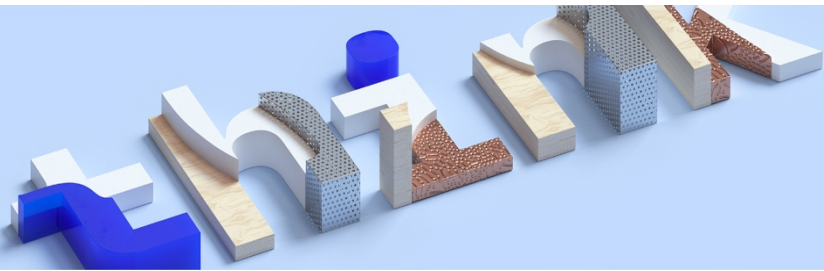


think 2018

IBM



Lab Center – Hands-on Lab

Session 7232

Hands-on Hybrid Management with IBM Cloud Private

Tom Neal, HCL Technologies, thomas.neal@hcl.com

Table of Contents

Disclaimer	4
Abstract	6
About this workshop	6
Lab overview	6
Accessing the lab environment	6
Environment description	6
Before you begin	6
Exercise 1: Import a version and deploy to the test environment	8
View microservices in IBM Cloud Private	8
View the StockTrader application environments	10
Import the StockTrader version	12
Fork the GitHub StockTrader project	12
Update the configuration and import the new version	13
Deploy the application to the test environment	15
Choose the component versions and run the application process	15
Observe the application process request	17
View the test results in Rational Performance Tester	20
Locate the snapshot and view the deployed components	21
Verify the application in the test environment	24
View the microservice in IBM Cloud Private	25
Exercise 2: Use properties to set the deployment environment	27
Create a new version and publish a release	27
Replace hard-coded environment specifications with tokens	27
Create and publish a release	29
Import the parameterized version into UrbanCode Deploy	31
View the Replace Tokens step in the component process	32
Deploy the parameterized YAML file to the test environment	35
Choose the component versions to deploy and run the application process	35
Verify the second deployment to the test environment	39
Exercise 3: Deploy the snapshot to the production environment	40
Deploy the snapshot to the production environment	40

Verify the application deployment to the production environment	42
View the microservice in IBM Cloud Private (production).....	44
Summary	45
What's next	45
We Value Your Feedback!	45

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.

In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

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 about this publication 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. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

© 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Getting acquainted with the lab

Abstract

IBM Cloud Private is a powerful platform for building your next-generation applications, especially when modernizing existing application investments. But the reality is that during a transformation effort, “pieces” of your applications will reside partly on the IBM Cloud Private platform and partly outside of it, either on existing infrastructure or a public cloud. Organizations on this journey need automated deployment and testing solutions that deliver reliability, repeatability, and consistency. This lab demonstrates how IBM UrbanCode continuous delivery tools and IBM Rational testing capabilities ensure quality—no matter what you need to deploy, or where you need to deploy it.

About this workshop

In this lab, you deploy a Java-based microservices application, named *StockTrader*, into an IBM Cloud Private cluster using UrbanCode Deploy.

Lab overview

As part of the UrbanCode Deploy deployment processes, you invoke Rational Performance Tester to ensure that the *StockTrader* application is deployed and running successfully. Next, you deploy *StockTrader* into both test and production environments to show how UrbanCode Deploy can help control reliability, repeatability, and consistency—even across environments with different topologies. Because you are working with a single IBM Cloud Private cluster, you use Kubernetes namespaces to separate the test and production environments. When the *StockTrader* application is deployed to the test environment, it uses a DB2 database that is running locally within IBM Cloud Private. This highlights the ability of UrbanCode Deploy to work with environments that have different topologies. When the *StockTrader* application is deployed into the production environment, it uses an on-premise DB2 database that runs on the VM, named *Ubuntu16x64*. A Kubernetes secret is used in each namespace to direct *StockTrader* to the correct database for the environment it's running in.

Accessing the lab environment

Environment description

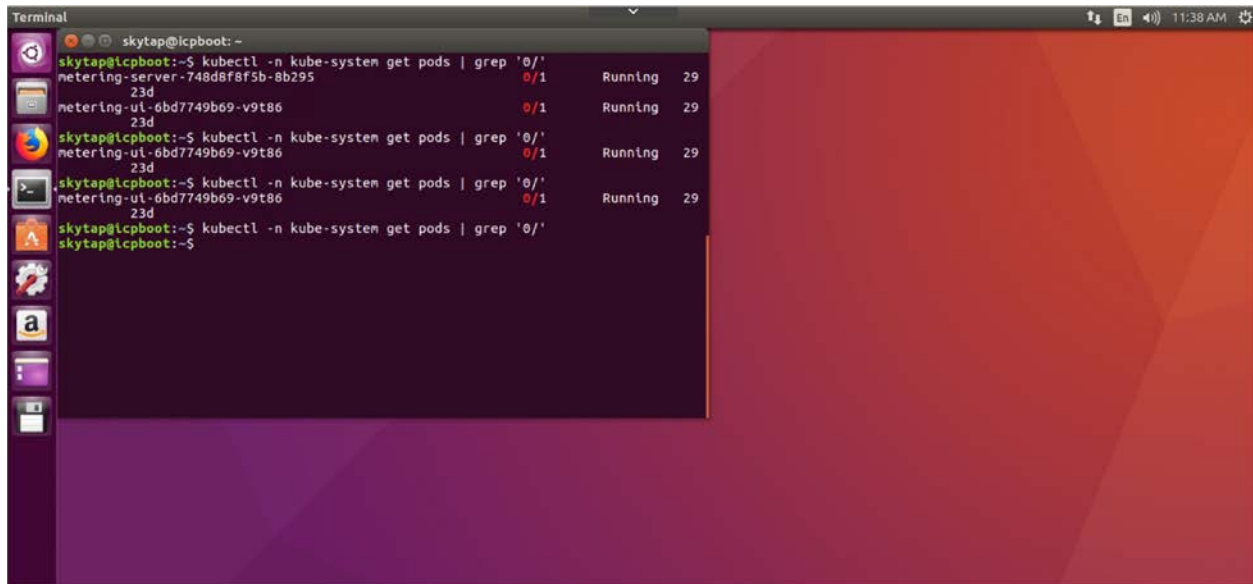
The lab environment consists of two VMs, named *Boot/Master Node* and *Ubuntu16x64*. The *Boot/Master Node* VM contains an installation of IBM Cloud Private 2.1.0.1, *kubectl*, and *helm* command-line clients, as well as an UrbanCode Deploy agent. The IBM Cloud Private cluster has two namespaces defined, *default* and *production*. These two namespaces support the two UrbanCode Deploy environments, named *TEST* and *PROD*. The IBM Cloud Private cluster contains a running instance of the UrbanCode Deploy server and various other containerized applications that support the *StockTrader* application (IBM MQ, IBM DB2, and Redis). The *Ubuntu16x64* VM contains an installation of the Rational Performance Tester workbench, IBM DB2 server, and an UrbanCode Deploy agent.

