource!!)

Next, use this same technique to add the **`JPetStore-DB`** component to the OTHER resource (machine), **`ucd.uat-db-server.com`**. Your screen should now look like this:



As the name of the machine implies, **`ucd.uat-db-server.com`** is a database server. It is almost always the case that a database server is a dedicated server with no other processes or services running on it. We have designed the UAT environment to reflect this… with two application components installed on **`ucd.uat-app-server.com`** and the database component installed on **`ucd.uat-db-server.com`**. We do this to demonstrate the capability of UrbanCode Deploy to accommodate different topologies in different environments that still require the same application. (Please recall that the **`SIT`** environment has a different physical topology… only one machine named **`ucd.sit-server.com`** on which all three application components are installed.)

After we encode the deployment processes (in the upcoming sections), the UrbanCode Deploy users simply have to push a single button to deploy to any environment. This is a very powerful feature of UrbanCode Deploy that can enable authorized managers to initiate deployments without having to be application experts or IT specialists (or rely on others for those skills). One common scenario is that a QA manager can initiate a deployment for their test groups without having to wait for the developers or release engineers to become available. This can dramatically reduce the time it takes to schedule and conduct QA operations.

6 UrbanCode Component Processes

UrbanCode Deploy we can create processes to do whatever work is required. Processes can be attached to Application objects, Component objects, or they can be unattached general processes.

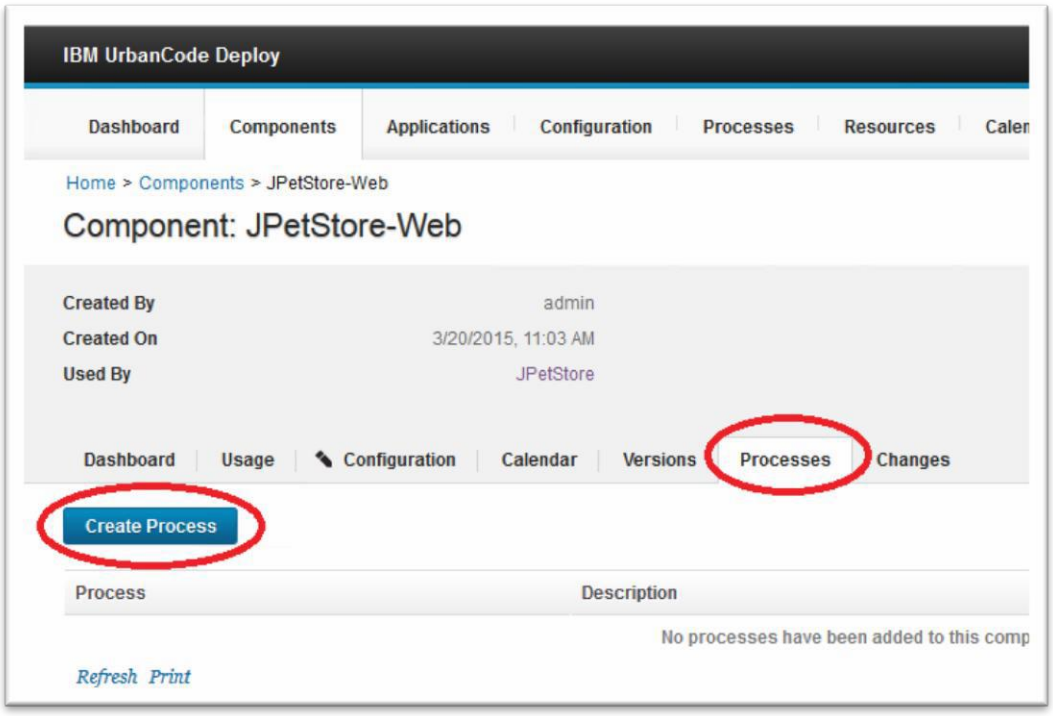
A Component Process is a script for deploying or managing the deployment of a component. In this section, you define the Process for deploying the JPetStore-Web component to a target computer.

Since the UrbanCode Deploy Component contains the Process or Processes it needs to deploy itself, that information is available no matter which environment (or on which machine) it must be deployed. The process can also be parameterized to accommodate configuration differences in different environments (or on different servers) so the UrbanCode Deploy operator needs only indicate the action to deploy the Application – and does not need to be concerned with (or have the knowledge of) *how* to deploy it (because the UrbanCode Components, themselves, already know!). Again, this removes bottlenecks for authorized managers who might need to wait for skilled technicians before they can continue with their work.

6.1 Deployment Process for JPetStore-Web component

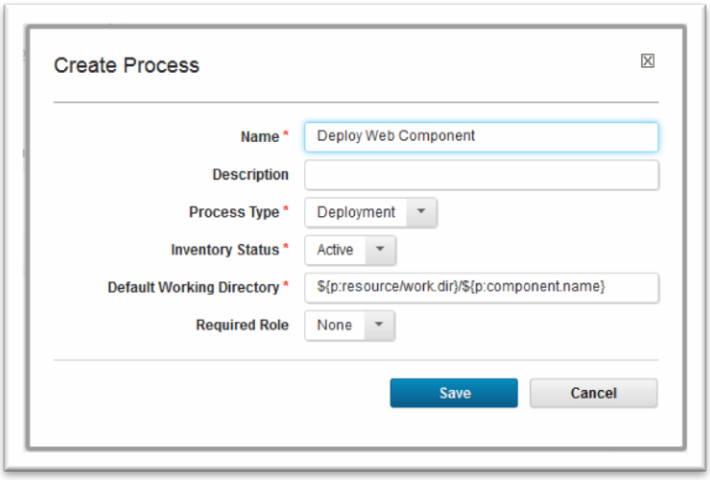
6.1.1 Create the new Process – Deploy Web Component

In the top row of tabs click the Components tab. In the resulting list, click **`JPetStore-Web`**.
In the lower row of tabs, click the Processes tab. (NOTE - please note the upper row of tabs also has a tab named Processes… you need to pick the LOWER tab to associate this process with the **`JPetStore-Web`** component.) Click the **Create Process** button.



Name the new process `Deploy Web Component`.

Make sure the **Process Type** field is set to `Deployment`.



Note the field **Default Working Directory**. Processes are executed by UrbanCode Agents on the target machine (resource). (They are NOT executed on the UrbanCode Server machine.) The **Default Working Directory** is a temporary area in which the UrbanCode Agent may deposit temporary files while it does its work.

The value of the **Default Working Directory** includes UrbanCode "**properties**", which allow us to parameterize processes. **Properties** are indicated by the characters `${p:` followed by the object to which they apply, and an identifier as to which property is desired. The property reference is terminated by a closing 'squiggle bracket' `}` .

In the **Default Working Directory** field there are two properties referenced:

`${p:resource/work.dir}`	the working directory on the machine (resource) where the UrbanCode agent is installed. This could vary from machine to machine. The value is set when the UrbanCode Agent is installed, and can be referenced by processes from then on.
`${p:component.name}`	the name of the component to which this process is attached. In our case, this is the `JPetStore-Web` component.

UrbanCode properties can be used in any editable field and are interpreted at runtime. There is a predefined and documented set of properties that are included with UrbanCode Deploy, and the operator can also create their own custom properties as needed.

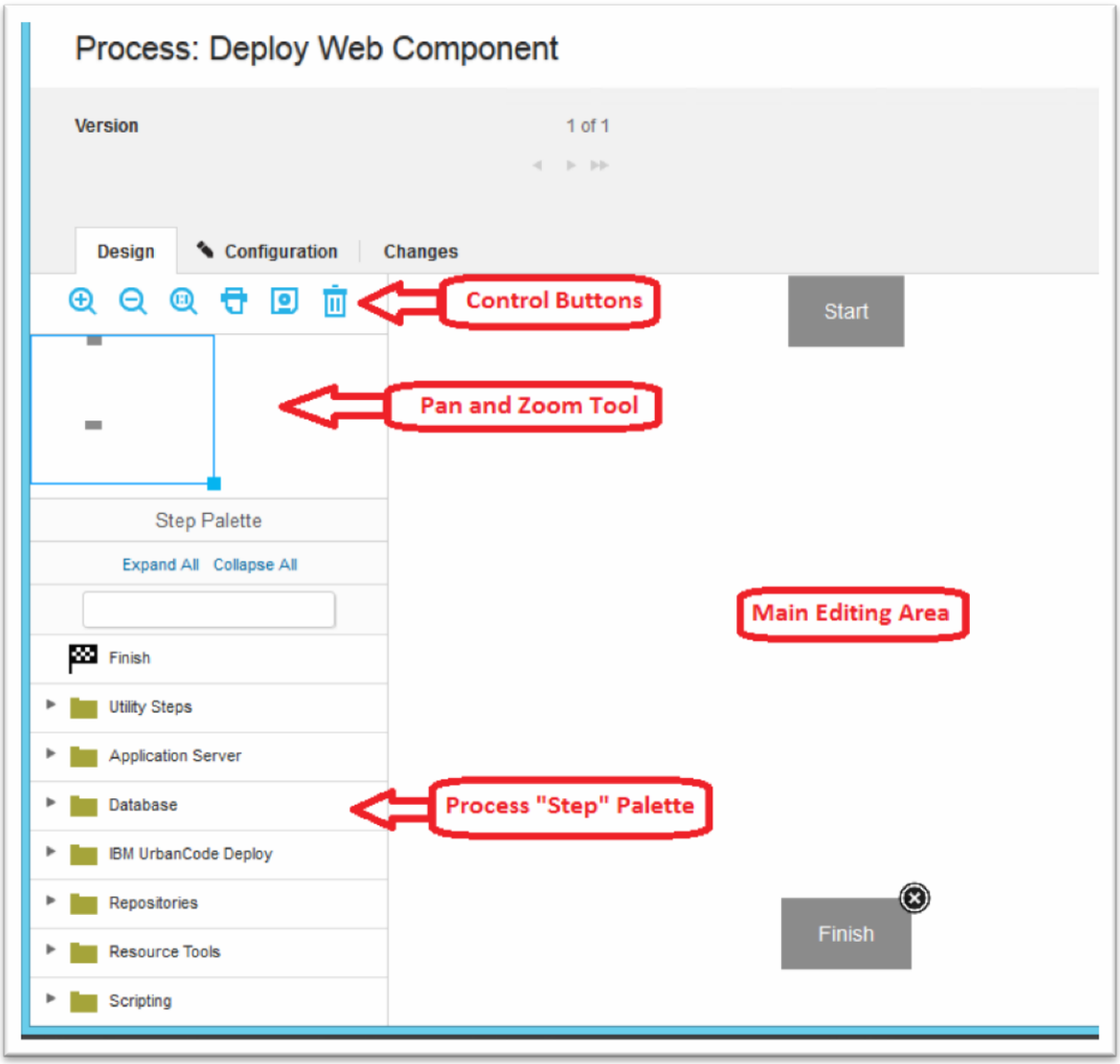
Most objects in UrbanCode Deploy can have properties associated with them – Components, Applications, Environments, Resources, etc.

(During the construction process, the UrbanCode operator can see the values expanded out. This is helpful for debugging processes, and also for audit trails to see exactly what is happening on a target server.)

Click the **Save** button to complete the **Create Process** action.

Click on the name of the new **Deploy Web Component** process. This will open the **UrbanCode Deploy Process**

Designer (a graphical editor that allows us to specify the steps and flow of execution for a process).



This editor allows us to drag-and-drop process “Steps” from the Palette (on the left) into the main Editing Area (on the right). Many steps are included by default with the base installation of UrbanCode Deploy. Others are included by loading the Plugins (which we did in earlier chapters). The DBUpgrader and Tomcat plugins have provided Steps in this Designer that we will use in the next section to describe what happens when this process is executed. (REMEMBER – the process will be executed by the UrbanCode Agent on the machine where the Agent is installed, not by the UrbanCode Server!)

6.1.2 **Define the Process Steps in the Process Designer**

Let’s take a moment to describe the big picture of what we’ll accomplish.

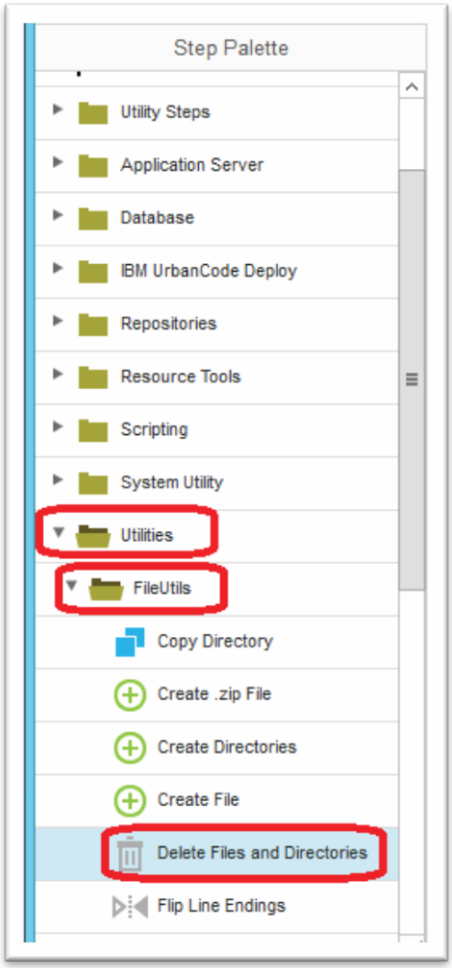
This component (JPetStore-Web) contains static web content that is intended to be deployed to a Tomcat server. This process will be responsible for the actions to make that happen. We will build steps to:

- Clean up the Agent's working directory (delete any files that might have been left by previous actions – this is a best practice we will employ in all deployment-related processes)
- Download new versions of the artifacts to the Agent machine
- Remove old content from the Tomcat server
- Deploy new content to the Tomcat server

The steps we put in this process will reflect these actions.

In the **Process Designer**, scroll down until you see the **Utilities** folder (NOTE – not the Utility Steps folder, but the Utilities folder lower in the Palette).

Expand the **Utilities** folder, then the **FileUtils** folder and look for the step named **Delete Files and Directories**.



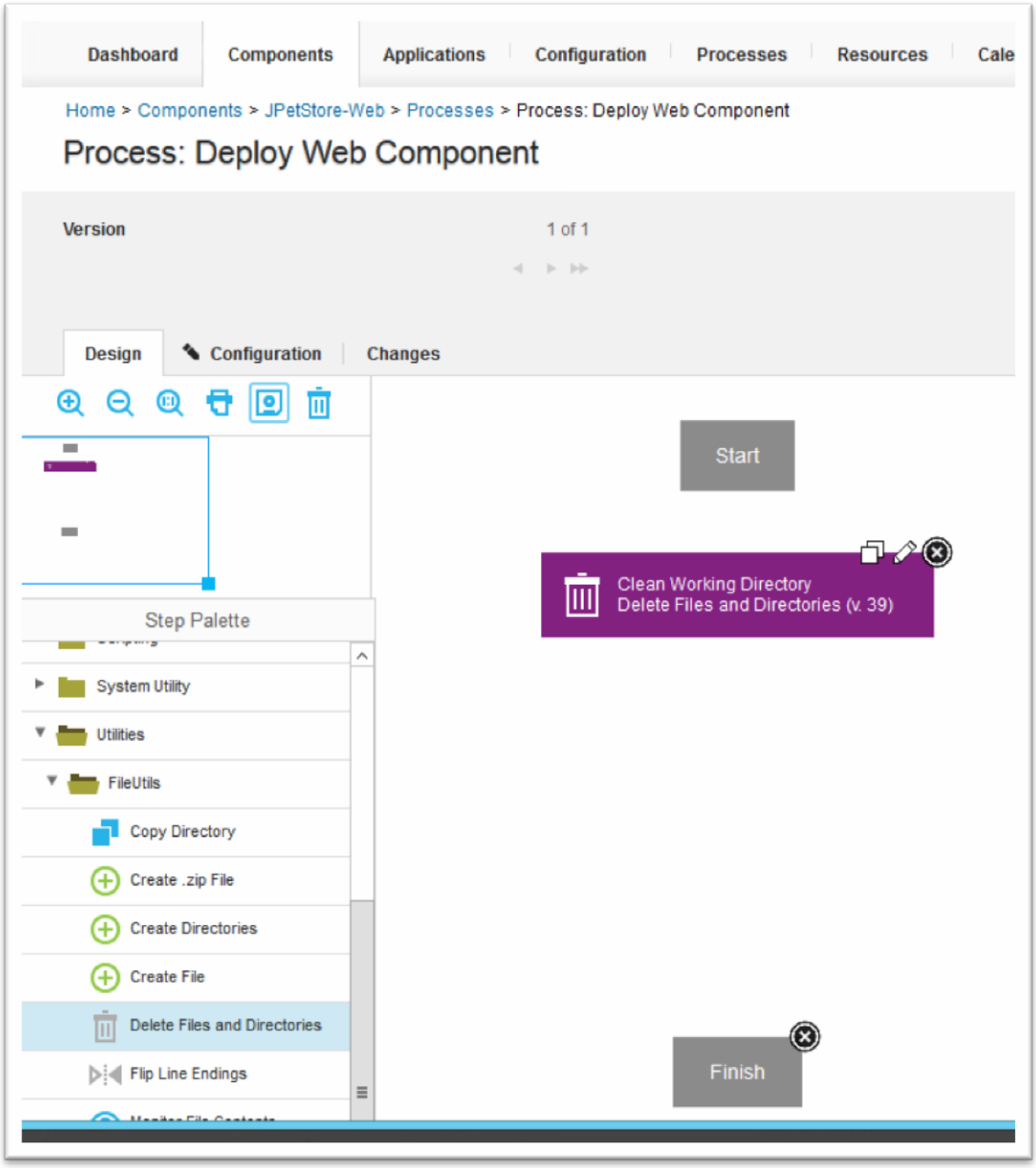
Drag the **Delete Files and Directories** step from the **Palette** into the **Main Editing Area** and drop it just below the **Start** block. This will produce the **Edit Properties** dialog.

Set the properties as shown below:

Property	Value
Name	`Clean Working Directory`
Base Directory	`.`
Include	`**/*`

Click the **OK** button.

Your screen should look something like this:



Now we'll indicate the direction of process flow.

Hover the mouse cursor over the **Start** step and you'll notice a small arrow in a circle appear in the center. Move the mouse over this arrow and the cursor changes to a hand. Click and drag the arrow from the **Start** block to the `Clean Working Directory` block we just created and release the mouse button. This will create an arrow from the **Start** block to the `Clean Working Directory` step, which indicates the direction of process flow.

The start of your process should look like this now:

Next we'll create a step to download the files associated with this component.

In the **Palette**, look for the folder named **Repositories** and expand it. Then expand the **Artifact** and the **IBM UrbanCode Deploy** folders. Locate the step named **Download Artifacts** and drag it into the editing area just under the `Clean Working Directory` step.

The default values should be adequate – no changes are needed to the properties. The key values should be set to the following:

Property	Value
Name	`Download Artifacts`
Directory Offset	`.`
Includes	`**/*`

Click the **OK** button (you may need to scroll down in the dialog to see it).

Draw a Flow Arrow from Clean Working Directory to the new Download Artifacts step we just created.

Now add another step to delete the current content in the Tomcat folder. Expand the **Utilities** folder, then the **FileUtils** folder and look for the step named **Delete Files and Directories**.

Drag and drop the Delete Files and Directories step into the Main Editing Area just below the **Download Artifacts** step. Set the properties as shown below:

Property	Value
Name	`Remove Old Content`
Base Directory	`${p:environment/tomcat.home}\webapps\JPetStore\images`
Include	`*`

*Note our use of UrbanCode Deploy properties in the **Base Directory** value. The property*

`${p:environment/tomcat.home}`

represents the install location of Tomcat, and could be different in each environment we address. This property lets us use this same Component and Procedure to install the component no matter where Tomcat has been installed.

For our lab we have created two environments: `SIT` and `UAT`. So we will set the value of the 'tomcat.home' property differently in each environment. (This is done a bit later in the lab.)

Click the **OK** button.

Draw a Flow Arrow from Download Artifacts to the new Remove Old Content step we just created.

One more step to add. In the **Palette** open the folders **Utilities** and **FileUtils** and locate the step named **Move Directory**. Drag and drop this step into the **Main Editing Area just under Remove Old Content**, but above the **Finish** block**.** (NOTE – you may have to drag the **Finish** block down in the diagram to make room.)

Set the properties as shown below:

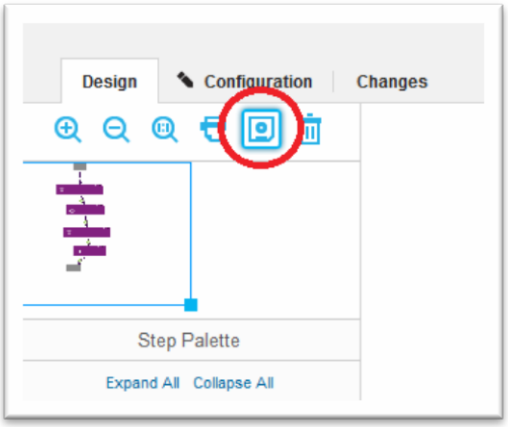
Property	Value
Name	`Deploy New Content`
Source Directory	`.\images`
Destination Directory	`${p:environment/tomcat.home}\webapps\JPetStore\images`
Include Files	`**/*`

Click the **OK** button.

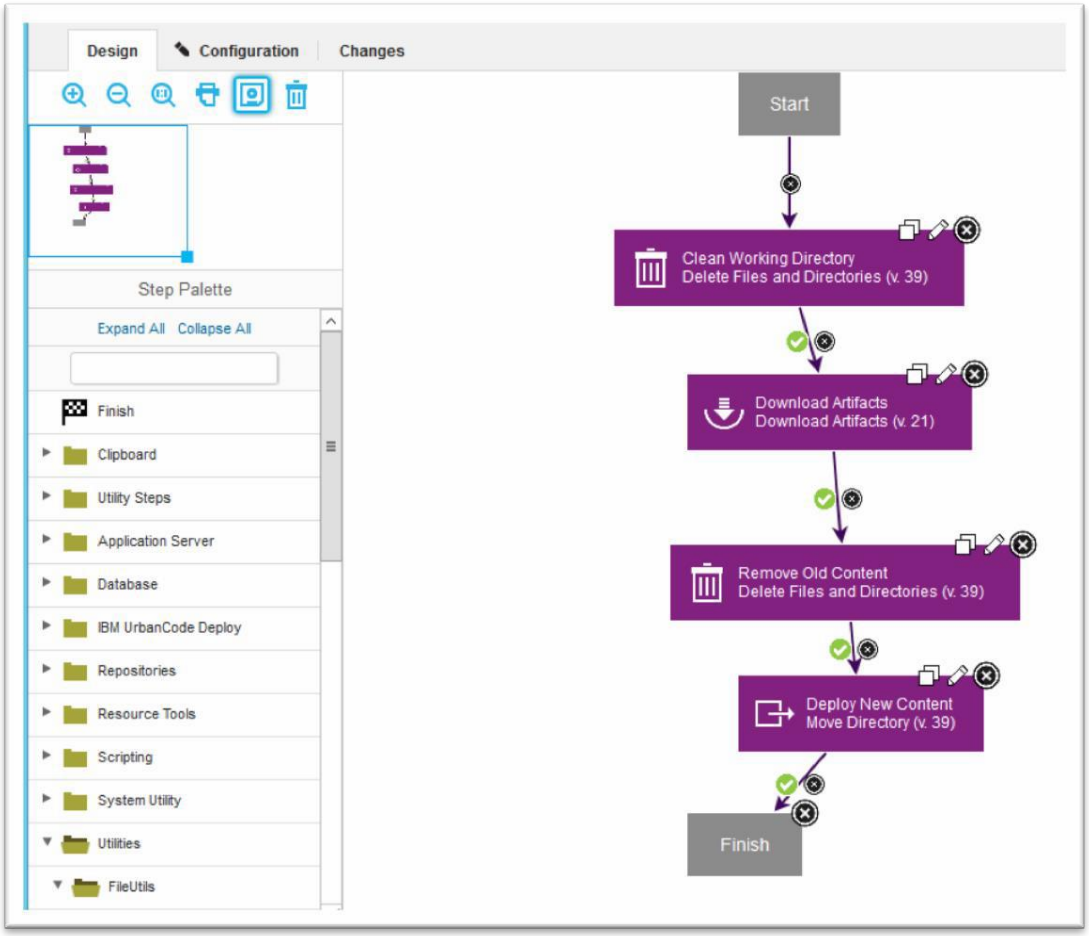
Draw a Flow Arrow from Remove Old Content to the new Deploy New Content step we just created.