Before you begin

Ensure that all pods are running by completing the following steps:

1. Click on the **Boot/Master Node** VM. This opens the desktop for the VM.
2. Right-click on the desktop and select **Open Terminal**.

3. Run the command `kubectl -n kube-system get pods | grep '0/'` to ensure that all pods are running.



The image shows a terminal window titled 'Terminal' with the user 'skytap@lcpboot: ~'. The terminal displays the following commands and output:

```
skytap@lcpboot:~$ kubectl -n kube-system get pods | grep '0/'
metering-server-748d8f8f5b-8b295 0/1 Running 29
23d
metering-ui-6bd7749b69-v9t86 0/1 Running 29
23d
skytap@lcpboot:~$ kubectl -n kube-system get pods | grep '0/'
metering-ui-6bd7749b69-v9t86 0/1 Running 29
23d
skytap@lcpboot:~$ kubectl -n kube-system get pods | grep '0/'
metering-ui-6bd7749b69-v9t86 0/1 Running 29
23d
skytap@lcpboot:~$ kubectl -n kube-system get pods | grep '0/'
skytap@lcpboot:~$
```

You are ready to go when the command returns no output.

Exercise 1: Import a version and deploy to the test environment

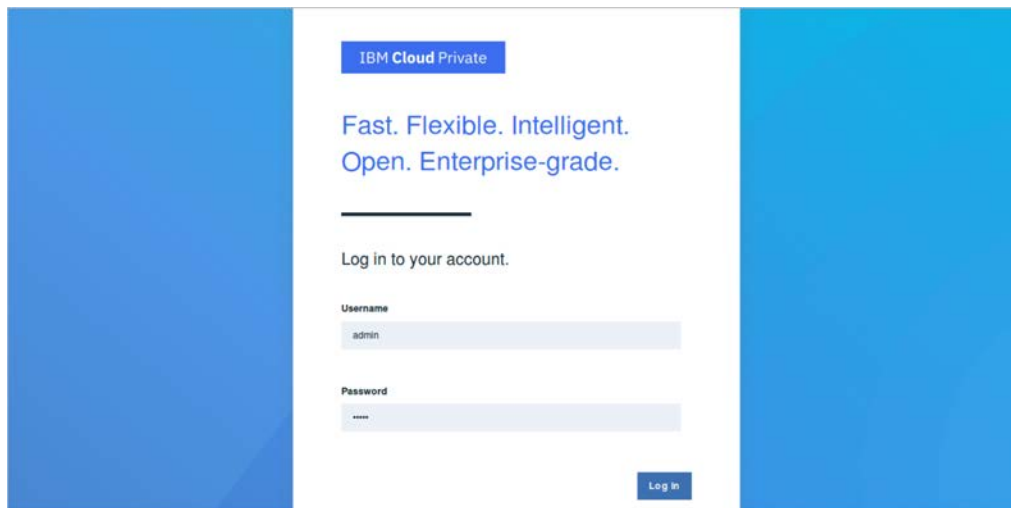
In this exercise, you complete the following tasks:

- Import the StockTrader source files into UrbanCode Deploy
- Deploy the StockTrader application to the test environment
- View the test results in Rational Performance Tester
- Verify the deployment in the web UI

View microservices in IBM Cloud Private

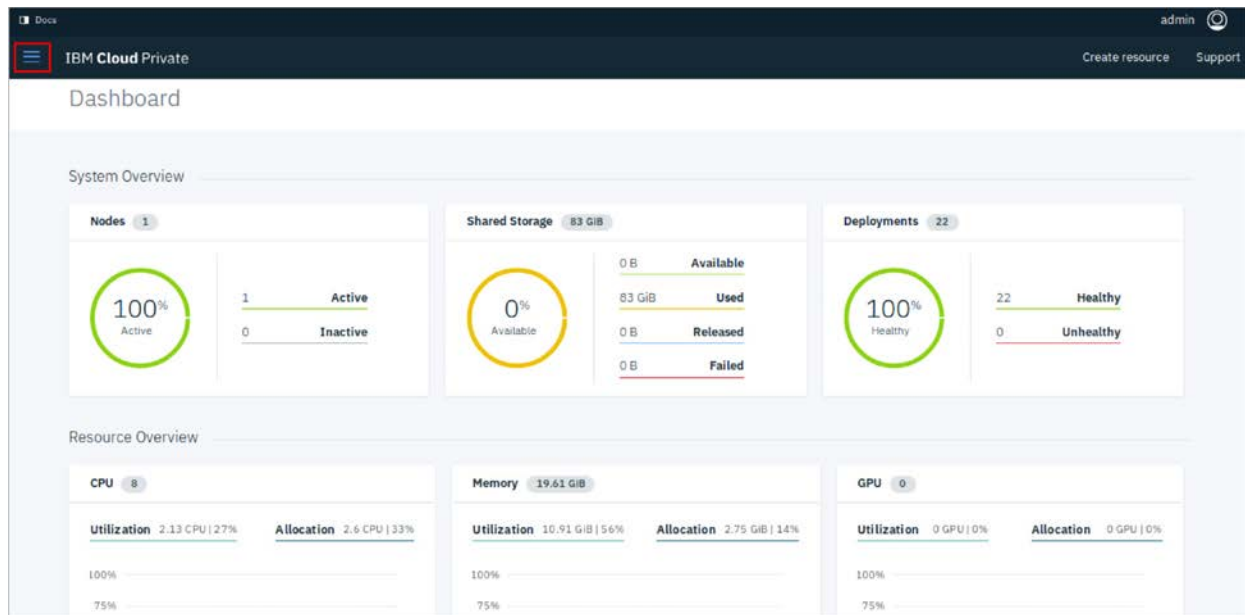
IBM Cloud Private is an application platform for developing and managing on-premises, containerized applications. It is an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image repository, a management console, and monitoring frameworks.

1. Click the **Firefox** web browser icon in the left panel of the desktop. Firefox should start with the IBM Cloud Private UI displayed. If not, click **IBM Cloud Private** on the bookmarks toolbar.
2. The username and password fields are prepopulated. Click **Log in**.

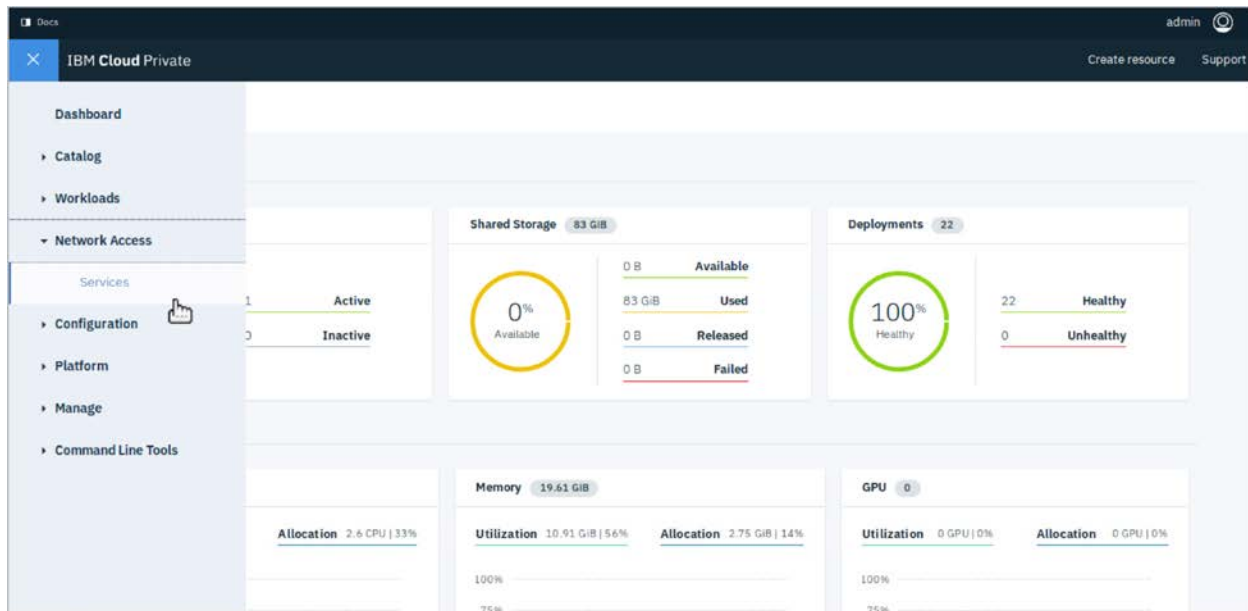


The IBM Cloud Private dashboard displays.

3. Click the **Menu** icon in the upper-left corner of the page.



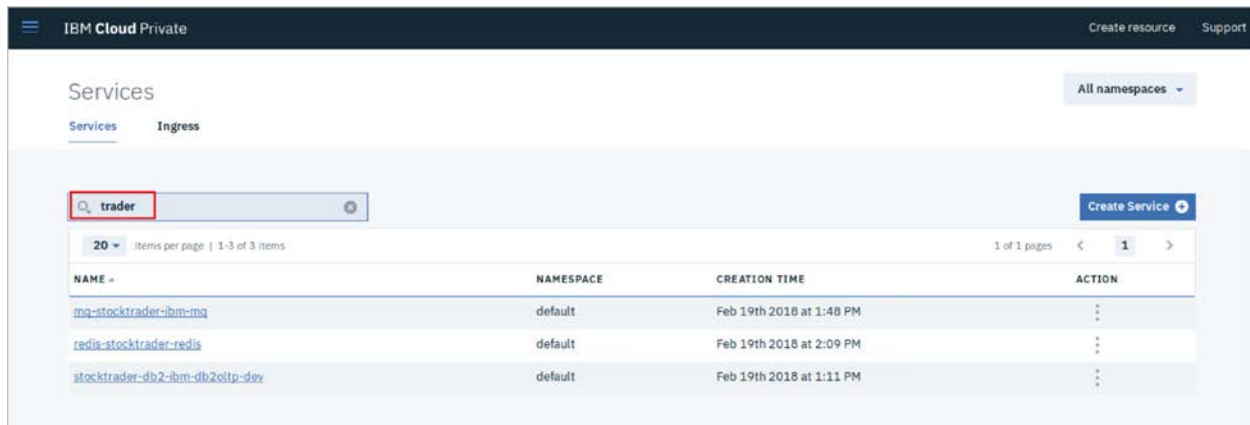
4. From the menu, click **Network Access**, and then click **Services**.



The StockTrader microservice is called *trader-service*.

5. In the **Filter** field, type `trader`.

The service is not yet there. You will come back to IBM Cloud Private after you deploy the StockTrader application.



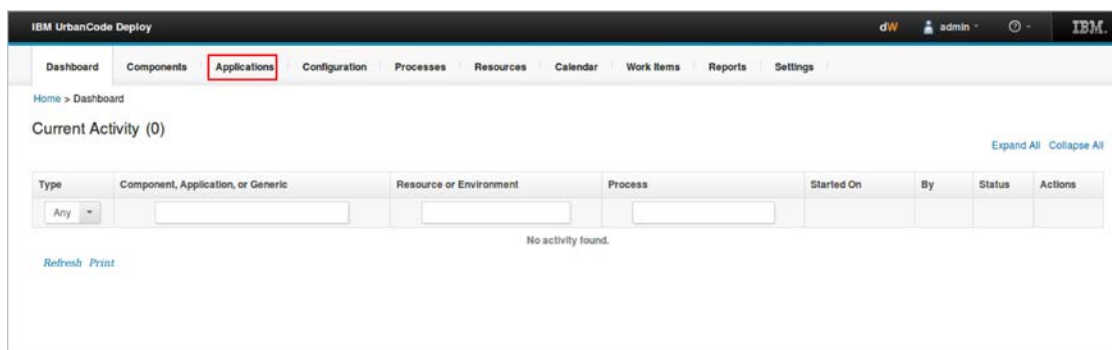
View the StockTrader application environments

An environment is the application's mechanism for bringing together components with the agent that deploys them. Environments are typically modeled on some stage of the software project lifecycle, such as development, QA, or production.

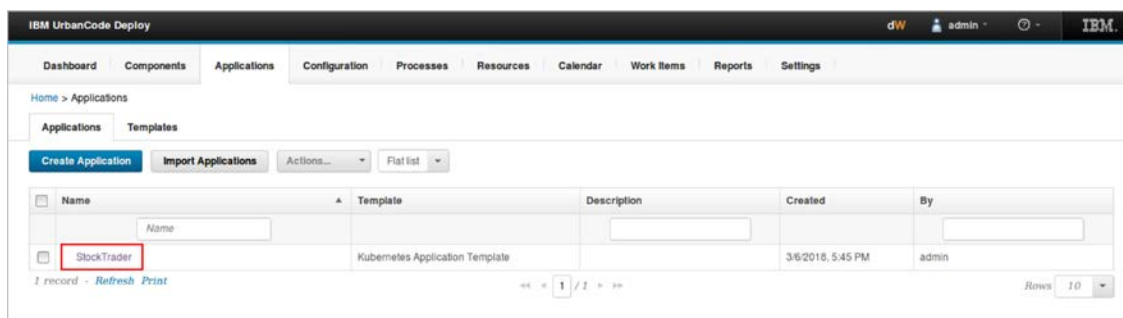
1. Open a new browser tab, and click the **IBM UrbanCode Deploy** bookmark.
2. If prompted for the username and password, type `admin` in both fields, and then click **Log in**.

The UrbanCode Deploy dashboard displays.