Now draw a Flow Arrow from `Deploy New Content` to the Finish block.

At the top of the **Palette** click the icon to **Save** the work you've done in the **Process Designer**. (It looks like a small blue floppy disk.)



Your screen should now look something like this:



6.2 Deployment Process for JPetStore-JEE component

Once again let's take a moment to review the intentions for this process.

The UrbanCode JEE component holds the .WAR file for the JPetStore application. The .WAR file contains information in a property file that must be modified to match the computer where it is installed. Specifically, the WAR file contains

the database connection information that will be different for SIT and UAT. We will use UrbanCode Deploy properties to modify that property file at deployment time.

Also, .WAR files need to be "Deployed" to the Tomcat server using API calls. The UrbanCode Deploy plugin for Tomcat makes this easy by including features to "Deploy" and "Undeploy" as process steps.

A Java .WAR file is compressed using ZIP technology. To modify configuration files, expand (unzip) the WAR file, update the appropriate files, and then compress (re-zip) the WAR file.

So the outline for this process will be to:

- Clean up the Agent's working directory
- Download new versions of the artifacts to the Agent's machine
- Extract the contents of the .WAR file so we can edit the properties file
- Edit the properties file to reflect the current Environment
- Re-zip the contents with the updated property file back to a .WAR file
- Signal the Tomcat server to start up
- Remove (un-deploy) the previous version of the .WAR file
- Deploy the new version of the .WAR file

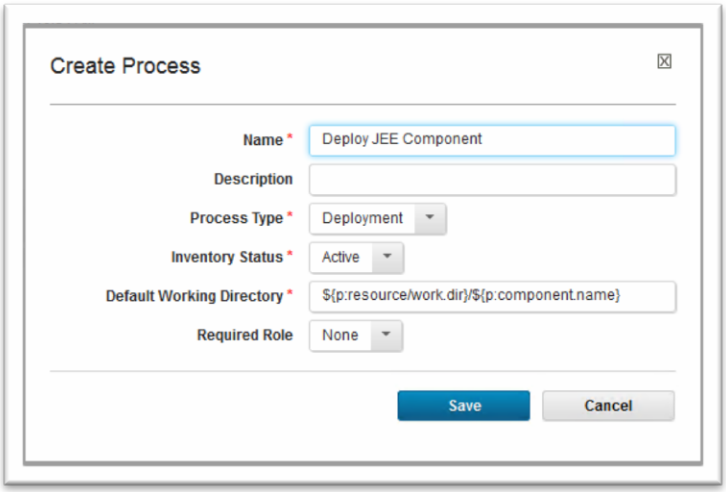
The steps we use in this process will reflect these actions.

6.2.1 Create the new Process – Deploy JEE Component

In the top row of tabs click the Components tab. In the resulting list, click **`JPetStore-JEE`**.
In the lower row of tabs, click the Processes tab. (NOTE – again, take care to pick the tab in the lower row.)

Click the **Create Process** button.

Name the new process **`Deploy JEE Component`**.
Make sure the **Process Type** field is set to **`Deployment`**.



Click the **OK** button.

6.2.2 Define the Process Steps in the Process Designer

In the previous section we navigated to process steps by opening folders and scrolling to find the step. But the Palette provides us a Search Bar to make the process easier – *if* we know the name of the step we're looking for. We'll use that feature to help create the next process.

Click on the name of the new `Deploy JEE Component` process. This will open the **UrbanCode Deploy Process Designer** (a graphical editor that allows us to specify the steps and flow of execution for a process).

In the **Palette Search Bar** type the words `delete files` and type the **Return** key. Notice that the list of folders and steps is reduced to just the one we're looking for… **Delete Files and Directories**! This can be a very handy feature once you get to know the steps. You won't have to remember which folder houses the step you are looking for.

Drag the **Delete Files and Directories** step from the **Palette** into the **Main Editing Area** and drop it just below the **Start** block**.** This will produce the **Edit Properties** dialog.

Set the properties as shown below:

Property	Value
Name	`Clean Working Directory`
Base Directory	`.`
Include	`**/*`

Click the **OK** button.

Draw a **Flow Arrow** from the **Start** block to the new `Clean Working Directory` step we just created.

Use the **Search Bar** again to locate the **Download Artifacts** step. Drag and drop one of these steps into the **Process Designer** below the `Clean Working Directory` step. The default values of all the properties should be sufficient.

Click the **OK** button.

Draw a Flow Arrow from Clean Working Directory to the new Download Artifacts step we just created.

Use the **Search Bar** again to locate the **Unzip** step. Drag and drop one of these steps into the **Process Designer** below the `Download Artifact` step. Set the properties as shown below:

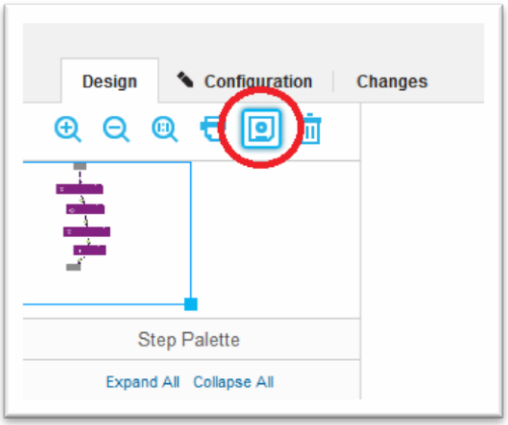
Property	Value
Name	`Expand WAR Contents`
Extract directory	`.\JPetStore_expanded`
.zip files	`JPetStore.war`

Include Files	`**/*`

Click the **OK** button.

Draw a Flow Arrow from Download Artifacts to the new Expand WAR Contents step we just created.

Click the **OK** icon in the Palette to preserve your process to this point.



Use the **Search Bar** again to locate the **Update Java Properties File** step. Drag and drop one of these steps into the **Process Designer** below the `Download Artifacts` step. Set the properties as shown below:

Property	Value
Name	`Update Property File`
Directory Offset	`.\JPetStore_expanded\WEB-INF\classes\properties`
File Includes	`database.properties`
Add/Update properties	`url=${p:environment/db.url}`

Notice again that the value of the **Add/Update properties** field uses an UrbanCode property… we will set this property for each environment later in the lab.

Click the **OK** button.

Draw a Flow Arrow from Expand WAR Contents to the new Update Property File step we just created.

Use the **Search Bar** again to locate the **Create .zip File** step. Drag and drop one of these steps into the **Process Designer** below the `Update Property File` step. Set the properties as shown below:

Property	Value
Name	`Update WAR File`
.zip File Name	`JPetStore.war`
Base Directory	`.\JPetStore_expanded`
Include	`**/*`
Update Existing	`<checked>`

Click the **OK** button.

Draw a Flow Arrow from Update Property File to the new Update WAR File step we just created.

Use the **Search Bar** again to locate the **Start Tomcat** step. Drag and drop one of these steps into the **Process Designer** below the `Update WAR File` step. Set the properties as shown below:

Property	Value
Name	`Start Tomcat`
Launcher	`${p:environment/tomcat.start}`
Startup timeout	`30`
Port	`${p:environment/tomcat.port}`
Catalina Home	`${p:environment/tomcat.home}`

Click the **OK** button.

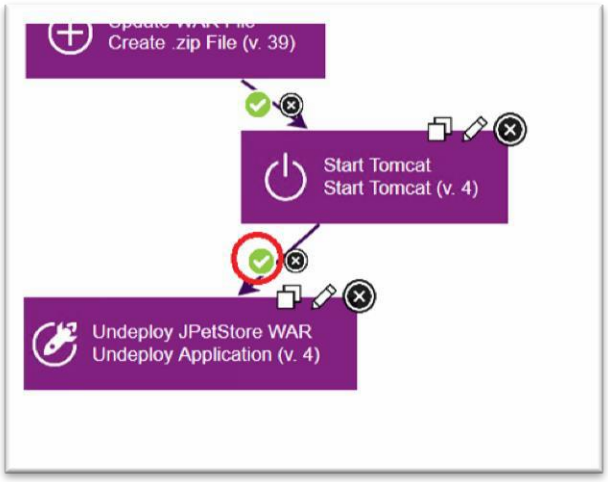
Draw a **Flow Arrow** from `Update WAR File` to the new `Start Tomcat` step we just created.

Use the **Search Bar** again to locate the **Undeploy Application** step. The one we want will be found in the **Tomcat** folder. Drag and drop one of these steps into the **Process Designer** below the `Start Tomcat` step. Set the properties as shown below:

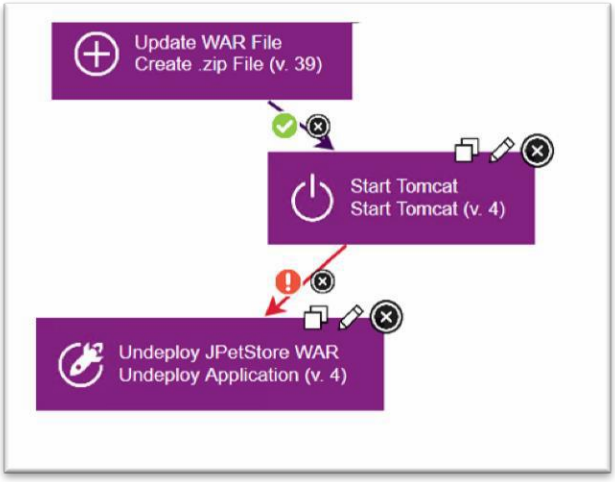
Property	Value
Name	`Undeploy JPetStore WAR`
Tomcat Manager URL	`${p:environment/tomcat.url}:${p:environment/tomcat.port}/manager/text`
Tomcat Manager Username	`demo`
Tomcat Manager Password	`demo`
Context Name	`/JPetStore`

Click the **OK** button.

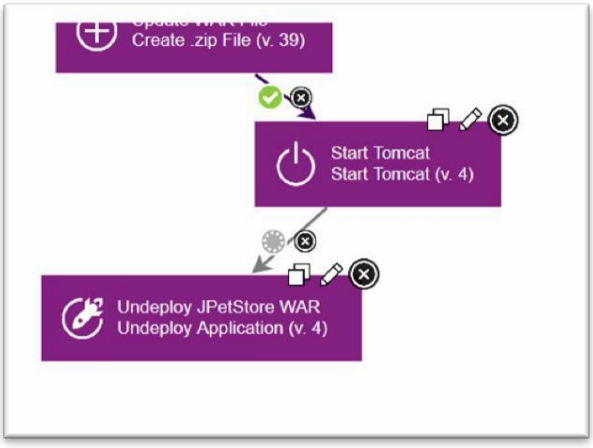
Draw a **Flow Arrow** from `Start Tomcat` to the new `Undeploy JPetStore WAR` step we just created. Now click on the green checkmark that appears on this last flow arrow:



The green check changes to a red exclamation point.



Click the red exclamation point and it changes to a grey circle.



	*Each of these states indicates some error control result in our process:* • *Green Check = continue only if step succeeds* • *Red Exclamation = continue only if step fails* • *Grey Circle = continue in all cases (failure or success)* *In our process, we've changed the state to a Grey Circle. Why? If the Tomcat server is already started, then the **Start Tomcat** step will fail with an error code. Because this situation is irrelevant to us (since all we want is the server to be started) we tell our process to continue whether or not the **Start Tomcat** step fails.* *We could also use this feature to indicate different paths based on whether a step fails or succeeds. (We can draw more than one arrow out of a step, and use the designations of green check or red exclamation to control process flow down two different paths.)* *There are also more complex steps for controlling flow logic (including a **Switch** step that allows us to traverse one of several paths based on some value) but that is subject for another lab.*

Use the **Search Bar** again to locate the **Deploy Application** step. The one we want will be found in the **Tomcat** folder.

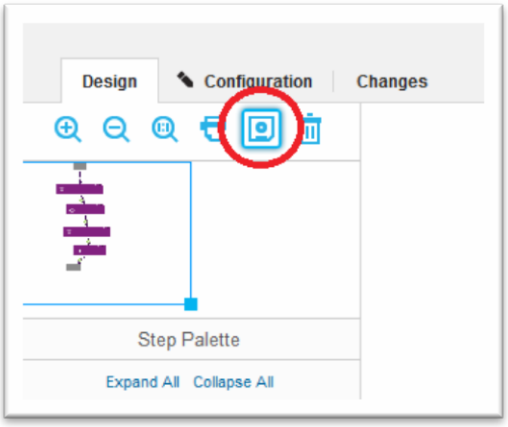
Drag and drop one of these steps into the **Process Designer** below the `Undeploy JPetStore WAR` step. Set the properties as shown below:

Property	**Value**
Name	`Deploy JPetStore WAR`
Tomcat Manager URL	`${p:environment/tomcat.url}:${p:environment/tomcat.port}/manager/text`
Tomcat Manager Username	`demo`
Tomcat Manager Password	`demo`
Context Name	`/JPetStore`
War File Path	`.\JPetStore.war`

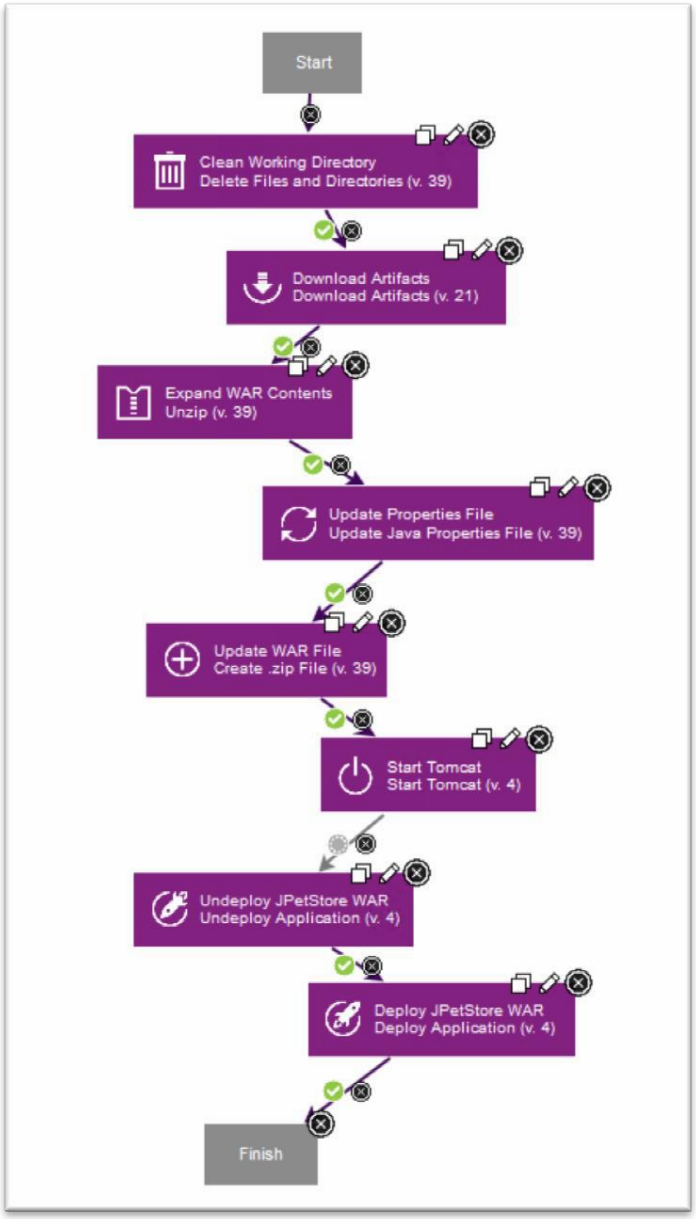
Click the **OK** button.

Draw a **Flow Arrow** from `Undeploy JPetStore WAR` to the new `Deploy JPetStore WAR` step we just created… then from the `Deploy JPetStore WAR` step to the **Finish** block.

Click the blue **Save** button on the **Palette** to save this process.



Your process diagram should now look something like this:


Start
Clean Working Directory
Delete Files and Directories (v. 39)
Download Artifacts
Download Artifacts (v. 21)
Expand WAR Contents
Unzip (v. 39)
Update Properties File
Update Java Properties File (v. 39)
Update WAR File
Create .zip File (v. 39)
Start Tomcat
Start Tomcat (v. 4)
Undeploy JPetStore WAR
Undeploy Application (v. 4)
Deploy JPetStore WAR
Deploy Application (v. 4)
Finish

7 UrbanCode Deploy Properties

7.1 Overview

As we mentioned earlier, UrbanCode Deploy allows us to use properties to improve the flexibility of our processes. Properties can be attached to several kinds of UrbanCode Deploy objects, including Environments (like our **SIT** and **UAT** environments). While building our processes we have been referencing properties in the fields of our steps. Many of those properties were "custom" and we must provide a value for them in the right context.

We can use UrbanCode Deploy properties to improve the flexibility and reusability of the elements we create. For instance, a property set on an Environment can be used to hold a value that is particular to that Environment (like the url for a database server in QA, which will almost certainly be a different value in the production environment).

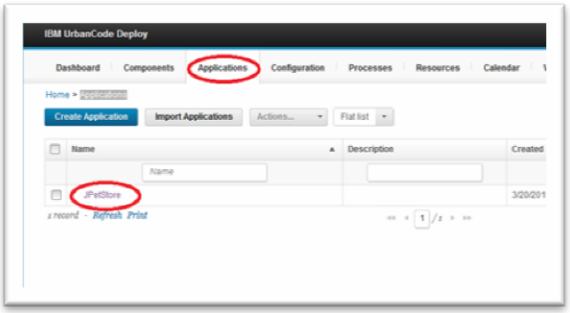
To set a property value in the context of an Environment, we must add that property (and its value) to each of the UrbanCode Deploy Environment objects.

Several deployment properties have been referenced as shown below. You need to define values for these properties.

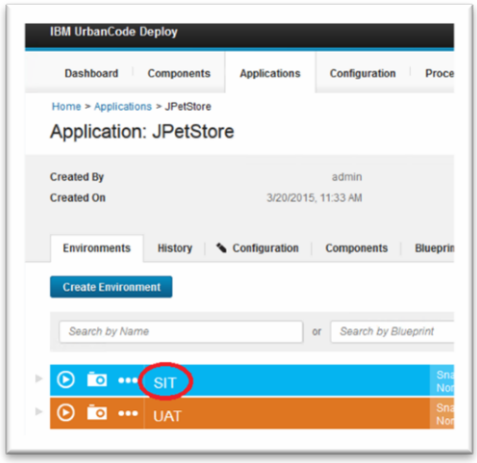
In this section we'll define the properties and their values for our two Environments.

7.1.1 Define SIT Environment Properties

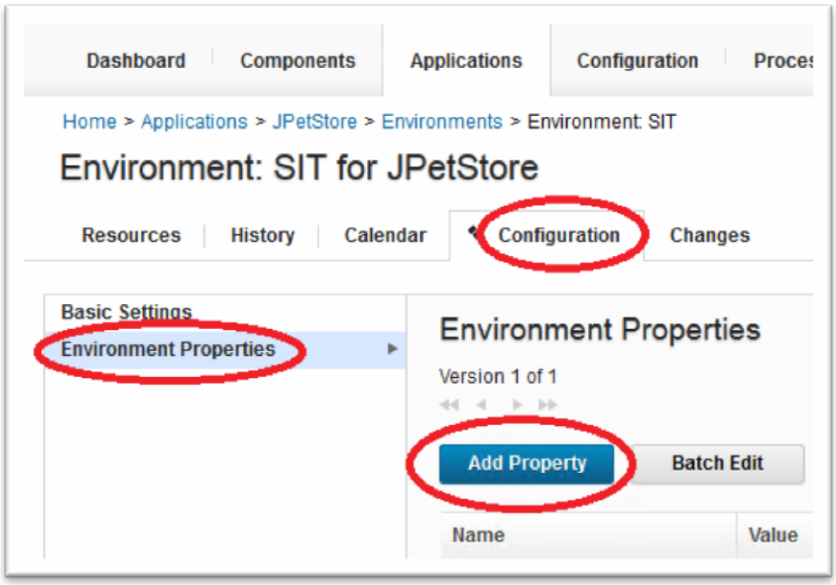
Click the **Applications** tab in the top row of tabs. Click on the link for the **JPetStore** application.



Click on the **SIT** environment (click the text for **SIT** as shown below).



In the lower row of tabs, click on the **Configuration** tab then select **Environment Properties** from the left hand side of the screen. Then click the **Add Property** button on the right hand side of the screen.