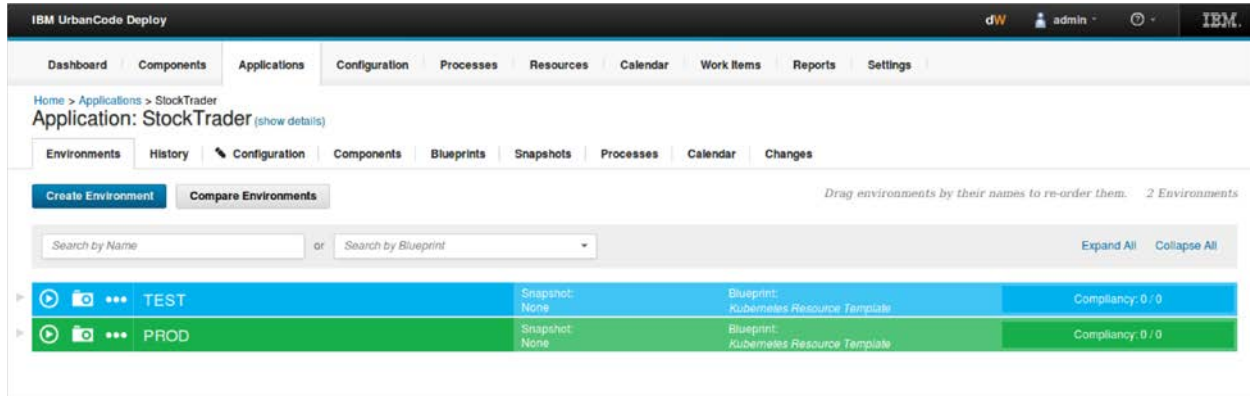
3. From the menu along the top of the page, click the **Applications** tab.



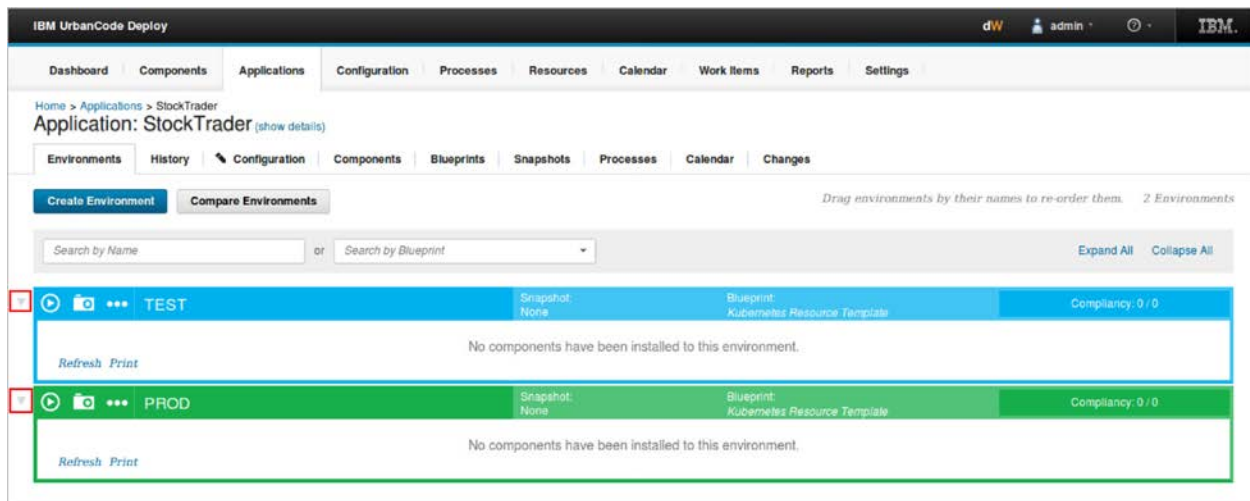
4. Click the **StockTrader** application name.



The StockTrader application has two environments: one is a test environment, the *Boot/Master Node* VM, that uses the DB2 test database in IBM Cloud Private, and the other is a production environment that uses the on-premise database, which is located on the *Ubuntu16x64* VM in this lab environment. This simulates the typical separation of test data and production data.



5. Expand the environment rows.



Currently, nothing is deployed to either environment.

6. Click the **Components** tab.

The StockTrader application contains three components: an automated test component, a database component, and the *stocktrader-all-in-one.yaml* component, which is the application resource description file used by Kubernetes to create the microservice containers.

7. Click **stocktrader-all-in-one.yaml**.

IBM UrbanCode Deploy

Dashboard Components Applications Configuration Processes Resources Calendar Work Items Reports Settings

Home > Applications > StockTrader

Application: StockTrader (show details)

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Add Component

0 Failed Version Import 0 Importing Version 2 No Version 0 Successful

Name	Actions	Last Import	Last Version	Description
AutomatedTests	Run Process Remove	No Import Data Available	NO VERSION	
Database	Run Process Remove	No Import Data Available	V1.0	Component to perform any required database operations
stocktrader-all-in-one.yaml	Run Process Remove	No Import Data Available	NO VERSION	

3 records - Refresh Print

Rows 10

8. Click the **Configuration** tab on the secondary navigation bar.

IBM UrbanCode Deploy

Dashboard Components Applications Configuration Processes Resources Calendar Work Items Reports Settings

Home > Components > stocktrader-all-in-one.yaml

Component: stocktrader-all-in-one.yaml (show details)

Dashboard Usage Configuration Calendar Versions Processes Changes

Basic Settings

Basic Settings

Name * stocktrader-all-in-one.yaml

Description

Teams +

Template Kubernetes

Template Version Always Use Latest

Component Type Standard

Version Source Configuration

Source Configuration Type Git

Repository URL * https://github.com/tpneal/Think2018-7232

Branch master

Username

Password ****

The YAML file source is coming from a GitHub repository. In the next step, you fork this repository and add your GitHub URL to the Repository URL field.

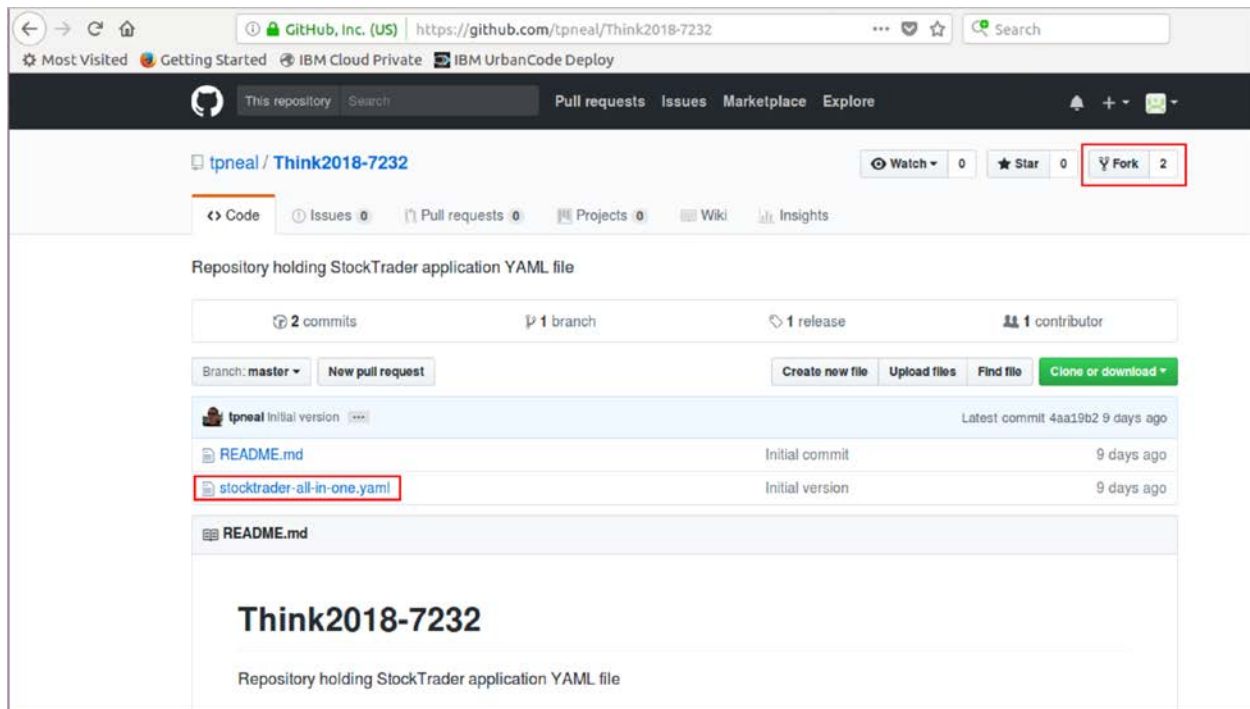
Import the StockTrader version

Every time a component's artifacts are modified and imported, a new version of the component is created.

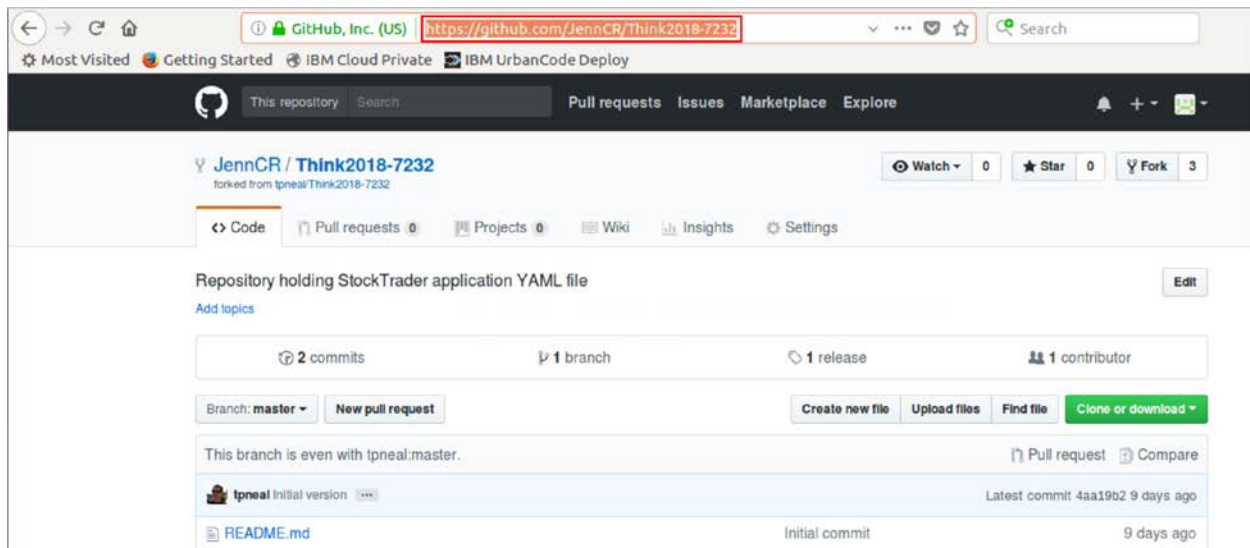
Fork the GitHub StockTrader project

1. Open a new browser tab, type `https://github.com/tpneal/Think2018-7232`, and log in with your username and password. If you don't already have a GitHub account, you need to sign-up for a free account before you can proceed.

2. In the upper-right corner, click **Fork**.



3. Copy your URL for the Think2018-7232 project.



Update the configuration and import the new version

1. Click the **IBM UrbanCode Deploy** tab, replace the URL that is in the **Repository URL** field with your URL, and click **Save**.

IBM UrbanCode Deploy

Dashboard Components Applications Configuration Processes Resources Calendar Work Items Reports Settings

Home > Components > stocktrader-all-in-one.yaml (show details)

Component: stocktrader-all-in-one.yaml

Dashboard Usage Configuration Calendar Versions Processes Changes

Basic Settings

Component Properties
Environment Property Definitions
Resource Property Definitions
Version Property Definitions
Configuration File Templates
Version Import History

Basic Settings

Name * stocktrader-all-in-one.yaml

Description

Teams +

Template Kubernetes

Template Version Always Use Latest

Component Type Standard

Version Source Configuration

Source Configuration Type Git

Repository URL * https://github.com/JennCR/Think2018-7232

Branch master

Username

Password ****

- Click the **Versions** tab, and then click **Import New Versions**.

IBM UrbanCode Deploy

Dashboard Components Applications Configuration Processes Resources Calendar Work Items Reports Settings

Home > Components > stocktrader-all-in-one.yaml (show details)

Component: stocktrader-all-in-one.yaml

Dashboard Usage Configuration Calendar Versions Processes Changes

Import New Versions

Version	Statuses	Type	Created By	Date
	Statuses	Any		

No versions found.

Refresh Print

- In the *Import New Versions* window, leave the fields blank, and click **Save**.
After a few moments, version 1.0 of the YAML file is imported. (You might need to click **Refresh**.)
- Click **V1.0** to see what files are in this version.

Dashboard Usage Configuration Calendar Versions Processes Changes

Import New Versions

Version	Statuses	Type	Created By	Date
V1.0	Statuses	Any		
		Full	UC Version Import	3/11/2018, 12:00 PM

1 record - Refresh Print

1 / 1

The only artifact in this version is the application description YAML file.

IBM UrbanCode Deploy

Home > Components > stocktrader-all-in-one.yaml > Versions > Version: V1.0

Version: V1.0 (show details)

Main Configuration History

Statuses

Add a Status

Status	Description	Created	By	Actions
No statuses have been assigned to this version.				

Refresh

Artifacts

Total: 7.7 KB (1 files)

Download All

Expand All Collapse All

Name	Size	Last Modified	Actions
stocktrader-all-in-one.yaml	7.7 KB	3/11/2018, 12:00 PM	Download

Refresh Print

Deploy the application to the test environment

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

Choose the component versions and run the application process

Now that you have a new version of the YAML file, deploy it to the test environment.