	Note the text referring to the 'Version' (in the picture above it refers to version 1 of 1). Each time you make a change to the properties (and other objects in UrbanCode Deploy) a new version is created. You can review previous versions, and re-instate them as the new 'current' version by using the 'forward' and 'reverse' icons below the text for versions (small sets of arrows pointing left and right). *This illustrates the audit trail maintained by UrbanCode Deploy. All actions and changes are tracked, and can be reviewed later.*

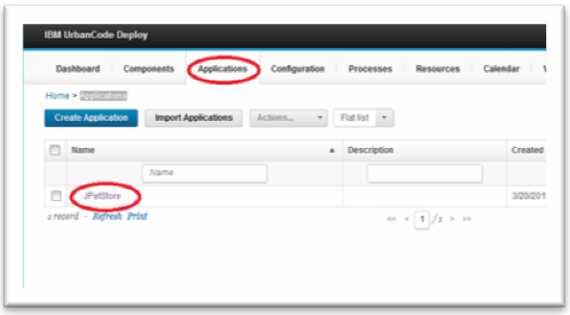
Add the following properties:

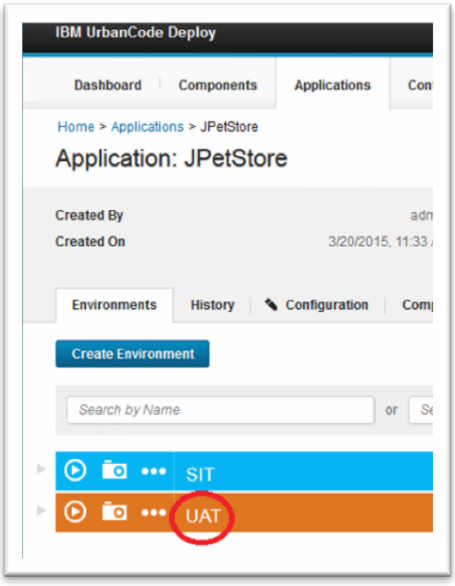
Property	Value
db.url	`jdbc:mysql://ucd.sit-server.com:3306/jpetstore_sit`
tomcat.home	`C:\Env_Web_Servers\SIT\apache-tomcat-xxxx`
tomcat.port	`8085`
tomcat.start	`C:\Env_Web_Servers\SIT\apache-tomcat-xxxx\bin\startup.bat`
tomcat.url	`http://ucd.sit-server.com`

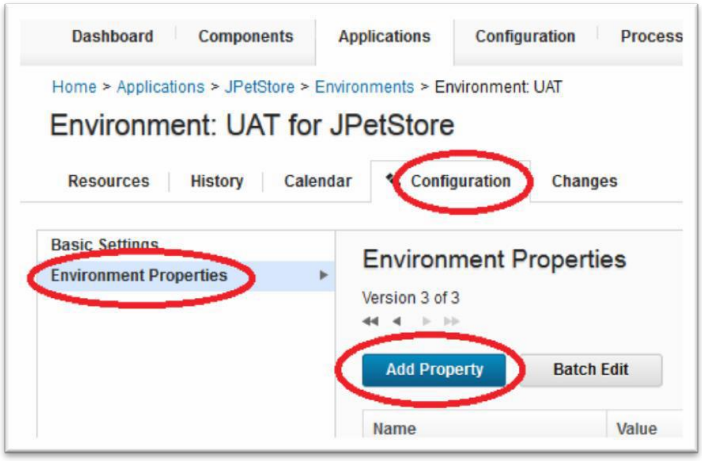
	*The reference to "xxxx" in the paths above represents the version of Tomcat that you installed. Substitute the correct reference for the version number you actually downloaded to make the path correct.*

7.1.2 Define UAT Environment Properties

Repeat the process above to add properties to the **`UAT`** environment.







Add the following properties:

Property	Value
db.url	`jdbc:mysql://ucd.uat-db-server.com:3306/jpetstore_uat`
tomcat.home	`C:\Env_Web_Servers\UAT\apache-tomcat-xxxx`
tomcat.port	`8086`
tomcat.start	`C:\Env_Web_Servers\UAT\apache-tomcat-xxxx\bin\startup.bat`
tomcat.url	`http://ucd.uat-app-server.com`

	*The reference to "xxxx" in the paths above represents the version of Tomcat that you installed. Substitute the correct reference for the version number you actually downloaded to make the path correct.*

8 UrbanCode Application Processes

In this section of the lab you define an **Application Process** to deploy the JEE and WAR components. Later, you will update the **Application Process** to include the DB component.

Applications consist of multiple components. Deploying an application consists of an organized process of deploying the *components* of that application.

In the prior sections, we have created two components each of which contains a process with the particulars of how to deploy that particular component. Next we will illustrate how the UrbanCode Deploy Application object can also have processes, which are used to coordinate the deployment of individual components associated with that application.

8.1 Deployment Process for the JPetStore Application

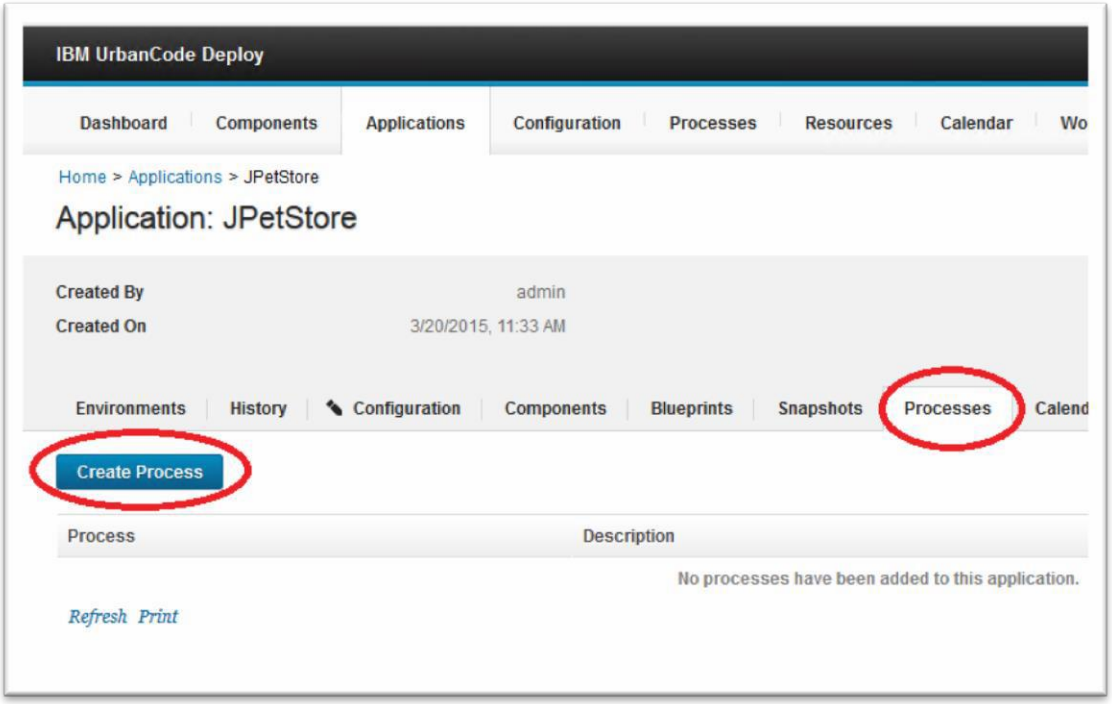
As stated earlier, each Component can contain a Process that carries with it the information needed to deploy itself. Now we will create a process on the UrbanCode Application logic to coordinate the execution of the individual component processes that make up that Application.

8.1.1 Create the New Application Process – Deploy JPetStore

From the upper row of tabs click the **Applications** tab.

Click the link for the `JPetStore` application.

From the lower row of tabs click the **Processes** tab, and then click the **Create Process** button.



Name the new process `Deploy JPetStore`. Click the **Save** button.

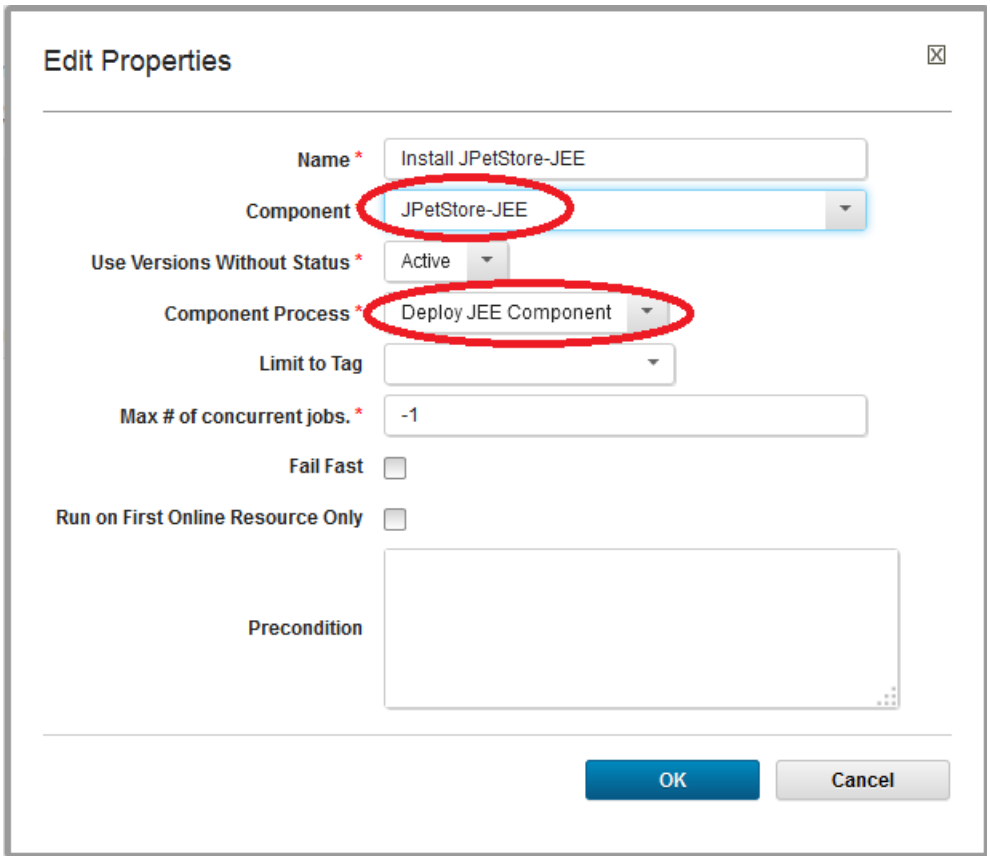
8.1.2 Define the Process Steps for the Application Process

Click on the name of the new `Deploy JPetStore` process. This will open the **UrbanCode Deploy Process Designer**.

	*You will notice a different set of **Steps** available in the **Palette**. This is because we are now editing a process attached to an application object instead of a component object like before.*

Drag the **Install Component…** step from the **Palette** into the **Main Editing Area** and drop it just below the **Start** block. This will produce the **Edit Properties** dialog.

BEFORE changing the **Name** field, set the **Component** field to `JPetStore-JEE`. You will see that the **Name** field automatically updates to reflect the name of the component we are deploying. Make sure the **Component Process** field is set to `Deploy JEE Component`.



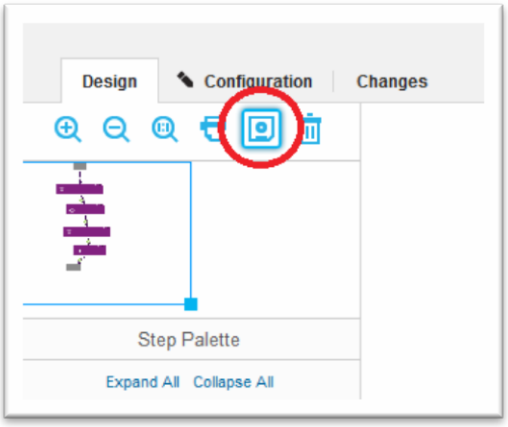
Click the **OK** button.

Draw a **Flow Arrow** from the **Start** block to the new `Install JPetStore-JEE` step we just created.

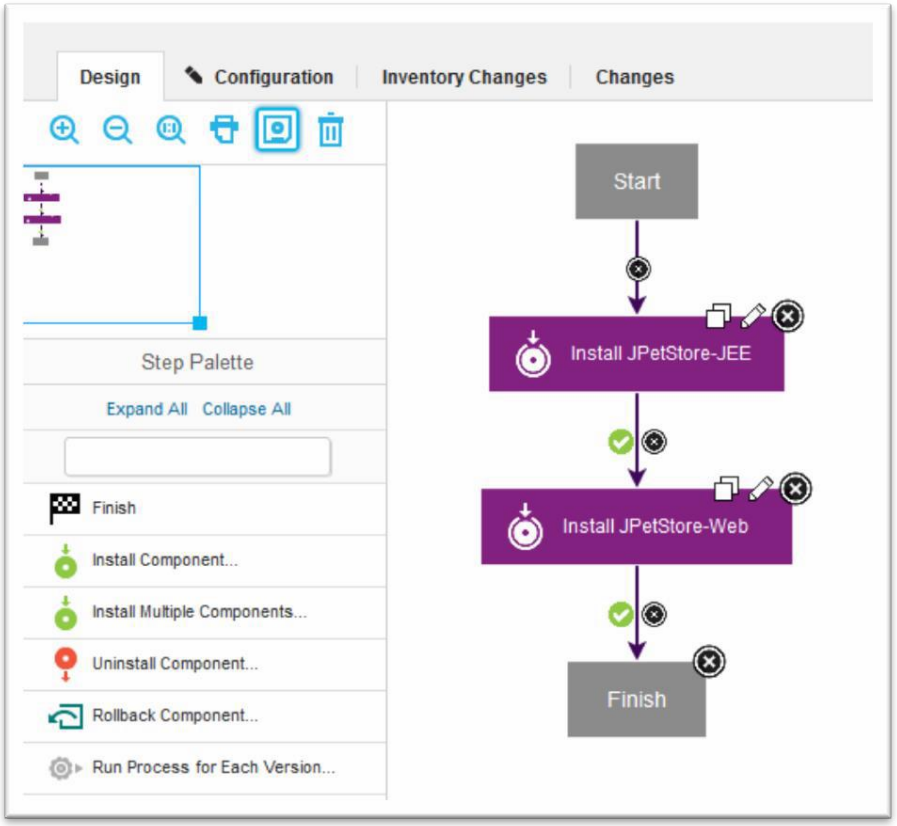
Repeat this process and add another **Install Component…** step to install the `JPetStore-Web` component, below the Install JPetStore-JEE step.

Draw a **Flow Arrow** from the `Install JPetStore-JEE` step to the `Install JPetStore-Web` step, then from `Install JPetStore-Web` to the **Finish** block.

Click the blue **Save** button on the **Palette** to save this process.



Your application process should look like this:

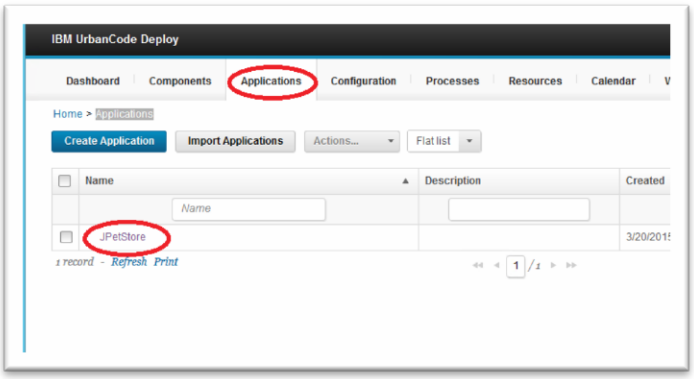


8.1.3 Test the Application Deployment Process in the SIT Environment

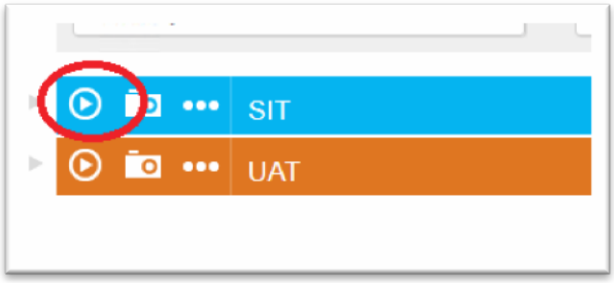
You are now ready to deploy part of the JPetStore application - the JEE and WEB components. (Without the DB component, the application won't run, but this is a good point to verify the work done so far.)

Click the **Applications** tab in the top row of tabs.

Click on the link for the **`JPetStore`** application. This will display the two environments we created earlier (**`SIT`** and **`UAT`**).

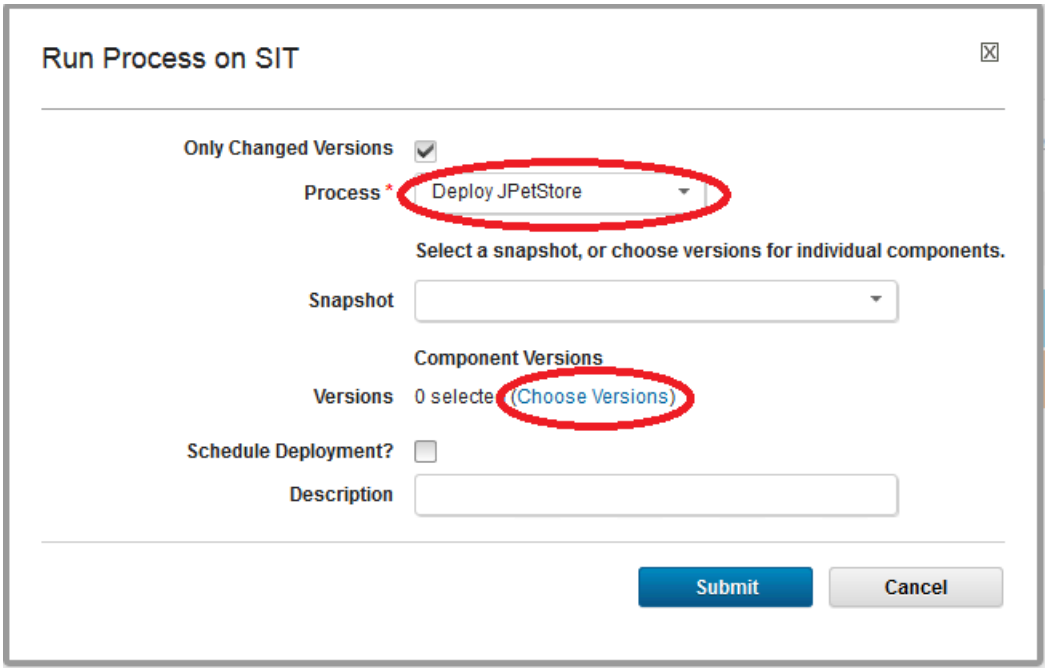


Look at the row for the **SIT** environment – on the far left you see a small icon like a “Play” button. This is the **Request Process** button. Click this button to initiate an UrbanCode Deploy process.

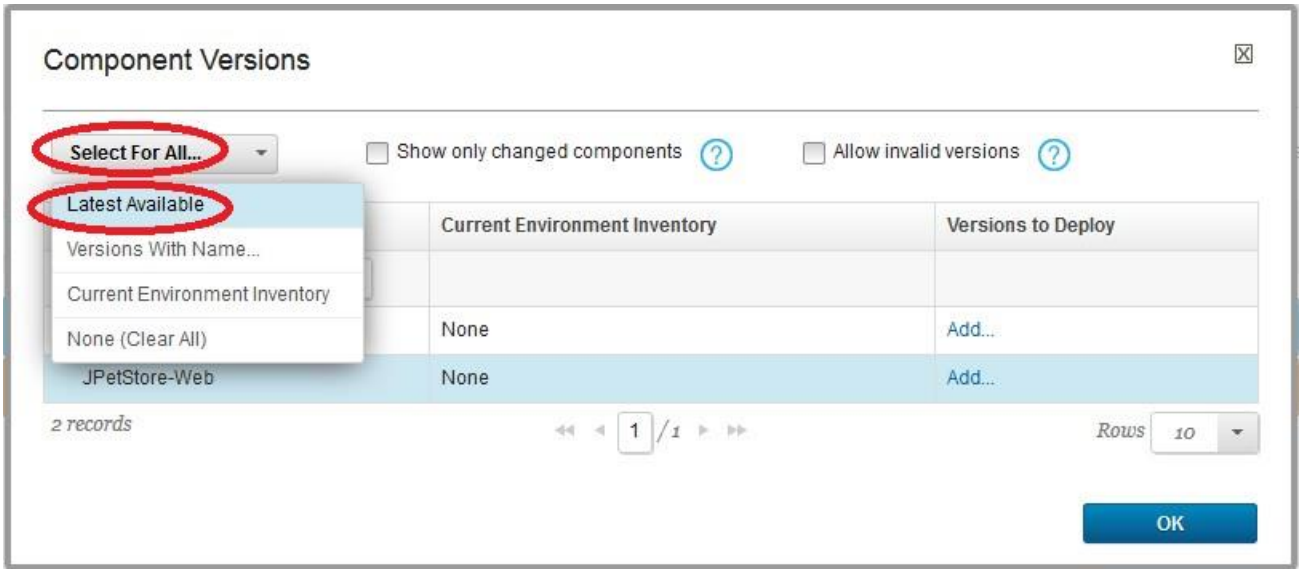


The dialog appears to request a process that will run against the **`SIT`** environment. Set the **Process** field to **`Deploy JPetStore`**.

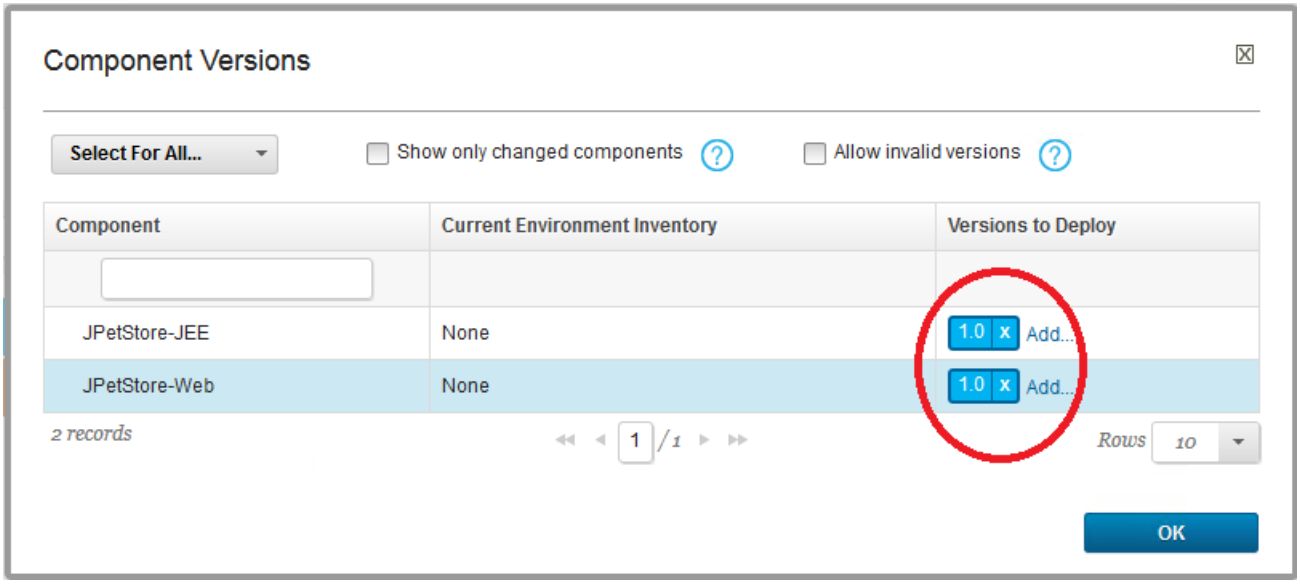
By the **Versions** field, click the link that says **Choose Versions**… another dialog appears.



In the **Component Versions** dialog, set the **Select for All…** dropdown box in the top left to `Latest Available`.



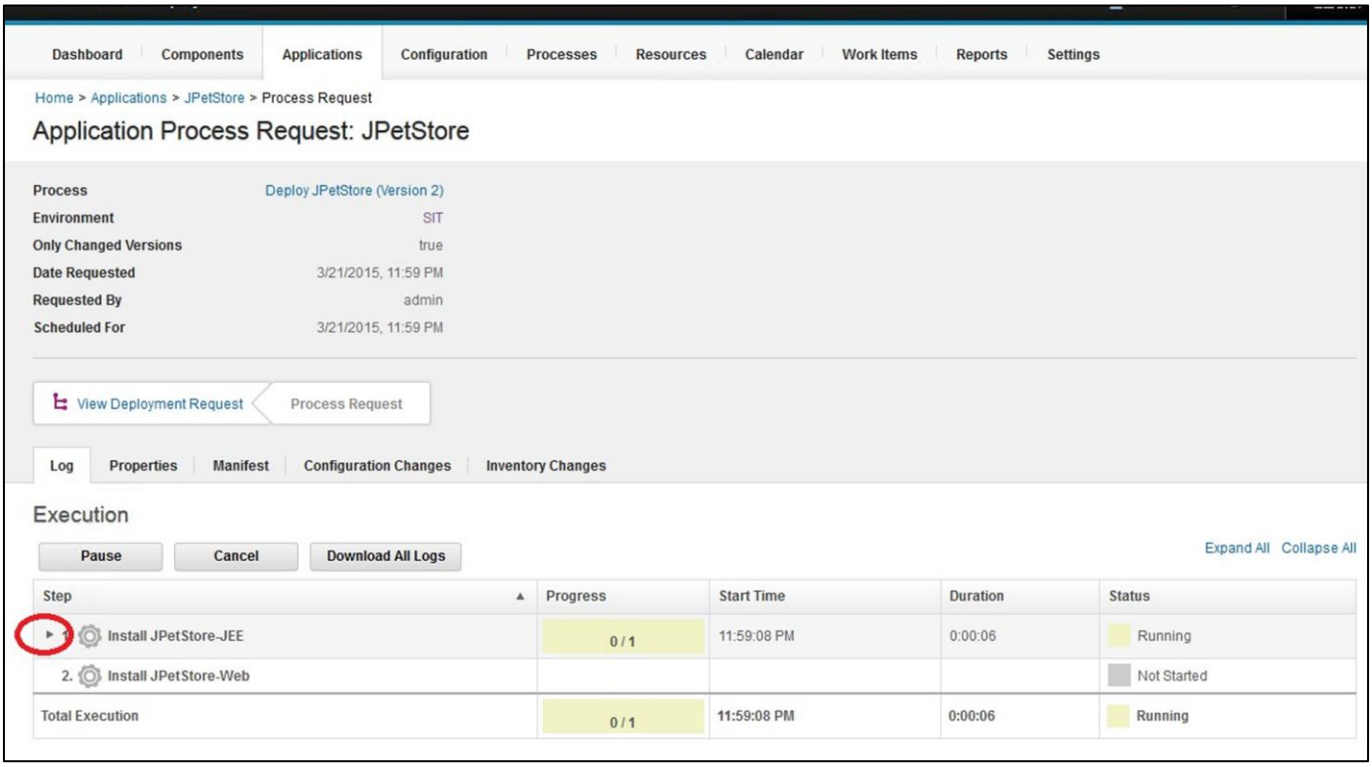
This will add the latest version of all components in the list to the **Versions to Deploy** column.



Click the **OK** button.

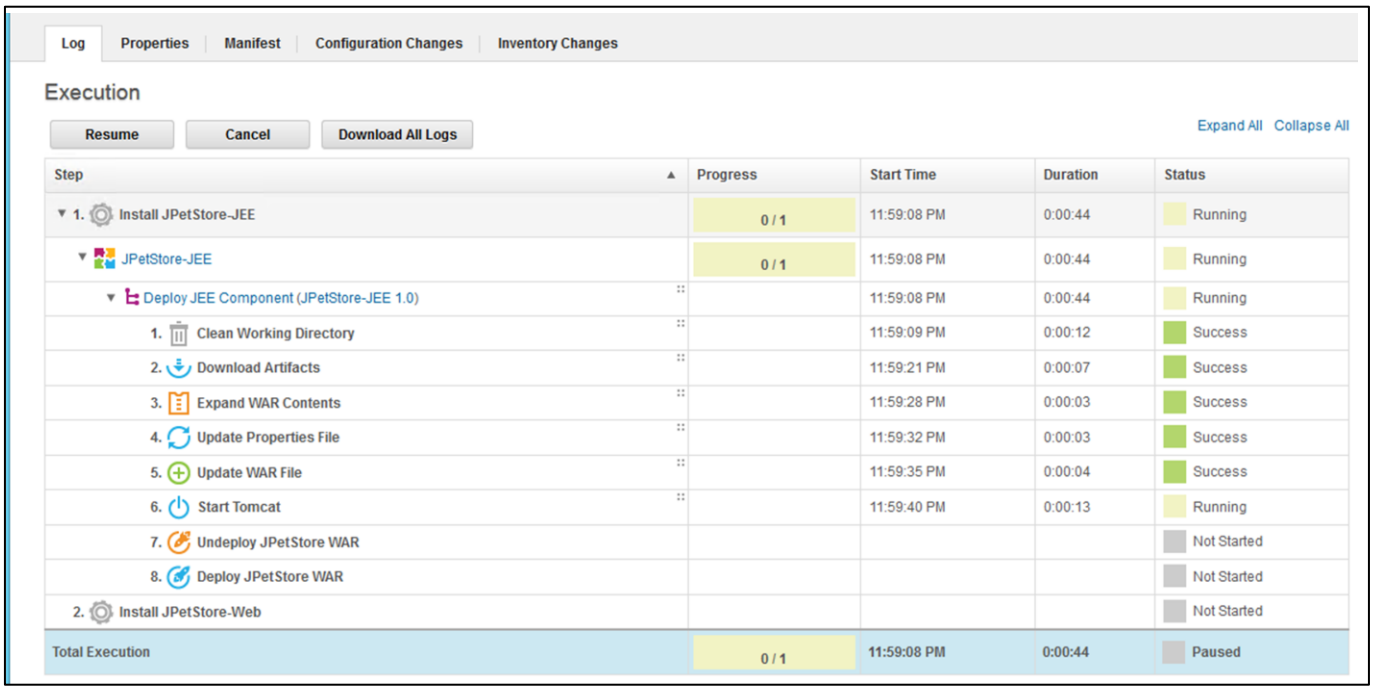
Back on the **Run Process on SIT** dialog, click the **Submit** button and UrbanCode Deploy starts the deployment process.

The system displays the current application deployment status. You can see which components aren't started, are running, were successful, or failed. Use the 'twistie' on the far left of any component to expand it and see additional information.

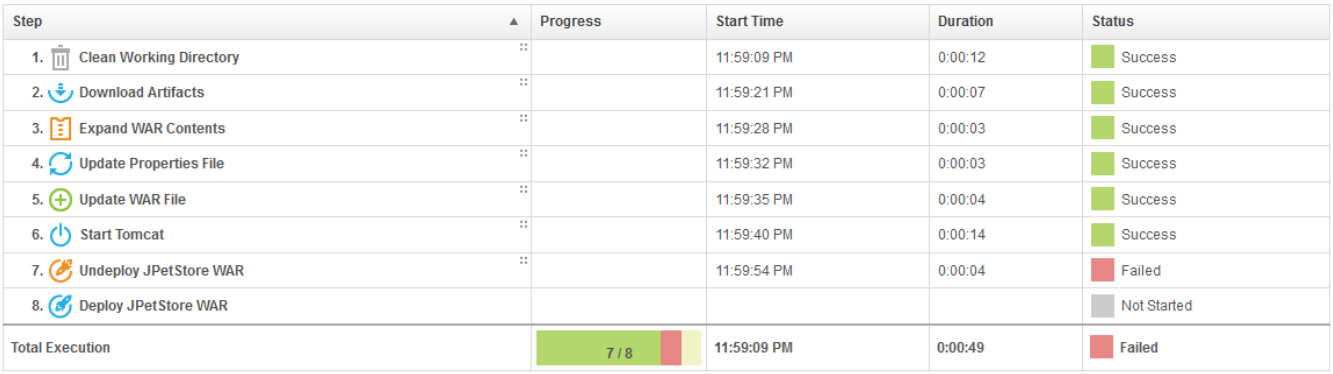


When expanded, you can see the steps that you created in the component process (on the left) and the status (on the

right) if they were successful or they failed, or if they are running now or have not yet started.



When the system finishes, you will see that the overall process has failed because one of the steps has failed. Below is the final progress report where we can see that **Step 7 Undeploy JPetStore WAR** has failed.



This makes sense because the JPetStore application has never been deployed at all. But the failure of this step has caused the overall process to terminate without trying to run the last step, **`Deploy JPetStore WAR`**.

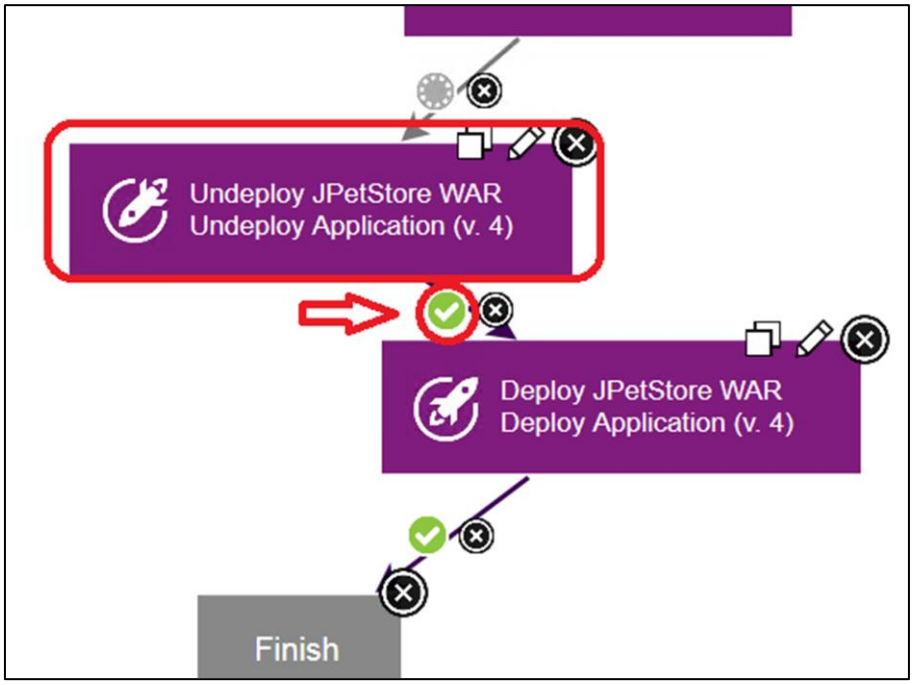
We can adjust our process to allow it to continue even if the **`Undeploy JPetStore WAR`** step fails. On the top row of tabs, click the **Components** tab.

From the list of components click the link for the **`JPetStore-JEE`** component. On the lower row of tabs, click the **Processes** tab.

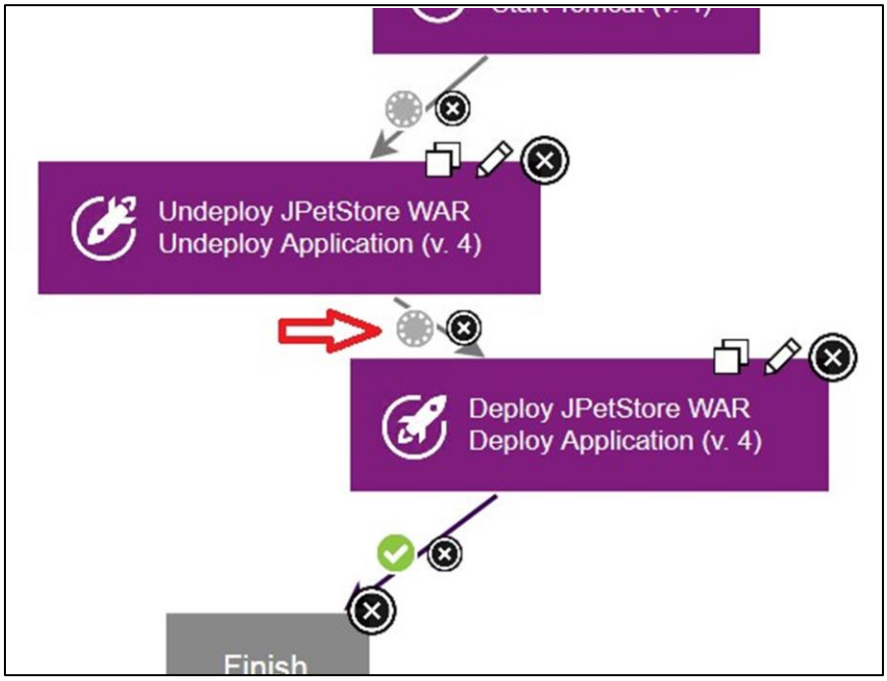
From the list of processes click the link for the **`Deploy JEE Component`** process.

In the **Process Designer**, scroll down until you can see the **`Undeploy JPetStore WAR`** step (and, more importantly,

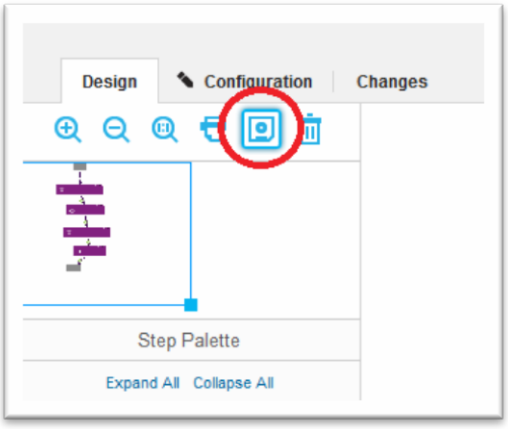
the **Flow Arrow** that leads out of it into the next step).



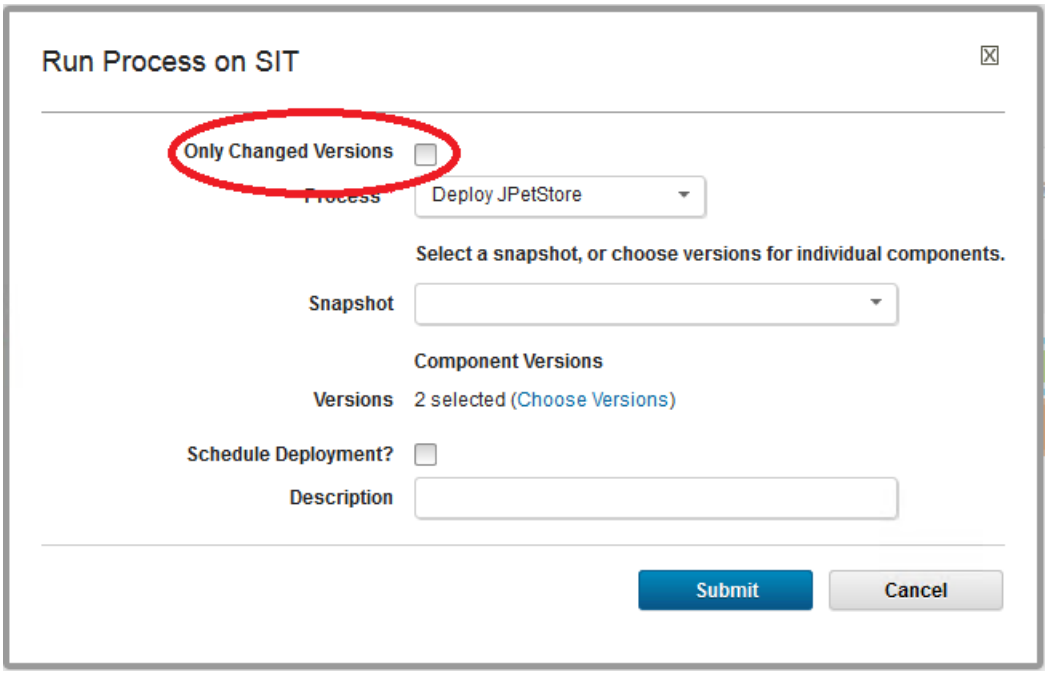
Click the green checkmark and it changes to a red exclamation point. Click it once more and it turns to a grey circle. This indicates to keep running even if the `Undeploy JPetStore WAR` step fails… and will allow us to reach the step where we actually deploy the JPetStore WAR file for the first time.



Click the blue **Save** button on the **Palette** to save this process.



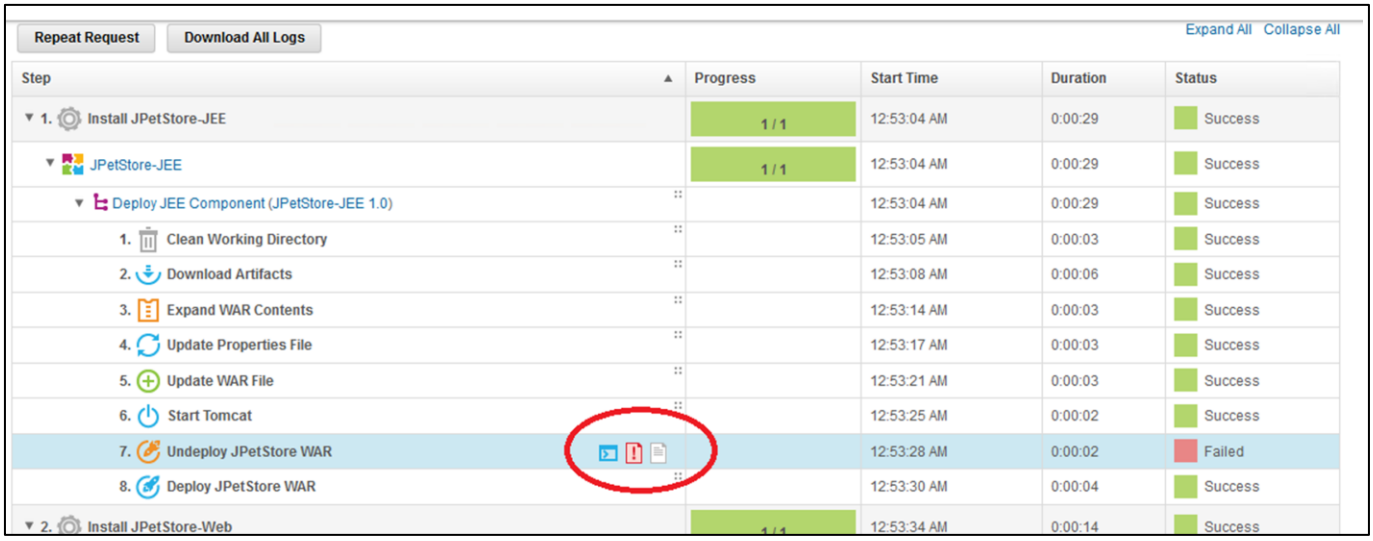
Now, rerun the Application process **`Deploy JPetStore in the SIT`** environment… BUT this time you must clear the **Only Changed Versions** check box in the **Run Process on SIT** dialog. (Review the steps at the beginning of this section if needed.)



This time, the process continues even though **`Undeploy JPetStore WAR`** fails…and the **`JPetStore-JEE`** component is successfully deployed.

Expand all the components with the 'twisties' on the left to see the results.

Hover your mouse over any of the steps and you will see three icons appear to the right of the step name.



Clicking the blue icon will show you the console output of that command step, as if you executed it from the command line of the Agent machine.

The red icon shows errors, if any. (If there were no errors, that icon will not even be present.)

The white icon shows you the expanded value of any UrbanCode Deploy properties at runtime. Click on each one to see what the output looks like.

Now that the application has been deployed once, all steps should show success in future runs.

9 Adding the Database Component

In an earlier section we created 3 Components… so far we've included two – the Web Component (**JPetStore-WAR**) and the Application Component (**JPetStore-JEE**). To have our Application actually run, however, we must now include the Database Component in our project.

9.1 Deployment Process for JPetStore-DB component

The database deployment plugin used in this workshop is not based on providing SQL script files. Instead an XML file is provided which lets you define explicit apply and roll-back capabilities.

Deploying the DB Component consists of the following steps:

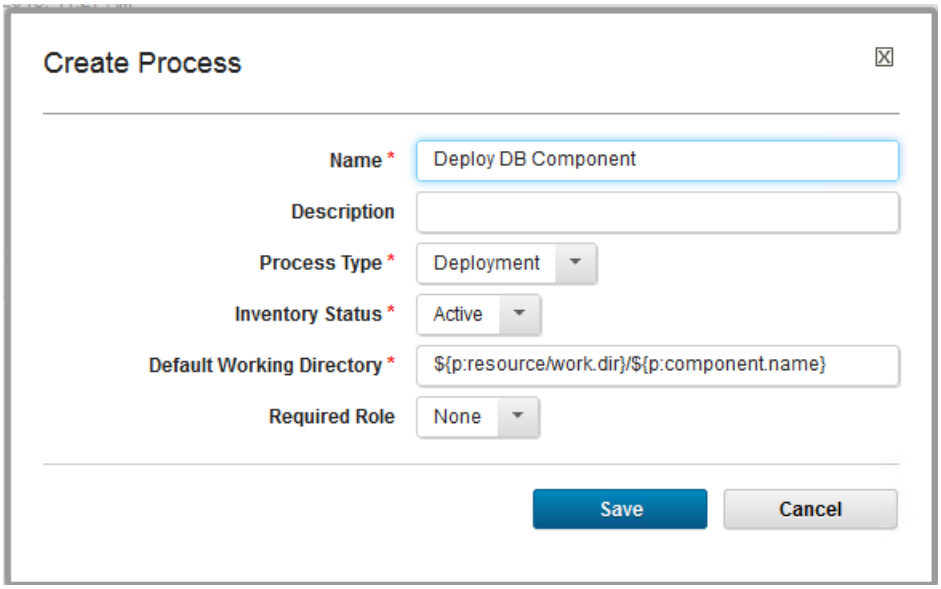
- Download the artifacts
- Upgrade the database

9.1.1 Create the new Process – Deploy DB Component

In the top row of tabs click the **Components** tab. In the resulting list, click `JPetStore-DB`.
In the **lower** row of tabs, click the **Processes** tab.

Click the **Create Process** button.

Name the new process `Deploy DB Component`.
Make sure the **Process Type** field is set to `Deployment`.



Click the **Save** button.

9.1.2 Define the Process Steps in the Process Designer

Click on the name of the new `Deploy DB Component` process. This will open the **UrbanCode Deploy Process Designer** (a graphical editor that allows us to specify the steps and flow of execution for a process).

In the **Palette Search Bar** type the words `delete files` and type the **Return** key. Notice that the list of folders and steps is reduced to just the one we're looking for… **Delete Files and Directories**! This can be a very handy feature once you get to know the steps. You won't have to remember which folder houses the step you are looking for.

Drag the **Delete Files and Directories** step from the **Palette** into the **Main Editing Area** and drop it just below the **Start** block**.** This will produce the **Edit Properties** dialog.

Set the properties as shown below:

Property	Value
Name	`Clean Working Directory`
Base Directory	`.`
Include	`**/*`

Click the **OK** button.

Draw a **Flow Arrow** from the **Start** block to the new `Clean Working Directory` step we just created.