1. Click **Applications**, and then click **StockTrader**.
2. Click the play button to the left of the TEST environment:

IBM UrbanCode Deploy

Home > Applications > StockTrader

Application: StockTrader (show details)

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Create Environment Compare Environments

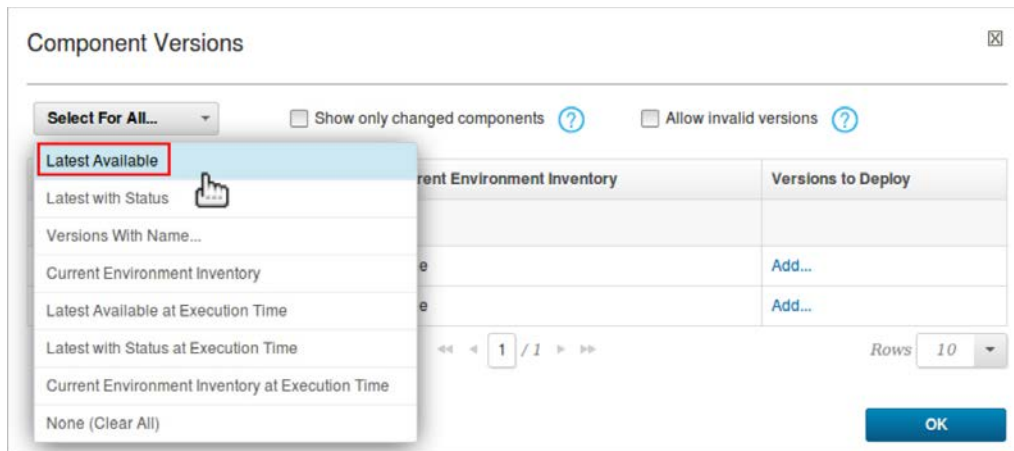
Drag environments by their names to re-order them. 2 Environments

Search by Name or Search by Blueprint

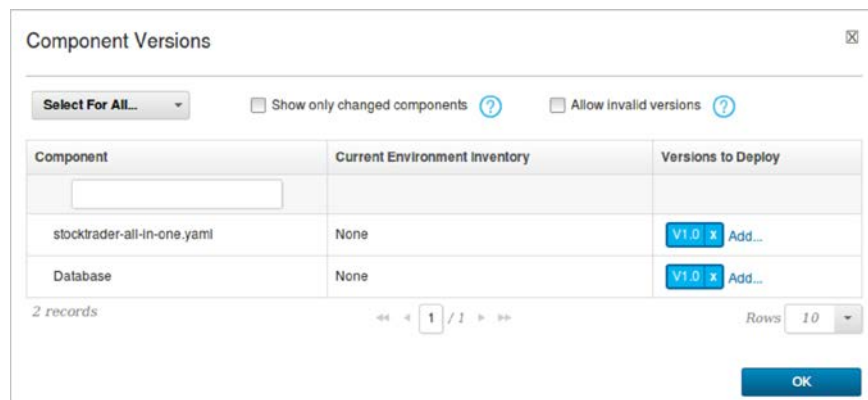
Expand All Collapse All

TEST	Snapshot: None	Blueprint: Kubernetes Resource Template	Compliance: 0 / 0
PROD	Snapshot: None	Blueprint: Kubernetes Resource Template	Compliance: 0 / 0

3. In the *Run Process on TEST* window, click **Choose Versions**.
4. In the *Component Versions* window, click **Select For All**, and then click **Latest Available**.



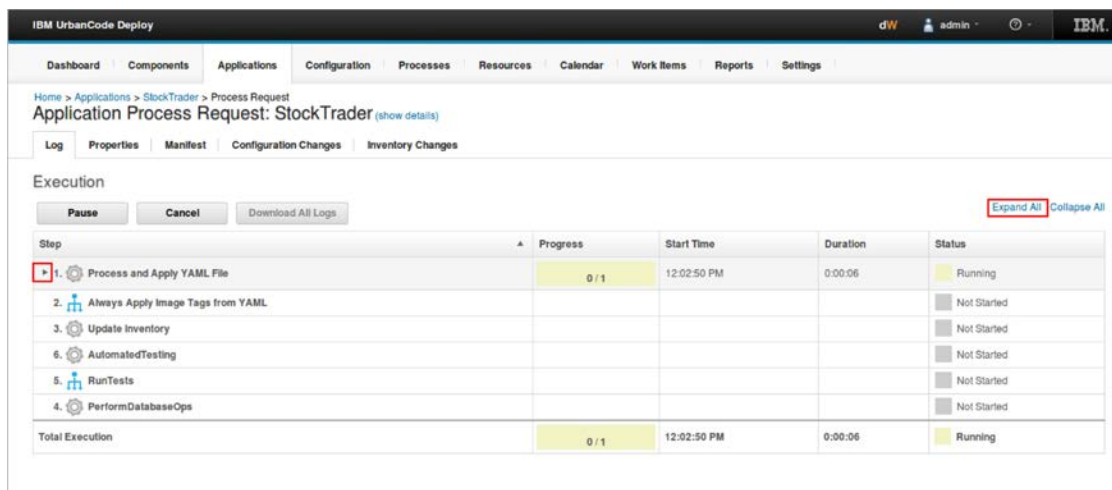
The latest versions are added for deployment.



5. Click **OK**, and then click **Submit**.

When an application process starts, the *Application Process Request* page displays information about the application's status and provides links to logs and the application manifest.

6. Expand the steps with the row arrows, or you can click **Expand All**.



Observe the application process request

This deployment takes several minutes to complete. Here is the breakdown of the steps that you can observe while it is running.

Note: Steps might display out of order, but they still process in the correct order.

Step 1: Process and Apply YAML File

This step first cleans the working directory and downloads the YAML file, which is the application description file that Kubernetes uses. Then, it replaces any tokens that are defined in the YAML file (none at the point) with the values from the UrbanCode Deploy environment properties. Next, it processes the YAML file by looking for image specifications and creating the child components associated with the images based on the image specifications of the YAML file. Finally, the `kubectl` command applies the resources into the cluster, and the StockTrader application is loaded into IBM Cloud Private.

Step 2: Always Apply Image Tags From YAML

This switch step works in concert with the corresponding field in the *Process YAML File* step from the Kubernetes plugin. The plugin step is invoked by the application step above. It allows users to decide whether the application description YAML file is the “source of truth” for image tags and versions, or if values specified by the user when doing a deployment are the “source of truth” for the deployment. Thus, if an UrbanCode Deploy environment property, named `alwaysApplyVersionsFromYaml`, has a value of `TRUE`, `true`, or `True`, then the application process ignores the image tags and version specified in the UrbanCode Deploy user interface and only uses the values specified explicitly in the application description YAML file. By default, the environment property, `alwaysApplyVersionsFromYaml`, does not exist, so the image tags and versions specified in the UrbanCode Deploy user interface is used.