Use the **Palette Search Bar** to locate the **Download Artifacts** step. Drag and drop one of these steps into the **Process Designer** below the `Clean Working Directory` step. The default values of all the properties should be sufficient.

Click the **OK** button.

Draw a Flow Arrow from Clean Working Directory to the new Download Artifacts step we just created.

Use the **Palette Search Bar** to locate the **Upgrade DB** step. Drag and drop one of these steps into the **Process Designer** below the `Download Artifacts` step. Set the properties as shown below:

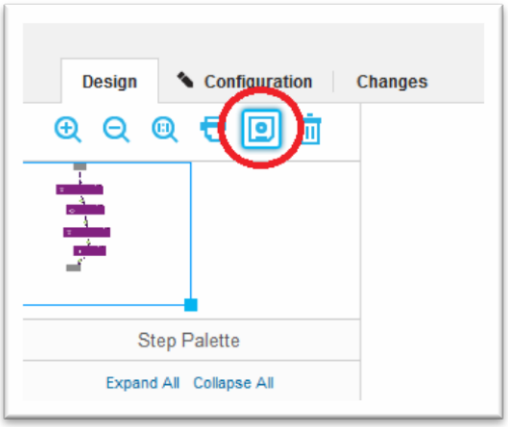
Property	Value
Name	`Upgrade DB`
Driver Classname	`com.mysql.jdbc.Driver`
DB Driver Jar	`lib\mysql-connector-java-5.1.20-bin.jar`
URL	`${p:environment/db.url}`
User	`jpetstore`
Password	`jppwd`
SQL File path	`.`
SQL File Include	`*.xml`
Current Version SQL	`SELECT VER FROM DB_VERSION WHERE RELEASE_NAME = ?`
Delete Version SQL	`DELETE FROM DB_VERSION WHERE RELEASE_NAME = ?`

Update Version SQL	`INSERT INTO DB_VERSION (RELEASE_NAME,VER) VALUES(?,?)`

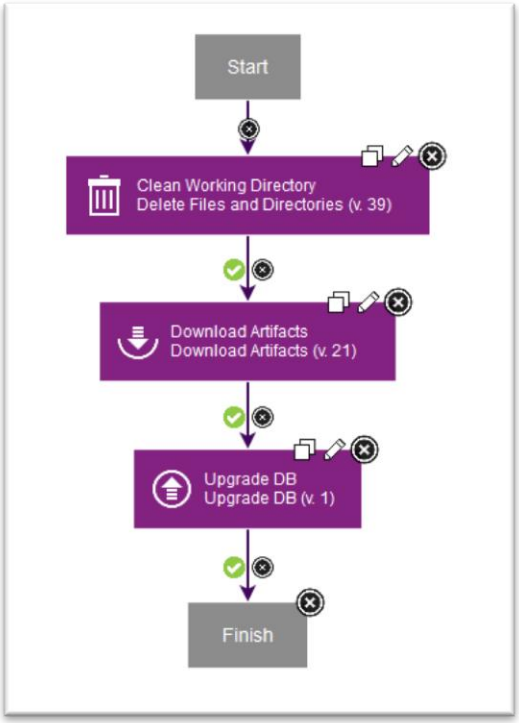
Click the **OK** button.

Draw a **Flow Arrow** from `Download Artifacts` to the new `Upgrade DB` step we just created. Now draw a **Flow Arrow** from `Upgrade DB` to the **Finish** block.

At the top of the **Palette** click the icon to **Save** the work you've done in the **Process Designer**. (It looks like a small blue floppy disk.)



Your screen should now look something like this:



9.2 Update the Application Deployment Process

9.2.1 Include the process to deploy the DB component

From the upper row of tabs click the **Applications** tab.

Click the link for the `JPetStore` application.

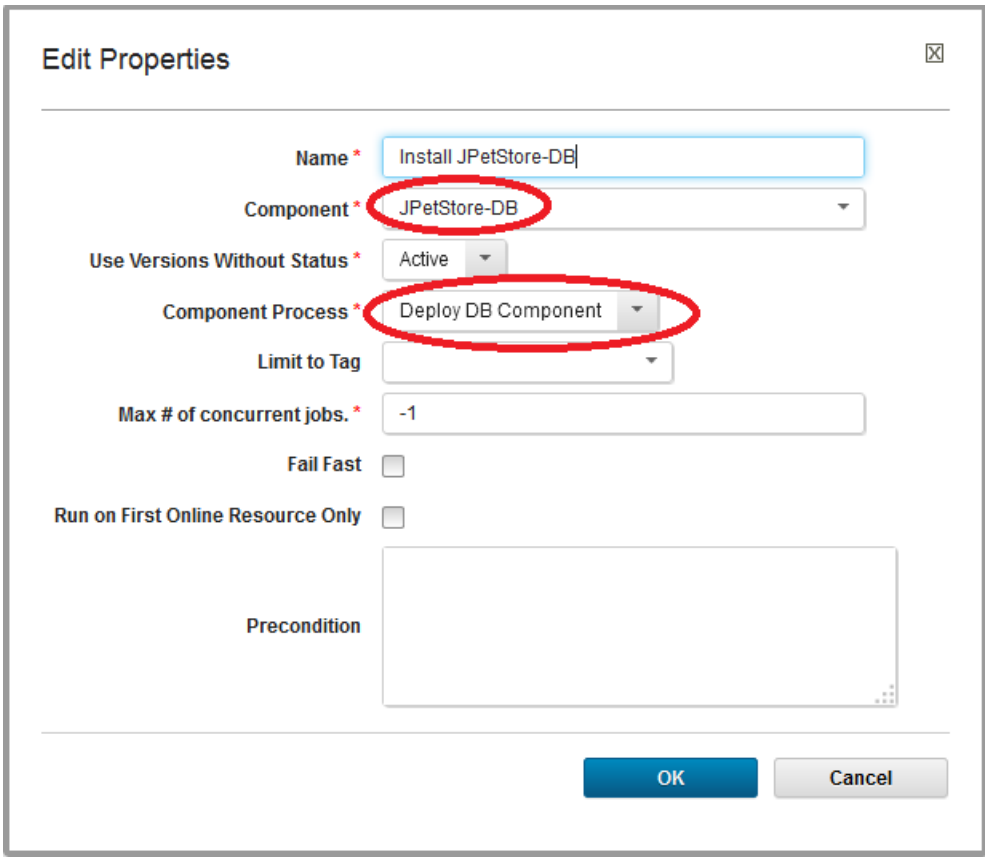
From the lower row of tabs click the **Processes** tab.

Click on the name of the new `Deploy JPetStore` process. This will open the **UrbanCode Deploy Process Designer**.

Drag the **Install Component…** step from the **Palette** into the **Main Editing Area** and drop it just below the `Install JEE Component` step and to the right of the `Install Web Component` step. This will produce the **Edit Properties** dialog.

BEFORE changing the **Name** field, set the **Component** field to `JPetStore-DB`. You will see that the **Name** field automatically updates to reflect the name of the component we are deploying.

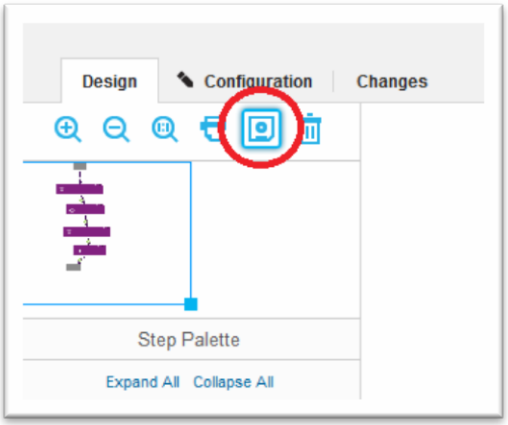
Make sure the Component Process field is set to `Deploy DB Component`.



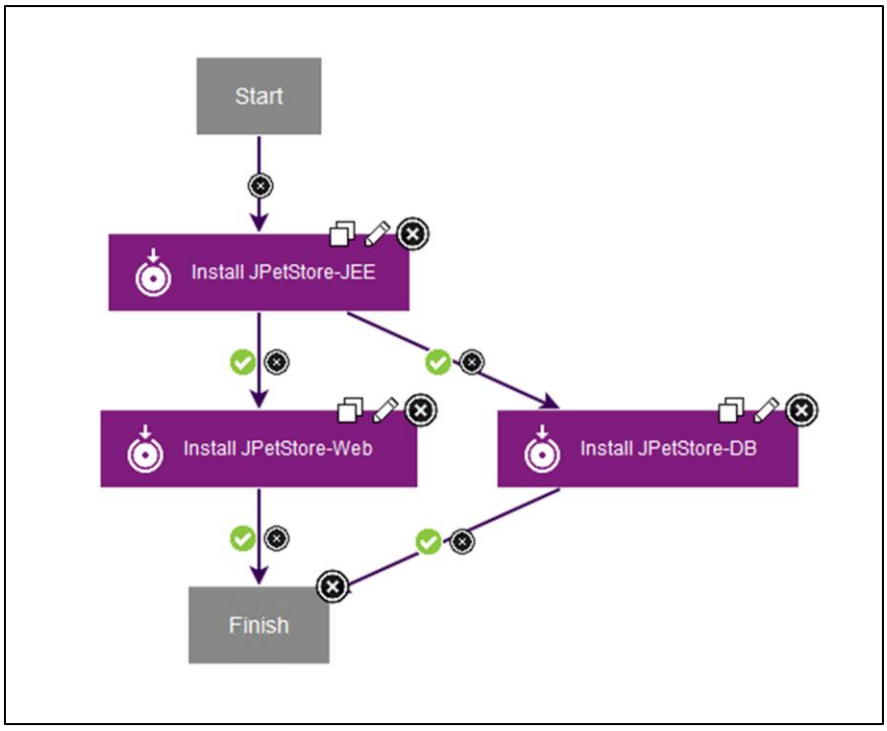
Click the **OK** button.

Draw a Flow Arrow from `Install JEE Component` to the new `Install JPetStore-DB` step we just created.

Draw a **Flow Arrow** from `Install JPetStore-DB` to the **Finish** block. Click the blue **Save** button on the **Palette** to save this process.



Your diagram should now look something like this:



	*Note how two flow arrows come out of the Install JPetStore-JEE step. Both flow arrows have the same green check, which means they will both be processed in parallel if the step is successful* *The Web component and the DB component do not depend on each other. So we can try to install them both at the same time, maybe saving some time for the overall process.*

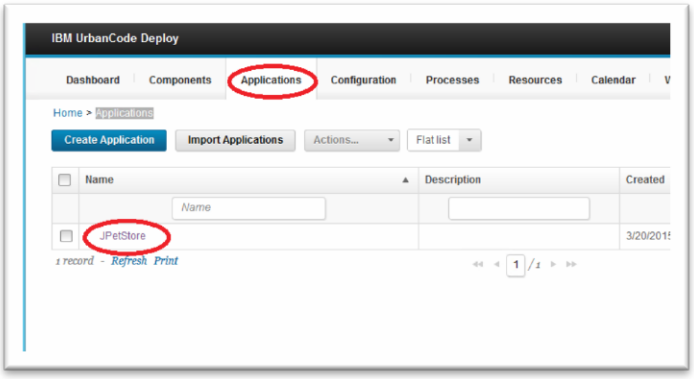
10 Deploy the Complete Application into each Environment

You are now ready to deploy all of the components of JPetStore v1.0 into the **SIT** environment and test the running application. We do this by executing the Application process (**Deploy JPetStore**) against one of the listed environments (**SIT**). Recall that the Application process will coordinate the installation of each of the components, by indicating the execution of each *Component's* deployment process.

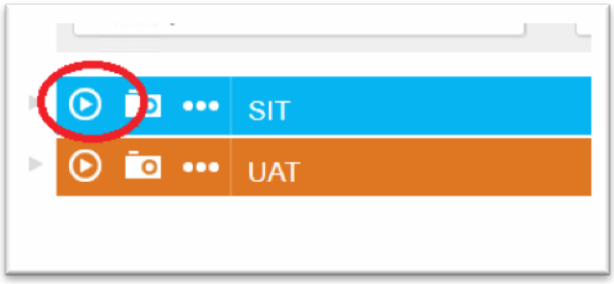
10.1 Deploy JPetStore into the SIT Environment

Click the **Applications** tab in the top row of tabs.

Click on the link for the **JPetStore** application. This will display the two environments we created earlier (**SIT** and **UAT**).

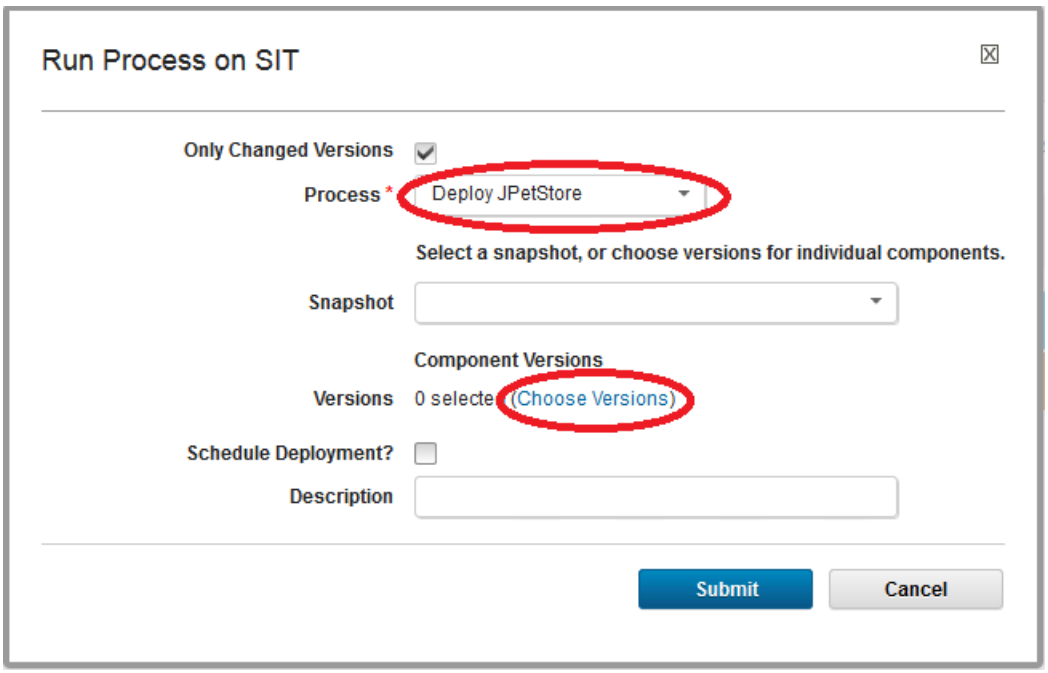


Look at the row for the **SIT** environment – on the far left you see a small icon like a "Play" button from a video player. This is the **Request Process** button. Click this button to initiate an UrbanCode Deploy process.

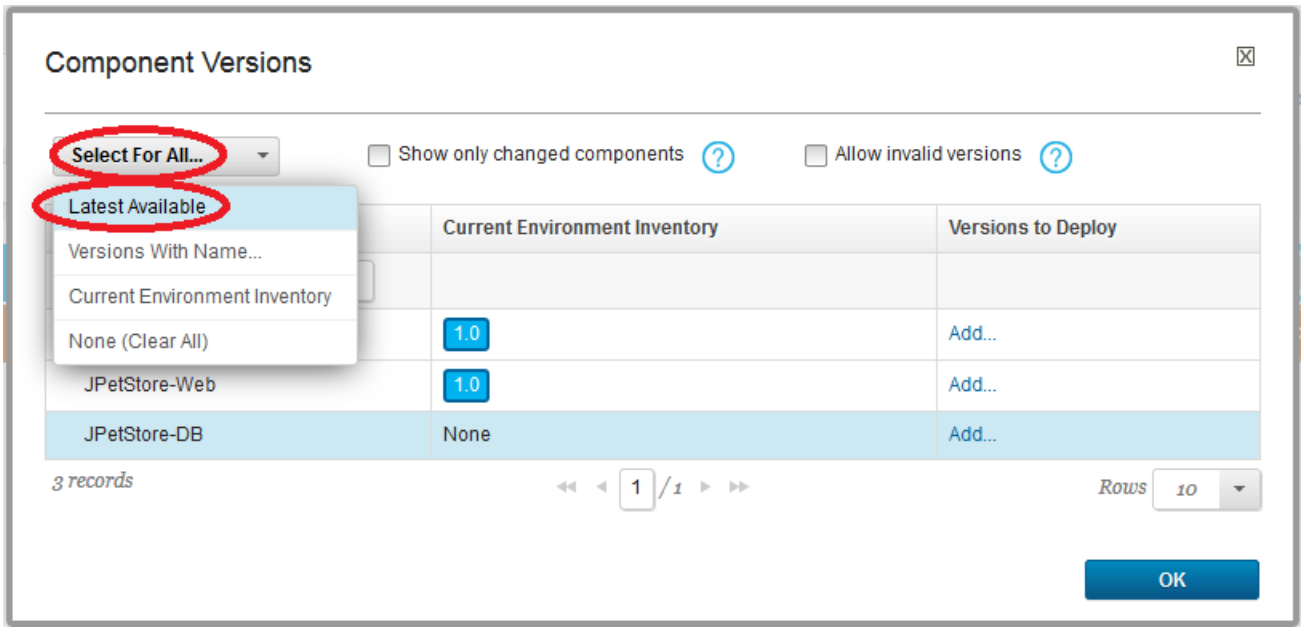


The dialog appears to request a process that will run against the **SIT** environment. Set the **Process** field to **Deploy JPetStore**.

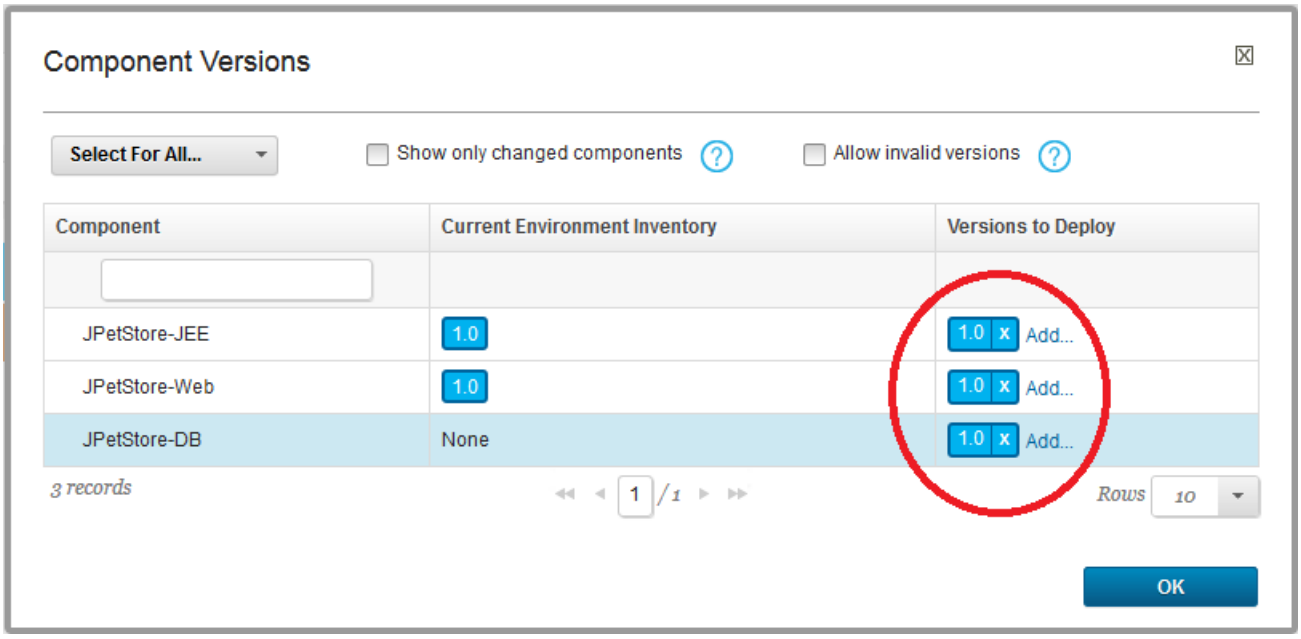
By the **Versions** field, click the link that says **Choose Versions**… another dialog appears.



In the **Component Versions** dialog, set the **Select for All…** dropdown box in the top left to `Latest Available`.



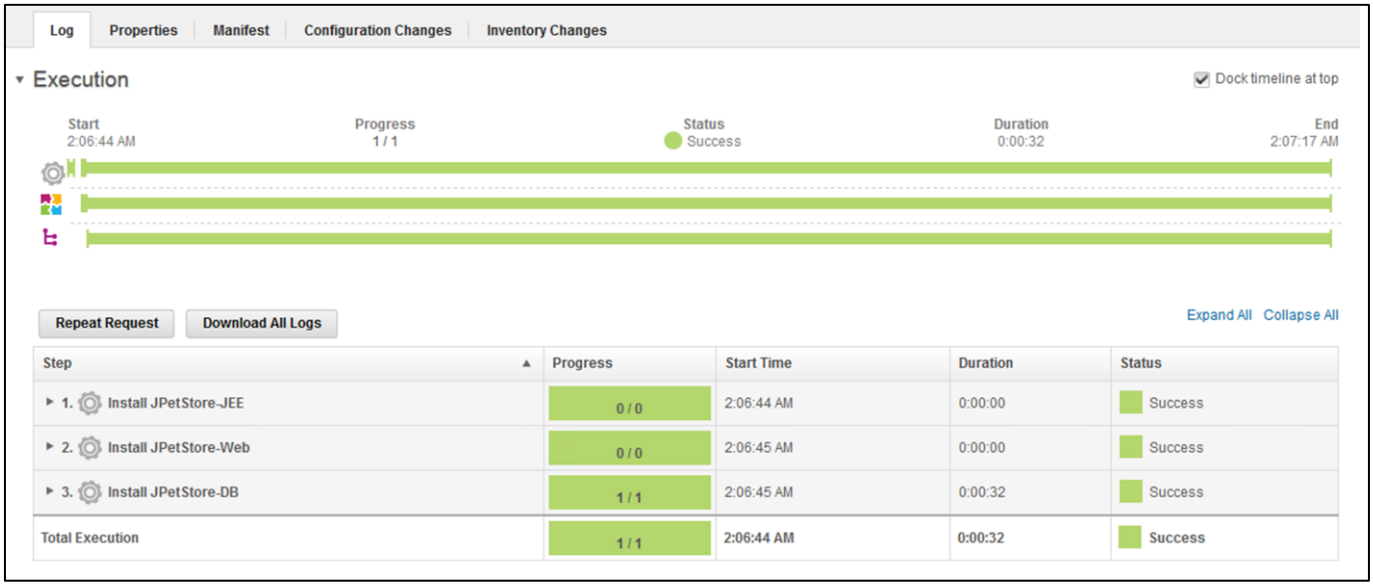
This will add the latest version of all components in the list to the **Versions to Deploy** column.



Click the **OK** button.

Back on the **Run Process on SIT** dialog, click the **Submit** button and UrbanCode Deploy starts the deployment process.