Execution					
<div>Pause</div> <div>Cancel</div> <div>Download All Logs</div>			<div>Expand All</div> <div>Collapse All</div>		
Step	Progress	Start Time	Duration	Status	
▼ 1. Process and Apply YAML File	1 / 1	12:02:50 PM	0:00:40	Success	
▼ stocktrader-all-in-one.yaml	1 / 1	12:02:50 PM	0:00:40	Success	
▼ Kubernetes	1 / 1	12:02:50 PM	0:00:40	Success	
▼ Process and Apply YAML File (stocktrader-all-in-one.yaml V1.0)	11	12:02:50 PM	0:00:40	Success	
1. Clean working directory	11	12:02:51 PM	0:00:06	Success	
2. Download Artifacts	11	12:02:57 PM	0:00:07	Success	
3. Replace Tokens	11	12:03:03 PM	0:00:04	Success	
4. Process YAML File	11	12:03:07 PM	0:00:18	Success	
5. Apply Resources	11	12:03:25 PM	0:00:05	Success	
2. Always Apply Image Tags from YAML	11	12:03:30 PM	0:00:00	Success	

Step 3: Update inventory

This step updates the inventory in UrbanCode Deploy and shows what versions of the images are in the environment. These subcomponents are all created automatically, based on the specifications in the YAML file. The names are a combination of the application name plus the name of the image.

▼ 3. Update Inventory	0 / 0	12:03:30 PM	0:00:00	Success
▼ StockTrader-loyalty-level	0 / 0	12:03:30 PM	0:00:00	Success
StockTrader-loyalty-level		12:03:30 PM	0:00:00	No Version Selected
▶ StockTrader-notification	0 / 0	12:03:30 PM	0:00:00	Success
▶ StockTrader-portfolio	0 / 0	12:03:30 PM	0:00:00	Success
▶ StockTrader-stock-quote	0 / 0	12:03:30 PM	0:00:00	Success
▶ StockTrader-trader	0 / 0	12:03:30 PM	0:00:00	Success

Step 4: Perform Database Ops

This step performs the operations on the database that StockTrader is using. It creates an index on the portfolio table of the database to ensure that subsequent searches run faster. It downloads a SQL file and the DB2 JAR file that's needed to execute DB2 commands, and then it runs SQL scripts from the SQL JDBC plugin to create the index.

You can click the **Output Log** icon of the *Execute SQL Scripts* step to view the output log.

▼ 4. PerformDatabaseOps	1 / 1	12:03:30 PM	0:00:29	Success
▼ Database	1 / 1	12:03:30 PM	0:00:29	Success
Database	1 / 1	12:03:30 PM	0:00:29	Success
▼ CreateOwnerIndex (Database V1.0)	::	12:03:30 PM	0:00:29	Success
1. Download Artifacts		12:03:31 PM	0:00:08	Success
2. Execute SQL Scripts		12:03:40 PM	0:00:21	Success

In the output log, notice the property named, *connectionString*. This property is used to determine the database to connect to.

Output Log

```
Working Directory /opt/ibm-ucd/agent/var/work/Database
1 =====
2 plugin: SQL-JDBC, id: com.urbancode.air.plugin.sql.JDBC, version: 10
3 plugin command: '/opt/ibm-ucd/agent/opt/groovy-1.8.8/bin/groovy' '-cp' '/opt/ibm-ucd/agent
4 working directory: /opt/ibm-ucd/agent/var/work/Database
5 properties:
6   PLUGIN_INPUT_PROPS=/opt/ibm-ucd/agent/var/temp/logs4355694035985025631/input.props
7   PLUGIN_OUTPUT_PROPS=/opt/ibm-ucd/agent/var/temp/logs4355694035985025631/output.props
8   autocommit=false
9   connectionString=jdbc:db2://10.0.0.1:31188/trader
10  delimiter=,
11  excludes=
12  files=
```

The value of this property comes from an environment property named *dbConnectionURL*. In the TEST environment, it is configured to point to the DB2 database running in IBM Cloud Private.

The screenshot shows the IBM UrbanCode Deploy interface. The breadcrumb navigation is [Home](#) > [Applications](#) > [StockTrader](#) > [Environments](#) > Environment: TEST. The page title is "Environment: TEST for StockTrader (show details)". The left sidebar has "Basic Settings" and "Environment Properties" (selected). The main content area is titled "Environment Properties" and shows "Version 5 of 5". There are "Add Property" and "Batch Edit" buttons. A table lists properties:

Name	Value
dbConnectionURL	jdbc:db2://10.0.0.1:31188/trader
KUBECTL-OPTS	--kubeconfig=/home/skytap/kube/config --context=bluedemocluster.icp-context --namespace=default










In the PROD environment, it is configured to point to the on-premise database with our production data.

The screenshot shows the IBM UrbanCode Deploy interface for the PROD environment. The breadcrumb navigation is [Home](#) > [Applications](#) > [StockTrader](#) > [Environments](#) > Environment: PROD. The page title is "Environment: PROD for StockTrader (show details)". The left sidebar has "Basic Settings" and "Environment Properties" (selected). The main content area is titled "Environment Properties" and shows "Version 4 of 4". There are "Add Property" and "Batch Edit" buttons. A table lists properties:

Name	Value
dbConnectionURL	jdbc:db2://10.0.0.2:50000/trader
KUBECTL-OPTS	--kubeconfig=/home/skytap/kube/config --context=bluedemocluster.icp-context --namespace=production

Step 5: Run tests

This step is a switch that checks if the current environment is named TEST. If it is the TEST environment, the process branches and runs a Rational Performance Tester schedule against the StockTrader application deployed to the TEST environment; otherwise, it skips the test.

5.  RunTests	12:03:56 PM	0:00:00	 Success	
▼ 4.  PerformDatabaseOps	1 / 1	12:03:30 PM	0:00:25	 Success
▼  Database	1 / 1	12:03:30 PM	0:00:25	 Success
▼  Database	1 / 1	12:03:30 PM	0:00:25	 Success
▼  CreateOwnerIndex (Database V1.0)	12:03:30 PM	0:00:25	 Success	
1.  Download Artifacts	12:03:31 PM	0:00:09	 Success	
2.  Execute SQL Scripts	12:03:40 PM	0:00:16	 Success	

Step 6: Automated Testing

This step takes about four minutes. It includes a Rational Performance Test, where five users connect to the StockTrader database and verify that they can execute various requests against the application, such as viewing, querying, and updating portfolios. If the test completes successfully, it records the current date and time and creates an UrbanCode snapshot of the TEST environment.

A snapshot consists of all the information about the deployed environment, such as the YAML, database component, and image versions. A snapshot can be used in subsequent deployments, as it includes components that are already tested and validated together.

▼ 6. AutomatedTesting	0 / 1	12:03:56 PM	0:00:35	Running
▼ AutomatedTests	0 / 1	12:03:56 PM	0:00:35	Running
▼ AutomatedTests	0 / 1	12:03:56 PM	0:00:35	Running
▼ RunPerfTests (/ StockTrader-1 / TEST / TestServer / apphost / AutomatedTests)		12:03:56 PM	0:00:35	Running
1. Run RPT Test		12:03:56 PM	0:00:35	Running
2. GetDateTime				Not Started
3. Create Snapshot Of Environment				Not Started

View the test results in Rational Performance Tester

Rational Performance Tester captures the network traffic that is rendered when the application under test interacts with a server. This network traffic is then emulated on multiple virtual users while playing back the test.