Monitor the status. When UrbanCode Deploy finishes processing, all the components should deploy successfully.



Note that IBM UrbanCode Deploy records each version of a component deployed to a resource. This maintains a historical record and avoids deploying the same version twice. The operator can override this option manually and re-deploy any component at any time, however.

If you expand all the process entries with the 'twisties' on the left, you can see that UrbanCode Deploy recognizes that Web 1.0 and App 1.0 have already been deployed to `SIT` and will not redeploy them. (See the status on the far right.)

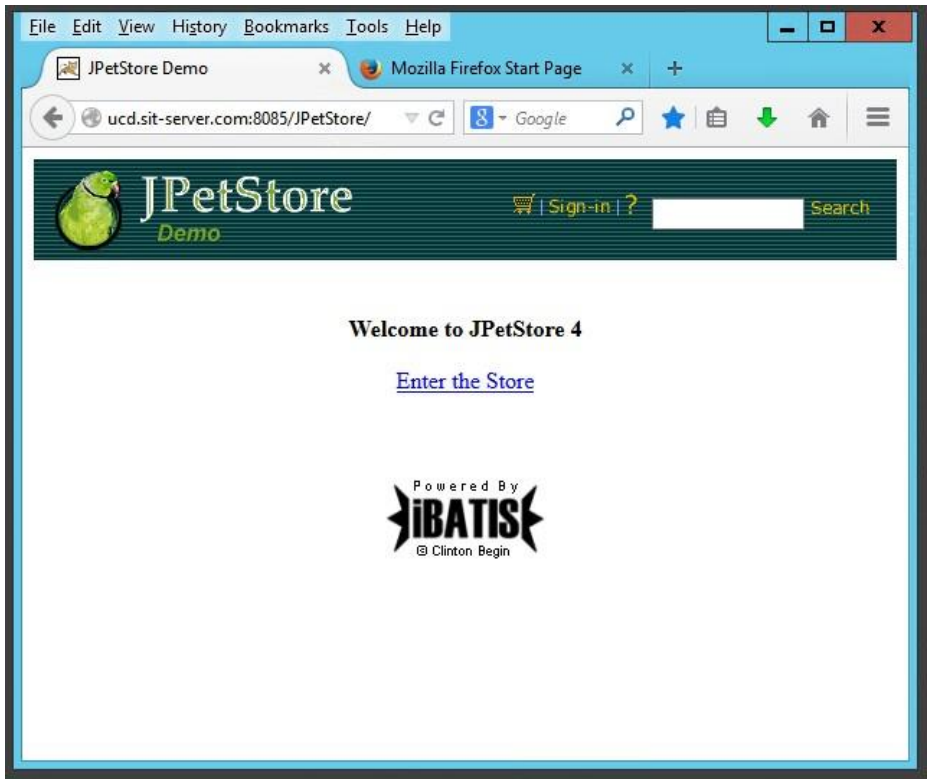
Step	Progress	Start Time	Duration	Status
▼ 1. Install JPetStore-JEE	0 / 0	2:06:44 AM	0:00:00	Success
JPetStore-JEE		2:06:45 AM	0:00:00	Already Installed
▼ 2. Install JPetStore-Web	0 / 0	2:06:45 AM	0:00:00	Success
JPetStore-Web		2:06:45 AM	0:00:00	Already Installed
▼ 3. Install JPetStore-DB	1 / 1	2:06:45 AM	0:00:32	Success
▶ JPetStore-DB	1 / 1	2:06:45 AM	0:00:32	Success
Total Execution	1 / 1	2:06:44 AM	0:00:32	Success

10.2 Run and Test JPetStore in the SIT Environment

Open your browser and go to the site for the SIT version of JPetStore:

http://ucd.sit-server.com:8085/JPetStore/

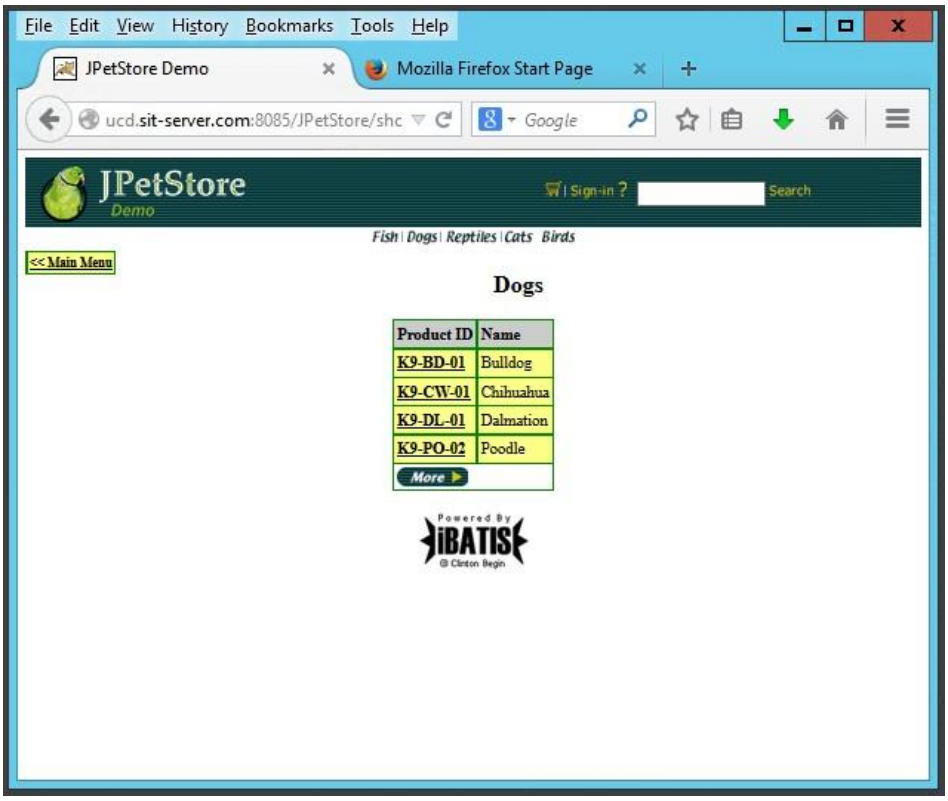
You should see the JPetStore application top page.



If the application has installed and is running correctly, you should be able to execute the following:

Click the **Enter the Store** link.

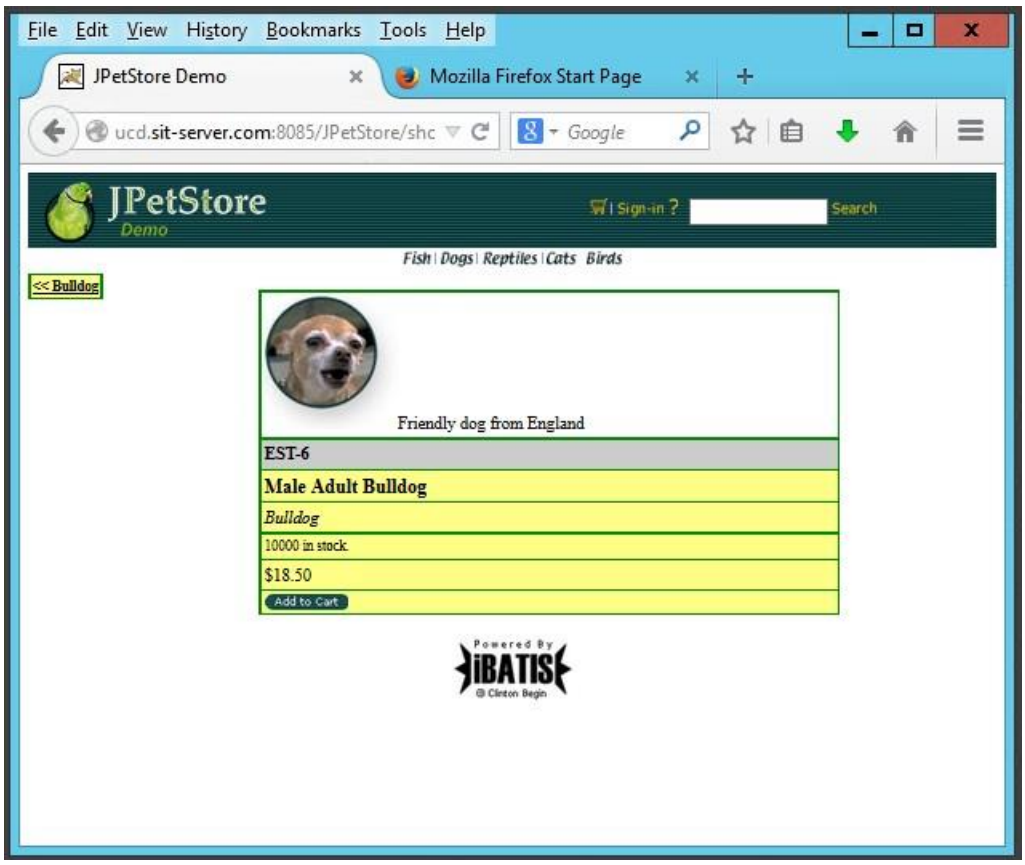
In the left-hand navigation bar, click **Dogs**.



In the main body of the page click the Product ID next to Bulldog (**K9-BD-01**).

In the main body of the next page click the Item ID **EST-6**.

This should show you the screen below. (Yes, the picture is not a bulldog, but that is an intentional content bug, used in our product demos.) If you see the screen below, then congratulations - the application deployment is a success!



10.3 Deploy JPetStore into the UAT Environment

Now repeat the last few sections and deploy the application into the **UAT** environment.

10.4 Run and Test JPetStore in the UAT Environment

Open your browser and go to the site for the UAT version of JPetStore:

http://ucd.uat-app-server.com:8086/JPetStore/

(NOTE – the port is different for the **UAT** environment… 8086)

As before, enter the store, select dogs and make sure you can see a picture of a dog.

10.5 Summary

In this module you created reusable deployment processes for the JPetStore components and application. You used

the processes to successfully deploy version 1.0 of the application to the **SIT** environment.

As we've noticed, however, at least one of the pictures is not correct. There may be other errors in version 1.0 of this application as well. Thankfully the developers have updated some of the components. In the next sections we will examine how UrbanCode Deploy can make use of the updates.

11 Upgrade the SIT Environment and Promote to the UAT Environment

In this lab you will deploy additional versions of the components to upgrade the **SIT** environment to a newer version of the content and database components. You will then create a snapshot of the **SIT** environment and promote the snapshot to the **UAT** environment.

A snapshot is a collection of specific versions of components, usually versions that are known to work together. Typically, a snapshot is created when a successful deployment has been run in an environment.

As the application components advance through the software development lifecycle, they will be installed on multiple environments. Snapshots help manage collections of components that are known to work together as they move through these environments on their way to production systems.

When you have completed this exercise, you will have promoted the application version from the **SIT** environment to the **UAT** environment using a snapshot.

11.1 Check for New JPetStore-DB Versions

Now we'll check to see if our development group has produced any new versions of our components. Although we do this manually, UrbanCode Deploy supports the ability to continually check for new versions and automatically iimport them into CodeStation (a code repository that is included with UrbanCode Deploy).

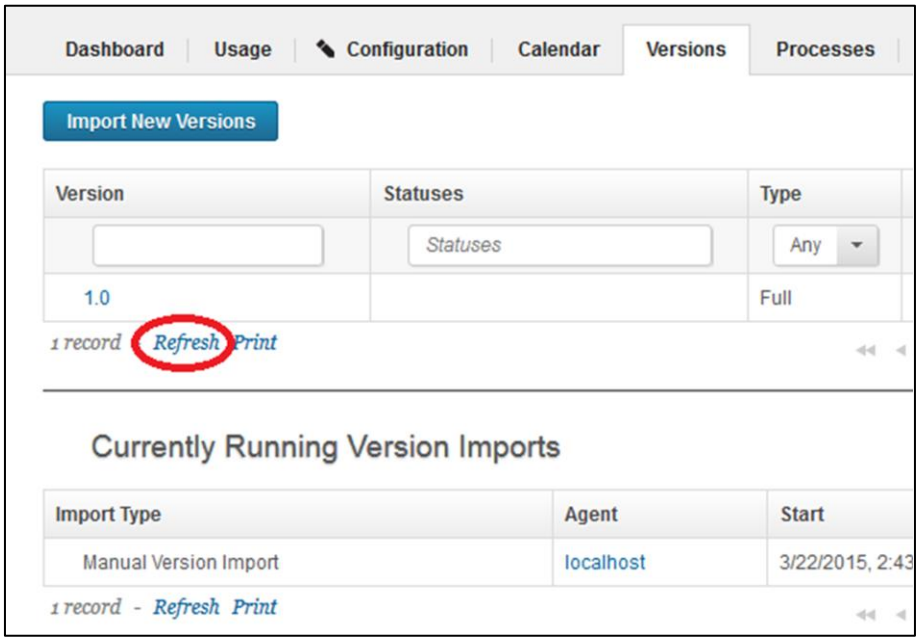
First we'll check for any new versions of the database component.

11.1.1 Check for a new Component version

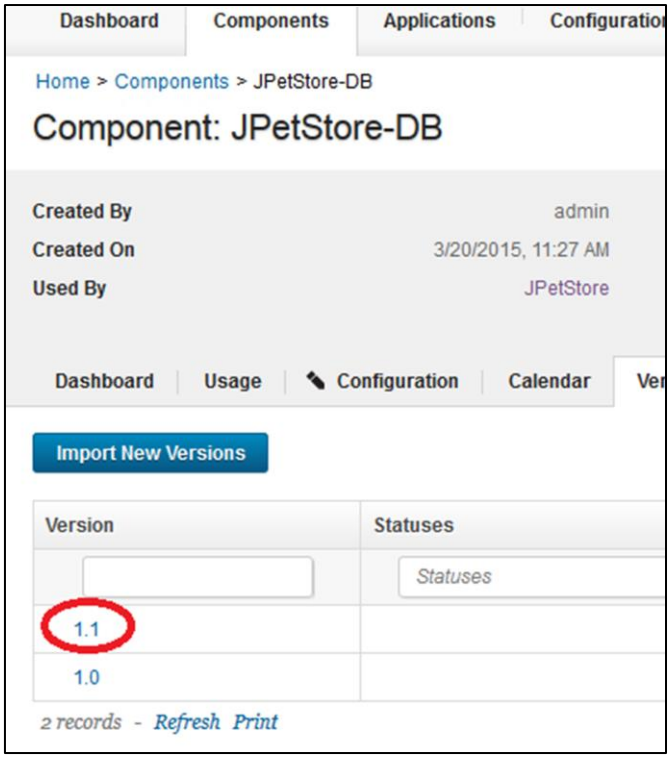
In the upper row of tabs click the **Components** tab.

In the resulting list, click **JPetStore-DB** to select that component. In the lower row of tabs click the **Versions** tab. Click the button to **Import New Versions**.

Wait a moment and click the **Refresh** link *above* the section named **Component Request History**. (UrbanCode Deploy does not automatically update this screen after the version import.)

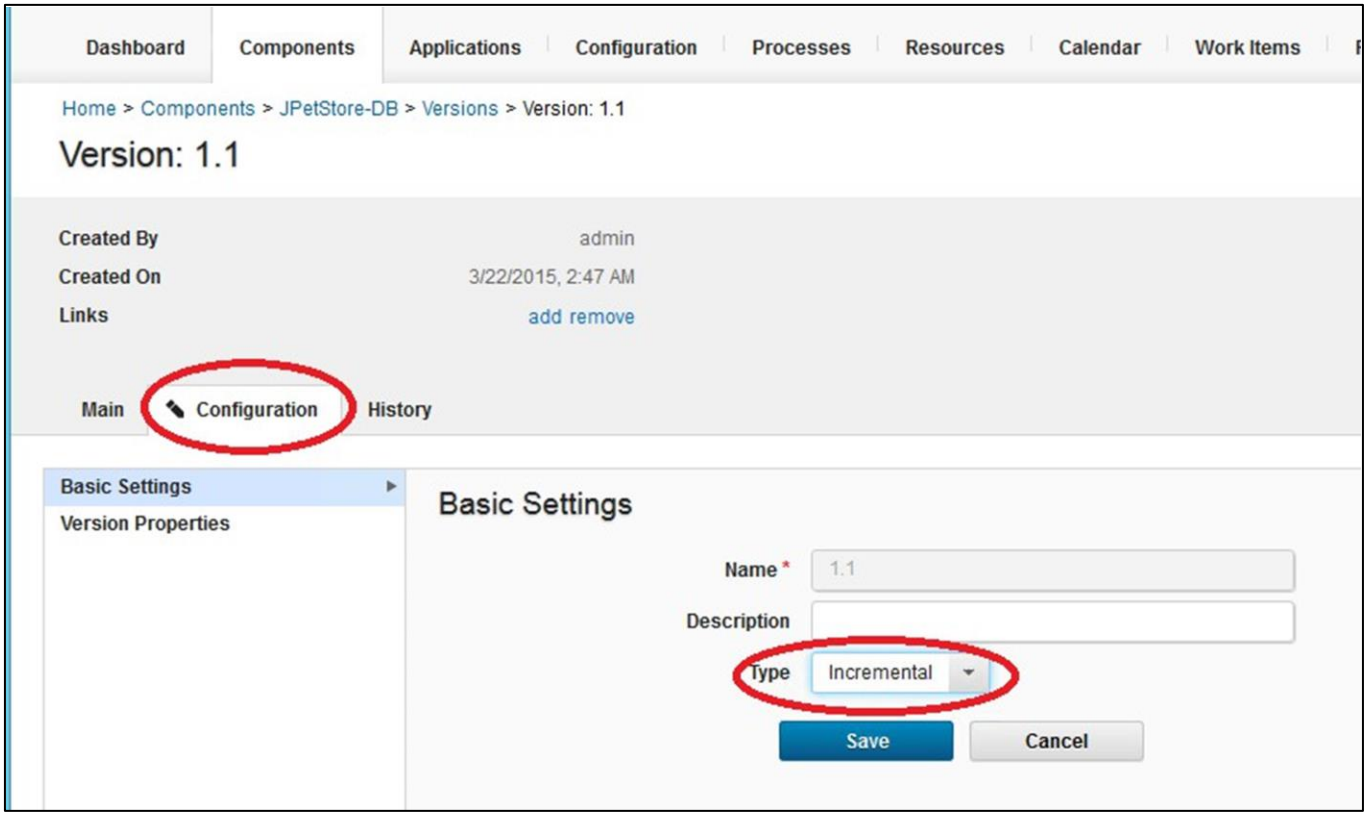


This should refresh the page and you should see that UrbanCode has discovered a new version of the database component and imported it into CodeStation! (Version 1.1)



Click on the new version of `JPetStore-DB` (Version 1.1).

In the lower set of tabs click the **Configuration** tab. Change the **Type Field** to `Incremental`.



Click the **Save** button.

Versions that are termed as “incremental” are dependent on previous versions. So in this case, we’ve told UrbanCode Deploy that Version 1.1 of the database is only an incremental update to the previous version.

`JPetStore-DB` version 1.0 loads a listing of all dogs EXCEPT the listing for the Bichon dog. **JPetStore-DB** version 1.1 is an incremental version that only contains one addition; the listing for the Bichon dog.

11.2 Deploy Updates to the SIT Environment

In this section you will update the deployed version of the `JPetStore` application out on the `SIT` Tomcat server. This should update the listing to include the Bichon dog.

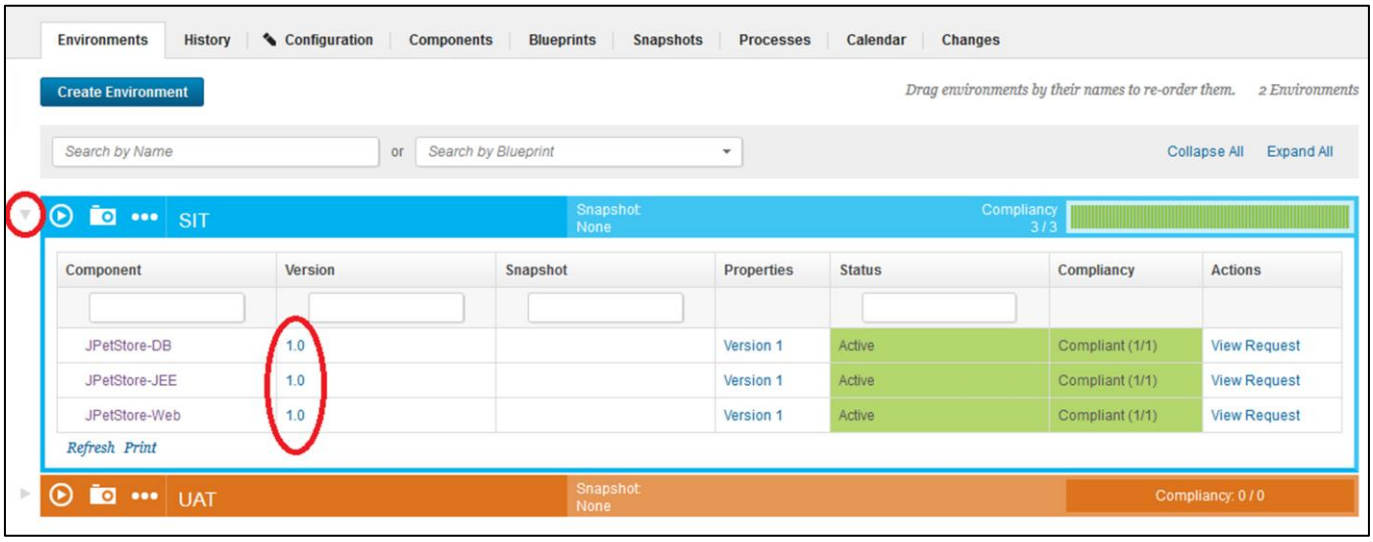
11.2.1 Verify the current contents of the SIT environment

In the upper row of tabs click the **Applications** tab.

Click on the link for the `JPetStore` application.

Click the ‘twistie’ to the left of the `SIT` environment to expand it.

See how it shows the version of each component that UrbanCode Deploy installed last (1.0 for each component).



Open another browser window and go to SIT version of the JPetStore website:

http://ucd.sit-server.com:8085/JPetStore/

Enter the store and navigate to the **Dog breed** page.

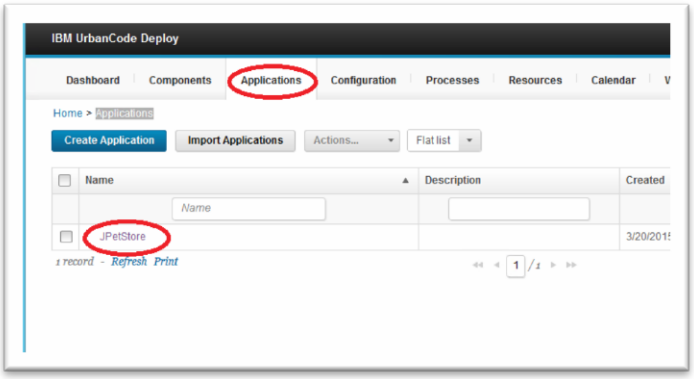
Confirm that there are only 6 products on the list of dogs. (Click on the **More** button to see the last two entries.) Note specifically the absence of a listing for a Bichon.

V1.0 of the `JPetStore-DB` component does NOT have any listings for Bichons.

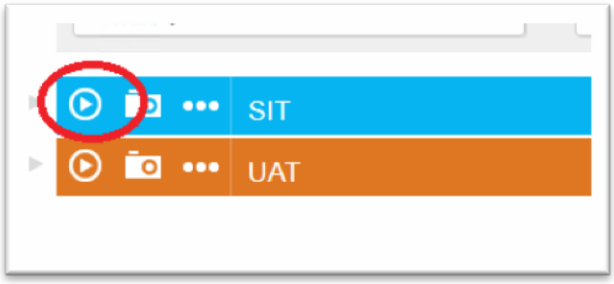
11.2.2 **Re-Deploy JPetStore into the SIT Environment with the New Component Version**

Click the **Applications** tab in the top row of tabs.

Click on the link for the `JPetStore` application. This will display the two environments we created earlier (`SIT` and `UAT`).

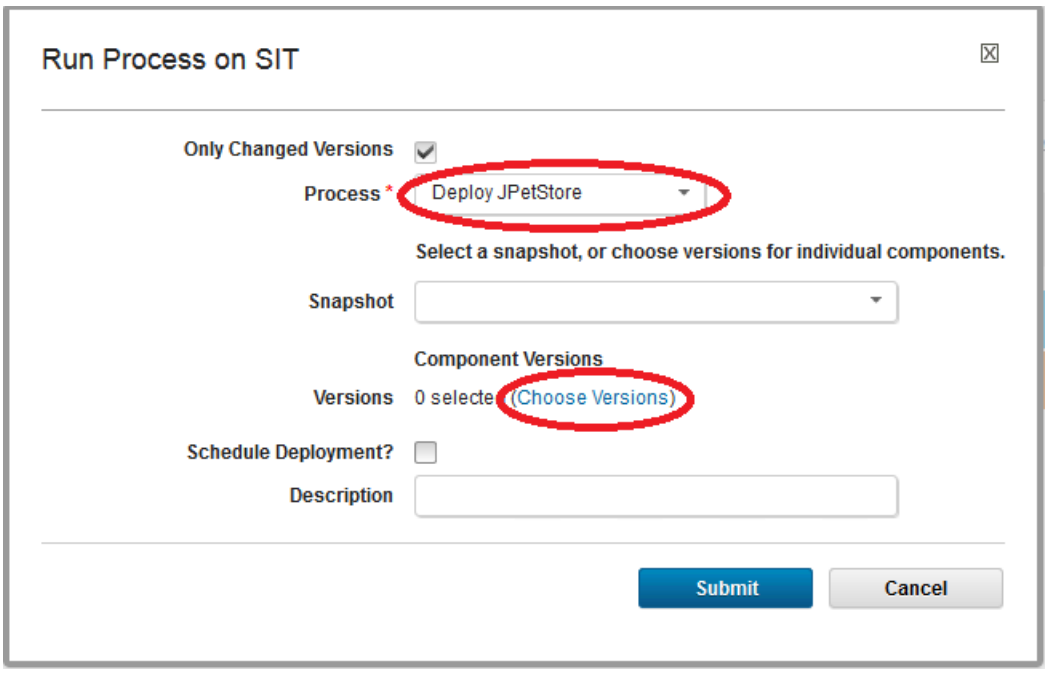


Look at the row for the `SIT` environment – near the left of the row you see a small icon like a “Play” button. This is the **Request Process** button. Click this button to initiate an UrbanCode Deploy process.

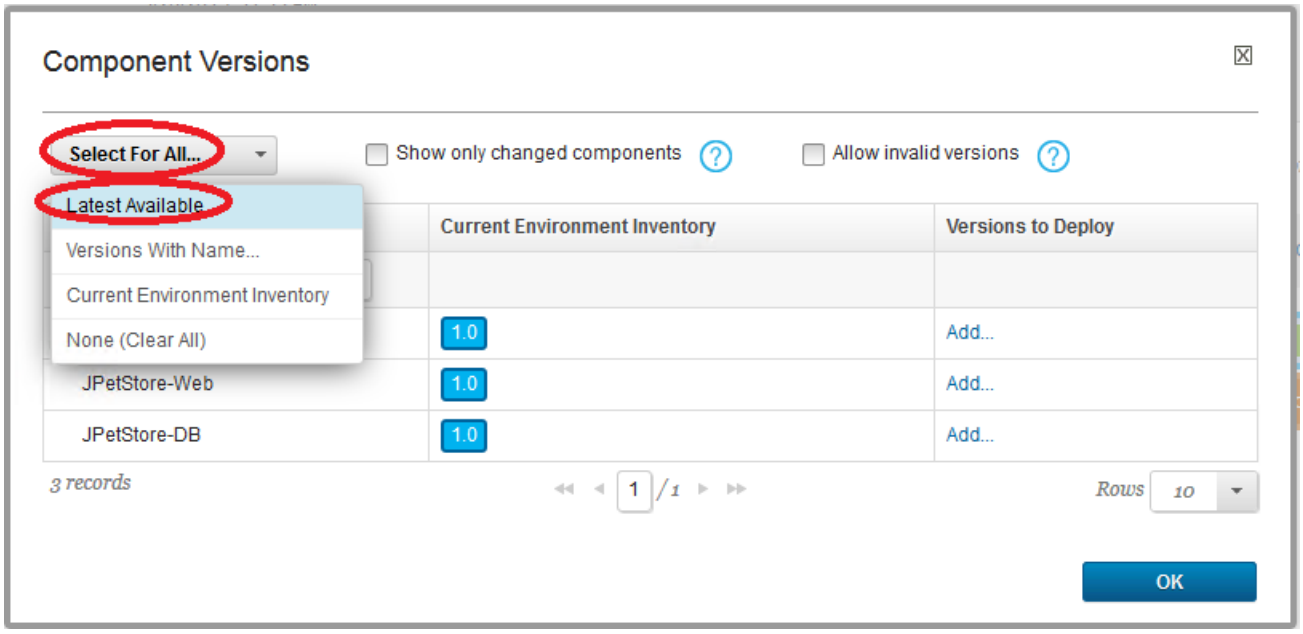


A dialog appears which allows us to request a process that will run against the **`SIT`** environment. Set the **Process** field to **`Deploy JPetStore`**.

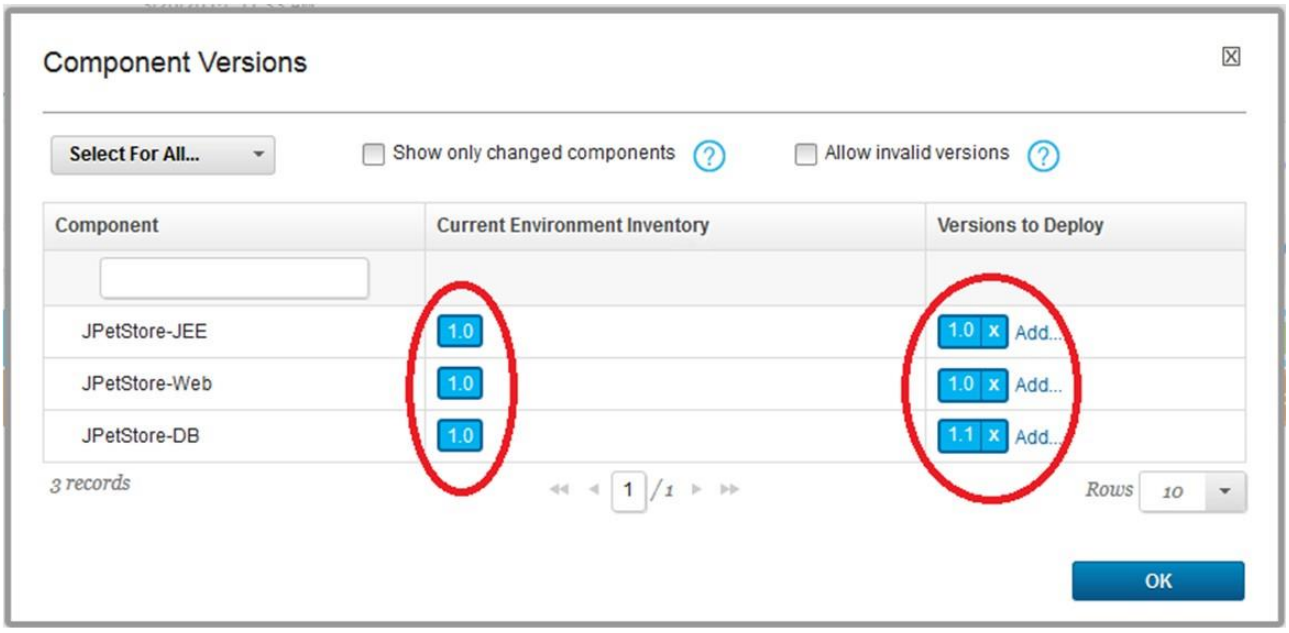
By the **Versions** field, click the link that says **Choose Versions**… another dialog appears.



In the **Component Versions** dialog, set the **Select for All…** dropdown box in the top left to **`Latest Available`**.



Compare the two columns to see what versions are currently installed, and what versions will be installed next.

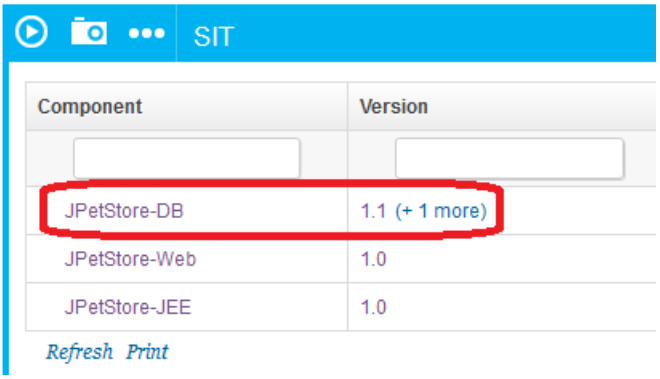


Click the **OK** button.

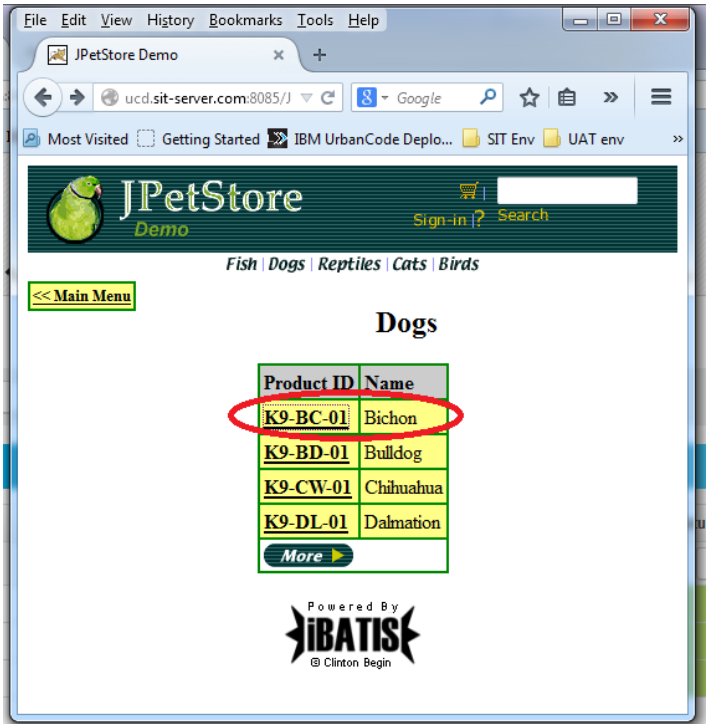
Back on the **Run Process on SIT** dialog, click the **Submit** button and UrbanCode Deploy starts the deployment process.

Monitor the status. When UrbanCode Deploy finishes processing, all the components should deploy successfully.

Return to the Application Environment page and expand the `SIT` environment again to see that the `JPetStore-DB` component has been updated to version 1.1.



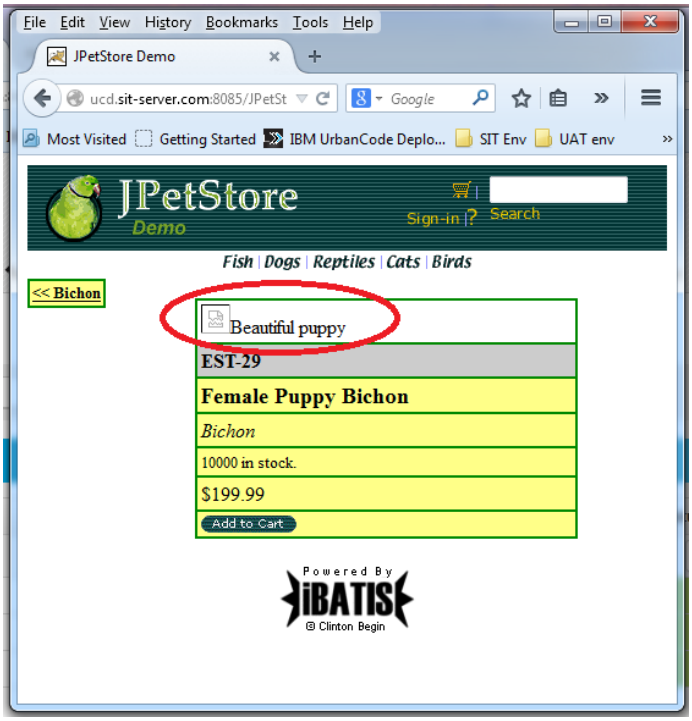
Return to the browser window showing the Pet Store application, and navigate to the list of dogs again. Note the list includes Bichon now!



In the main body of the page click the Product ID next to the new Bichon listing (**K9-BC-01**).

Again in the main body of the next page click the Item ID **EST-29**.

Note that the picture of the Bichon is missing. This is an indication that the developers may have provided updates to other components of our application.



	If you are using this document to build a Virtual Machine for a product demo, you should stop at this point and take a snapshot of your Virtual Machine.

11.3 Check for New JPetStore-Web Versions

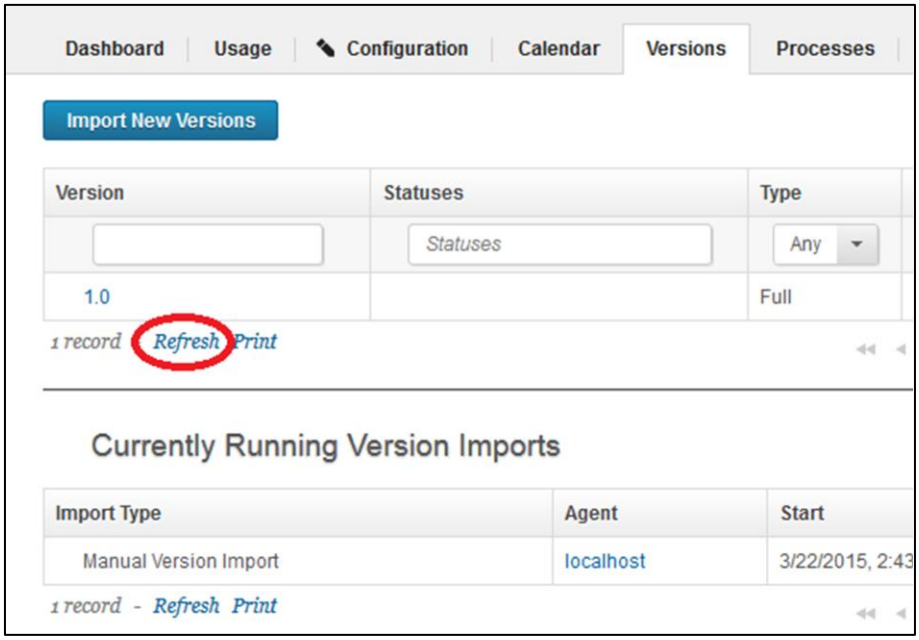
Now we'll check for any new versions of the web component (which contains the image files).

11.3.1 Check for a new Component version

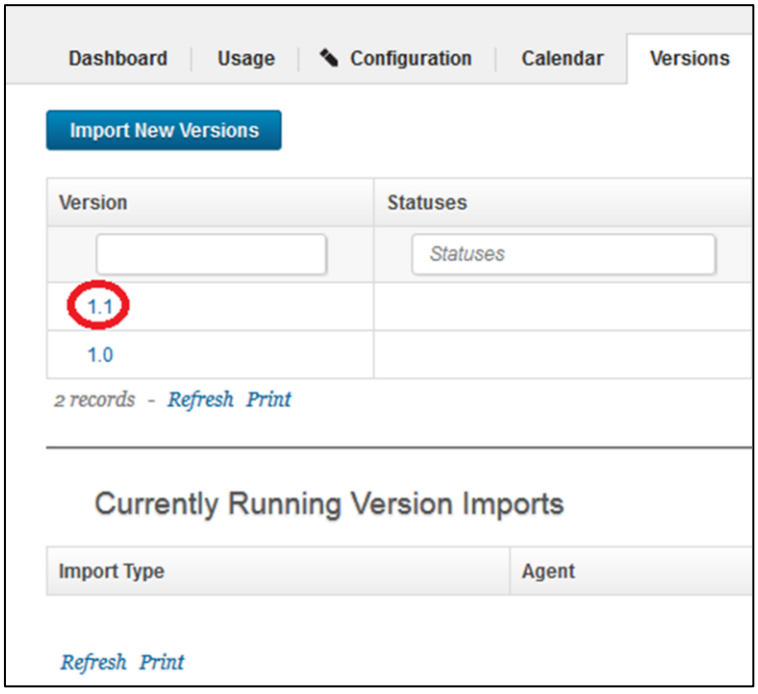
In the upper row of tabs click the **Components** tab.

In the resulting list, click `JPetStore-Web` to select that component. In the lower row of tabs click the **Versions** tab. Click the button to **Import New Versions**.

Wait a moment and click the **Refresh** link *above* the section named **Component Request History**. (Again, UrbanCode Deploy does not automatically update this screen after the version import.)



Like before, you should see that UrbanCode has discovered a new version of this component and imported it into CodeStation! (Version 1.1)



11.3.2 **Deploy Updates Again to the SIT Environment**

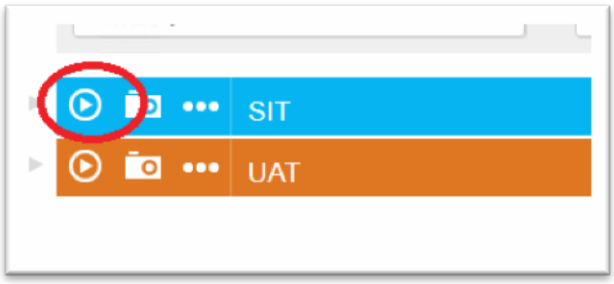
Now that we have found another new version of a component, we need to rerun the deployment process for this application. Notice now that this is the on-going procedure that we use when new versions of components come out of development. This is what we mean when we talk about “one button deployments”. The configuration has been done in the application, components, environments and properties in UrbanCode Deploy – and ongoing deployments simply amount to executing the deployment processes.

11.3.3 **Re-Deploy JPetStore into the SIT Environment with the New Component Version**

Click the **Applications** tab in the top row of tabs.

Click on the link for the `JPetStore` application. This will display the two environments we created earlier (`SIT` and `UAT`).

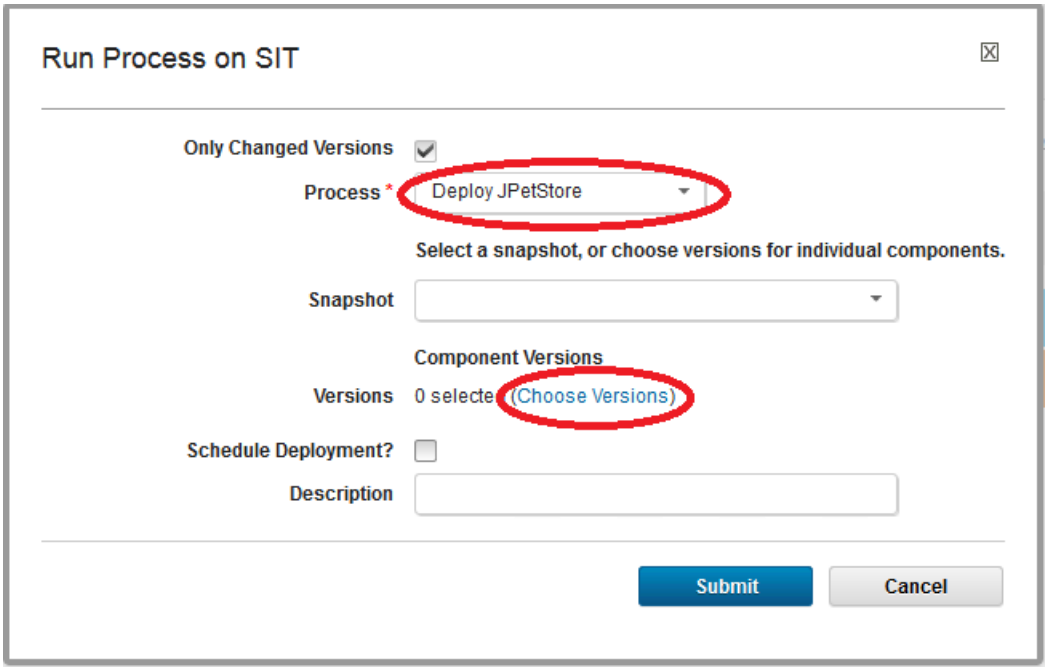
Again, click the **Request Process** button for the `SIT` environment.



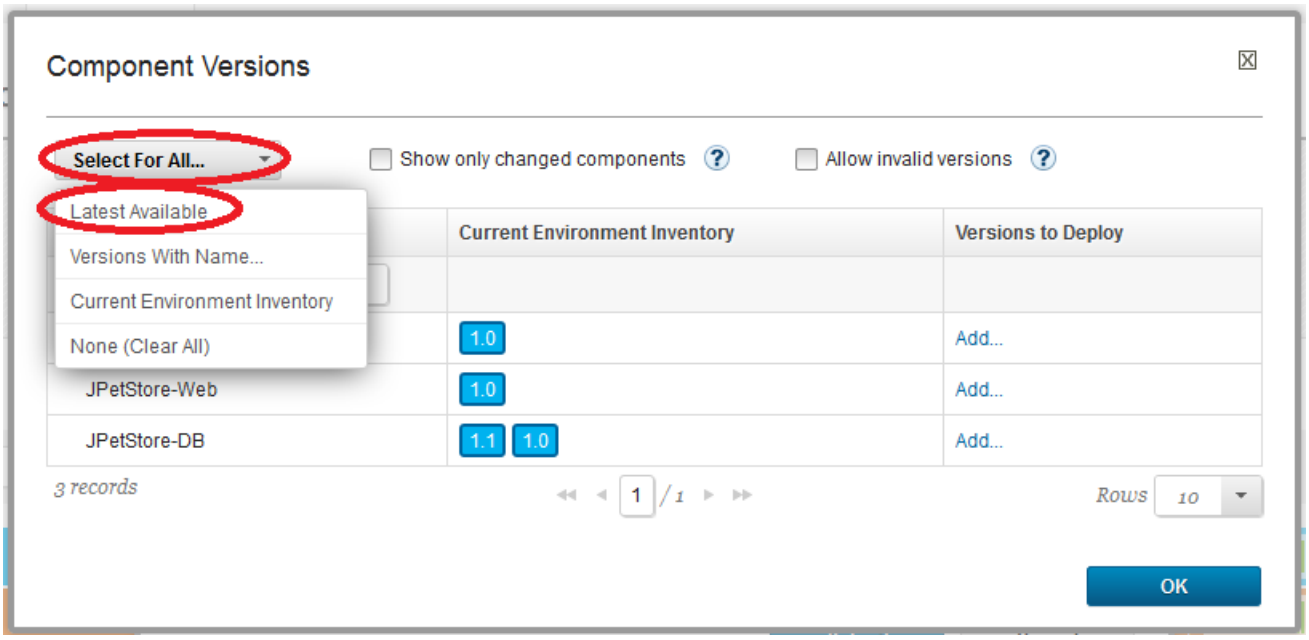
The dialog appears to request a process that will run against the `SIT` environment. Set the **Process** field to `Deploy JPetStore`.

In the **Run Process on SIT** dialog set the Process field to `Deploy JPetStore`.

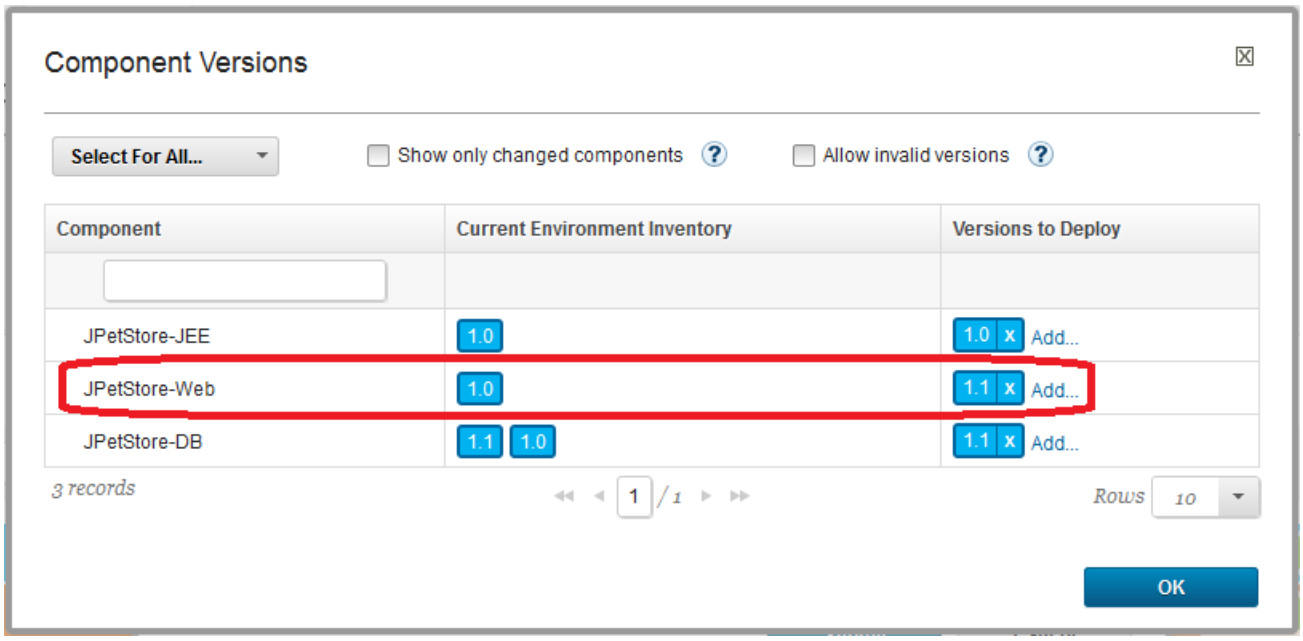
By the **Versions** field, click the link that says **Choose Versions**… another dialog appears.



In the **Component Versions** dialog, set the **Select for All…** dropdown box in the top left to `Latest Available`.



Compare the two columns to see what versions are currently installed, and what versions will be installed next. Note that the **`JPetStore-Web`** component will be upgraded to version 1.1.



Click the **OK** button.

Back on the **Run Process on SIT** dialog, click the **Submit** button and UrbanCode Deploy starts the deployment process.

Monitor the status. When UrbanCode Deploy finishes processing, all the components should deploy successfully.

Return to the Application Environment page and expand the **SIT** environment again to see that the **JPetStore-Web** component has been updated to version 1.1.

Return to the browser window showing the Pet Store application, and navigate to the list of dogs again.

In the main body of the page click the Product ID next to the Bichon listing (**K9-BC-01**).

Again in the main body of the next page click the Item ID **EST-29**.

You should now see a picture included with the Bichon listing. (Although the picture is not a Bichon… it is a Pitt Bull puppy - - so our developers have more work to do! But this is ok for now.)

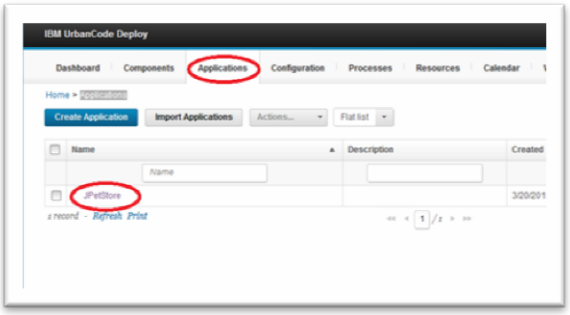
11.4 Promote JPetStore from SIT to UAT with a Snapshot

In this lab you will create a snapshot of the component versions now deployed into the **SIT** environment and then promote those components to the **UAT** environment by using that snapshot.

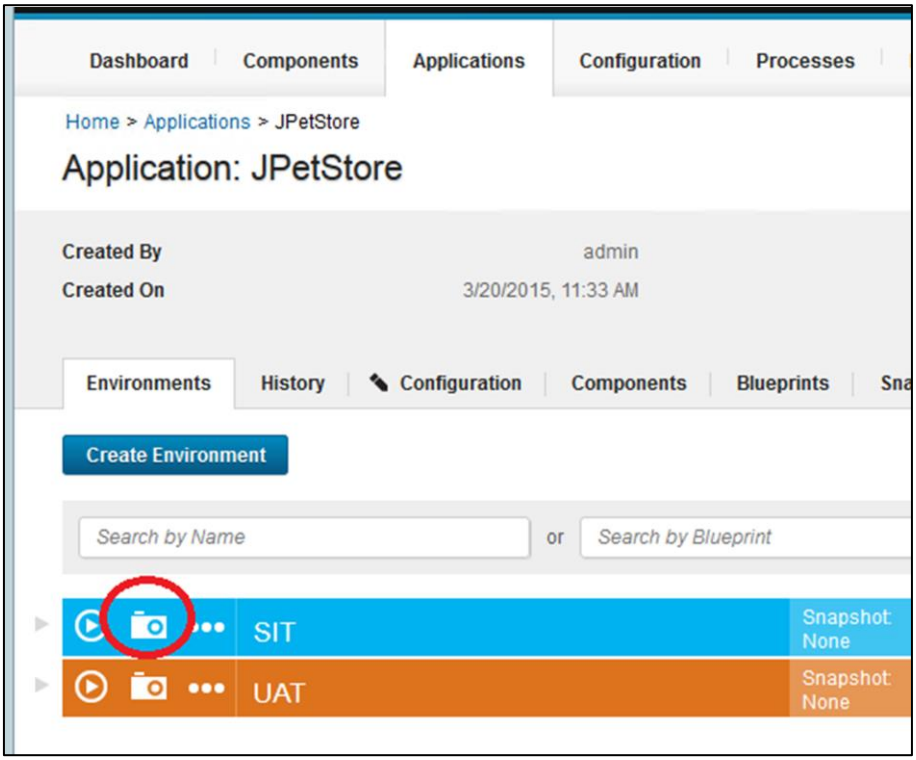
11.4.1 Create the snapshot of SIT

Click the **Applications** tab in the top row of tabs.

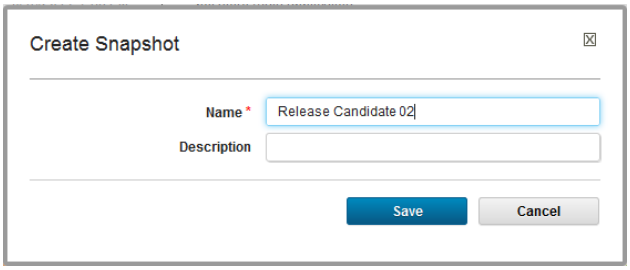
Click on the link for the `JPetStore` application.



In the row for the `SIT` environment, click the icon that looks like a camera – this starts the process to create a snapshot of whatever versions of components are currently deployed in that environment.

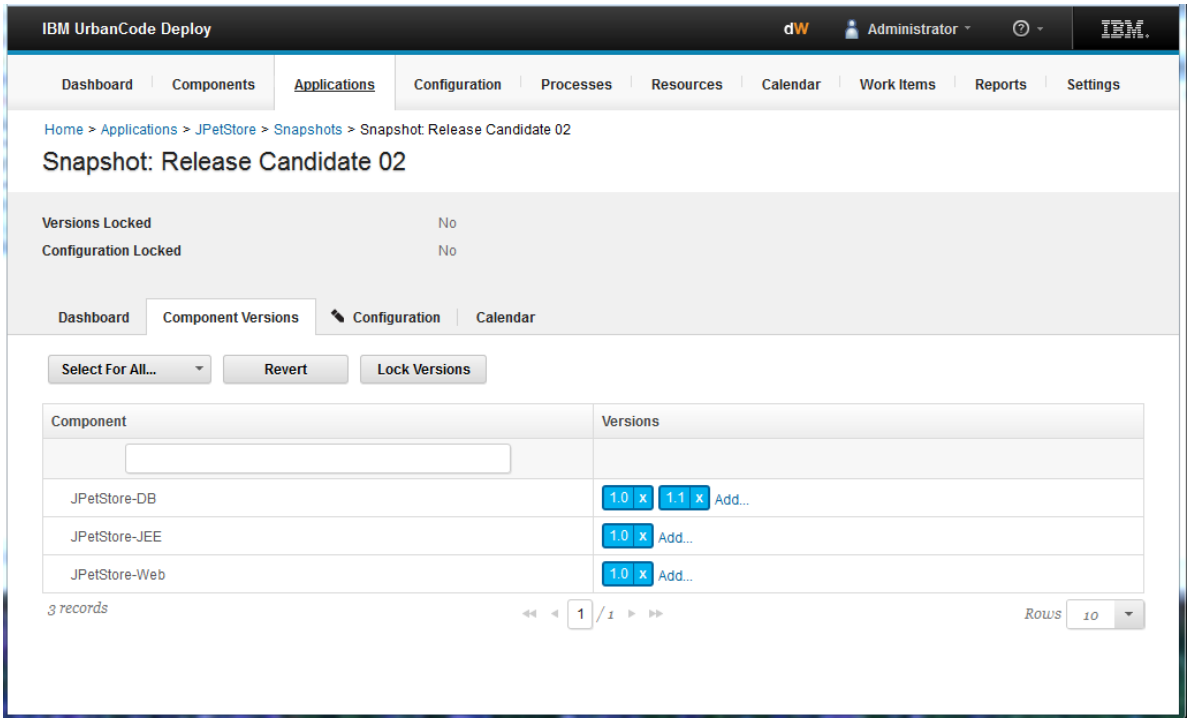


Enter `Release Candidate 02` in the **Name** field and click the **Save** button.



The snapshot is populated with the current content of the **SIT** environment.

Go to **Component Versions** tab. Notice that the Database component includes two versions – the Incremental version 1.1 is not sufficient to recreate the environment, so both are automatically included. Both are needed to re-create the application currently deployed in **SIT**.



11.4.2 **Promote the Snapshot to the UAT Environment**

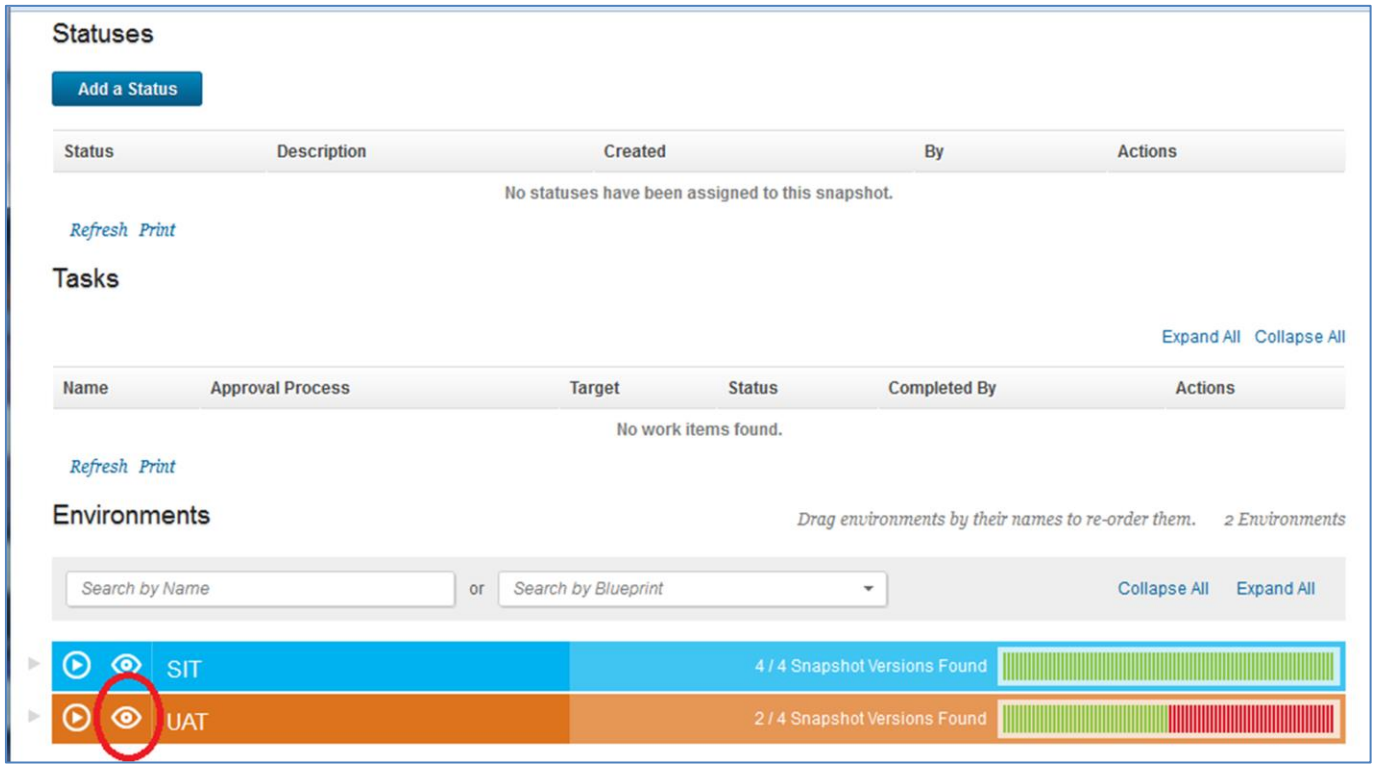
Before we actually run the deployment, UrbanCode Deploy gives us a way to see what will happen to the **UAT** environment ahead of time. The **Preview** feature of UrbanCode Deploy Snapshots provides this capability.

Click the **Applications** tab in the top row of tabs.

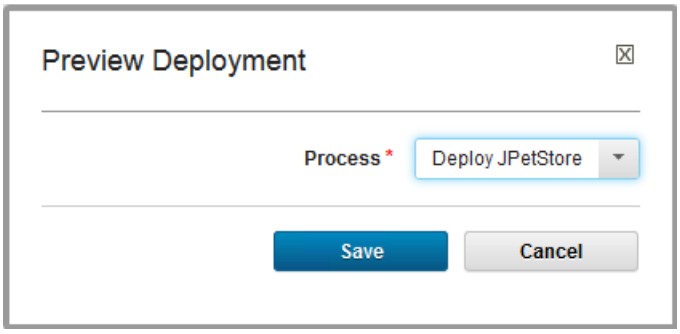
Click on the link for the **JPetStore** application.

On the lower row of tabs click the **Snapshots** tab.
In the list of snapshots, click the link for **Release Candidate 02**.

Scroll down until you can see the environments and click the icon that looks like an **eye** to the left of the **UAT** environment.

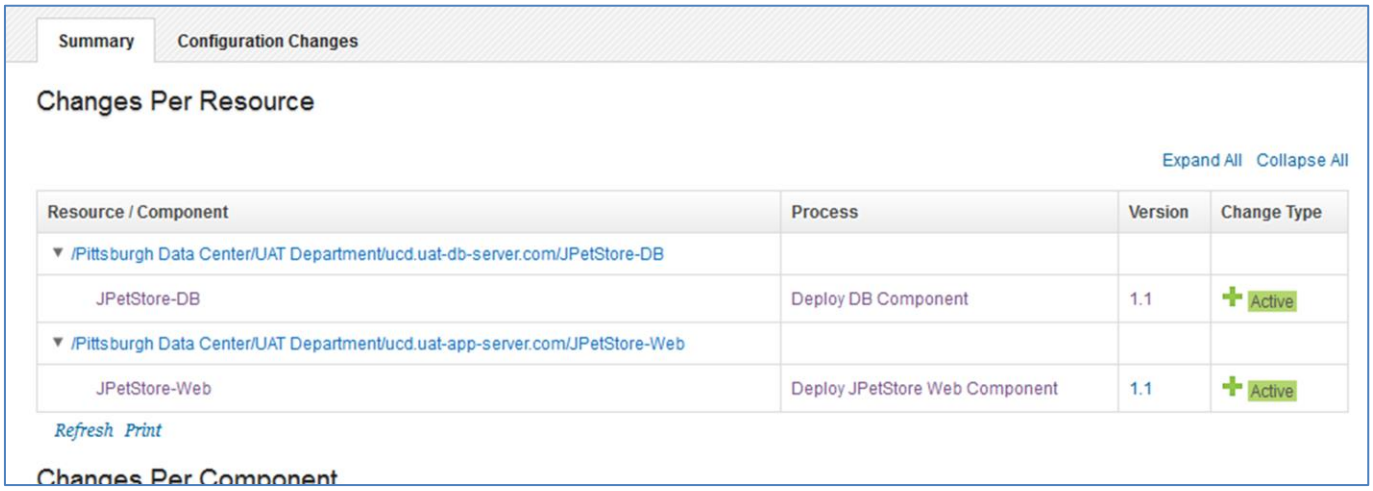


Click this **Preview** button and the **Preview Deployment** dialog appears.



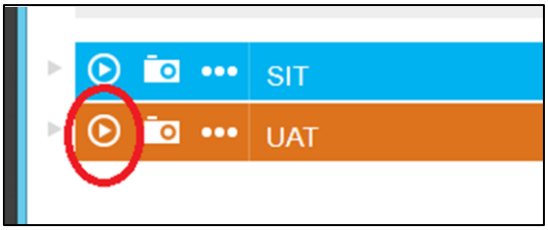
Make sure the **Process** field is set to `Deploy JPetStore` and click the **Save** button.

In the resulting view, look in the section entitled **Changes Per Resource** and use the 'twisties' on the left to expand each row. On the right you can see the version of each component that UrbanCode Deploy will install if you use this snapshot.



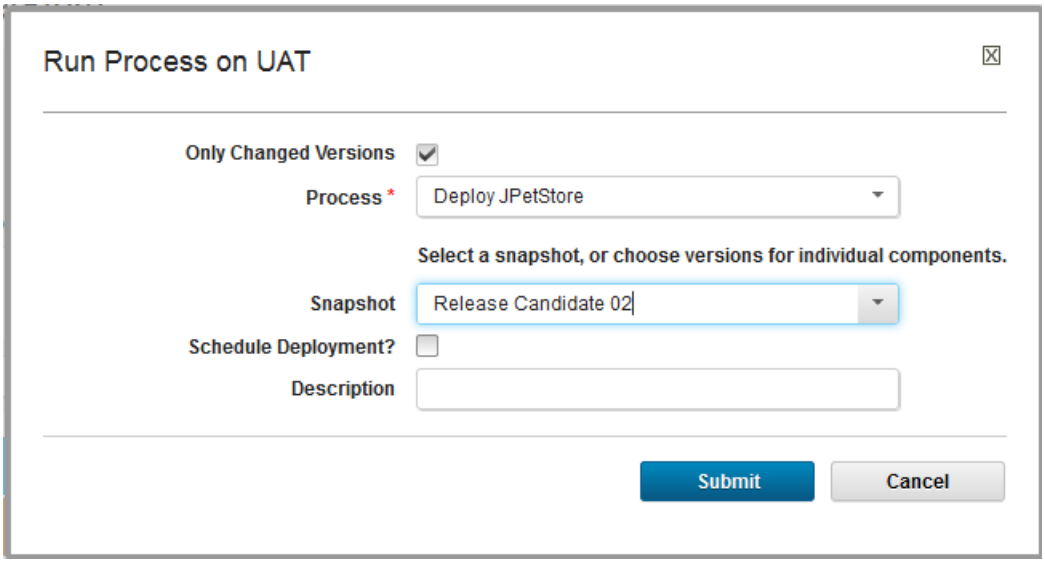
Now that you have verified what will be promoted to the **UAT** environment with this snapshot, you are now ready to request the deployment to be carried out.

Click the **Applications** tab in the top row of tabs. Click on the link for the `JPetStore` application.
On the row for the `UAT` environment, click the Request Process button (near the left).



In the **Run Process on UAT** dialog set the Process field to `Deploy JPetStore`.

Set the Snapshot field to `Release Candidate 02`.



Click the **Submit** button to start the deployment.

Monitor the progress and look for errors. The process should complete successfully.

Now the **SIT** environment is promoted to **UAT** environment.

You can also verify the promotion by opening a new browser and pointing to the **UAT** application server machine:

http://ucd.uat-app-server.com:8086/JPetStore

You should see the top page, be able to enter the store and see pictures of the animals – including the new Bichon with picture.

11.5 Summary

In this lab, you learned how to deploy specific component versions, in this case version 1.1 of **JPetStore-Web** and **JPetStore-DB** to the **SIT** environment. You also learned how to create a snapshot using the components in the **SIT** environment and promote that snapshot to the **UAT** environment.

12 Appendices

Appendix A. Notices

This information was developed for products and services offered in the U.S.A.

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 Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku Tokyo 106-0032, Japan

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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have

been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix B. Trademarks and Copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
System z	*Tivoli*	*WebSphere*	*Workplace*	*System p*	

Firefox is a trademark of the Mozilla Foundation, All rights reserved JPetStore application Copyright (c) 2004 Clinton Begin. All rights reserved VMware is a trademark of VMware, Inc.
Deploy is a trademark of UrbanCode, an IBM Company

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.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

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.

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.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

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

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

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.
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.