1. From the Skytap UI, right-click the **Ubuntu16x64** VM, and select **Open Link in New Tab**.
2. When prompted with the password screen, click **ibmuser**, and type `passw0rd` as the password. The desktop displays.
3. Right-click the desktop, and select **Open Terminal**.
4. At the command prompt run these two commands:

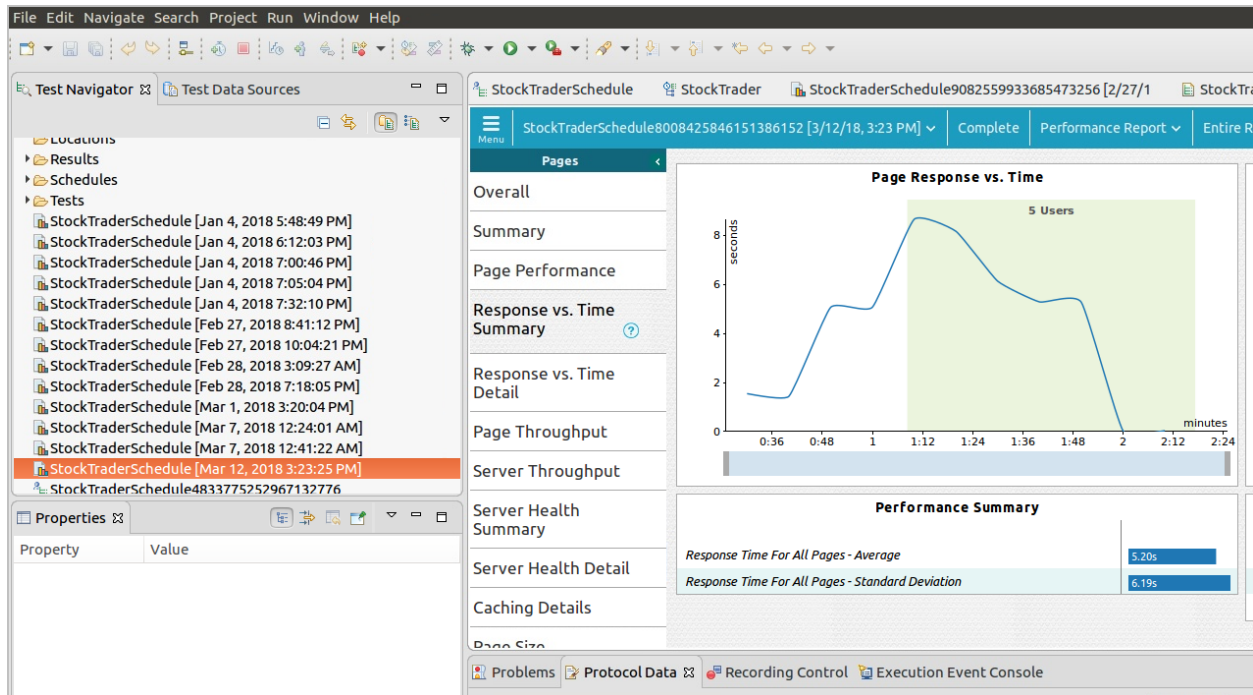
```
. .bash_profile
eclipse &
```

The Rational Performance Tester workbench starts.

5. In the *Select a directory as a workspace* window, click **OK**. The workbench interface is displayed.
6. Find the most recent *StockTraderSchedule* file run under the **Test** folder in the *Test Navigator*. Then, double-click it to open it.

You can click the various reports pages, such as Summary and Page Performance, to see information about the performance test run.

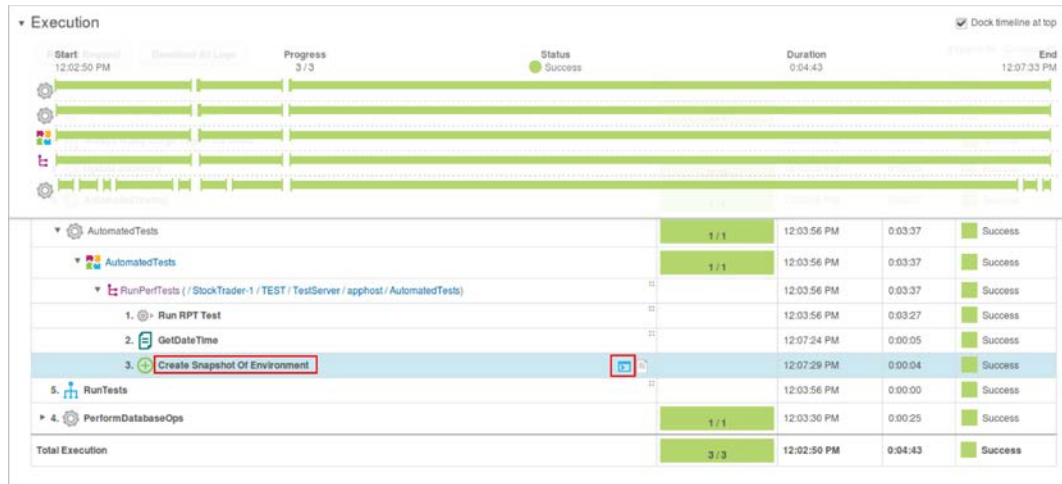
Note: When you are done looking at the results, be sure to select **File > Exit** to exit the Rational Performance Tester workbench; otherwise, when UrbanCode Deploy attempts to do another test run, it will fail because the workbench is already open or busy.



Locate the snapshot and view the deployed components

After the deployment is complete, and the Rational Performance Tester schedule has passed all tests, it creates a *snapshot*. A snapshot is a collection of specific versions of components and processes. Typically, a snapshot represents a set of component versions that are known to work together. In most cases, snapshots include all the components in an application.

1. Click the **UrbanCode Deploy** tab to return to the deployment.
2. Click the **Output Log** icon to open the output log.



The output log shows that the snapshot was created:

Output Log

```

Working Directory /opt/ibm-ucd/agent/var/work/AutomatedTests
1
2 plugin: IBM UrbanCode Deploy Applications, id: com.urbanocode.air.plugin.uDeployApplication, version: 77
3 plugin command: '/opt/ibm-ucd/agent/opt/groovy-1.8.8/bin/groovy' '-cp' '/opt/ibm-ucd/agent/var/plugins/co
4 working directory: /opt/ibm-ucd/agent/var/work/AutomatedTests
5 properties:
6   PLUGIN_INPUT_PROPS=/opt/ibm-ucd/agent/var/temp/logs3631994658560256519/input.props
7   PLUGIN_OUTPUT_PROPS=/opt/ibm-ucd/agent/var/temp/logs3631994658560256519/output.props
8   application=StockTrader
9   description=All performance tests passed
10  environment=TEST
11  name=TestsPassed-2018-03-11T19:07:29.314+0000
12 environment:
13   AGENT_HOME=/opt/ibm-ucd/agent
14   AH_AUTH_TOKEN=****
15   AH_WEB_URL=https://10.0.0.1:30386/
16   AUTH_TOKEN=****
17   DS_AUTH_TOKEN=****
18   DS_SYSTEM_ENCODING=UTF-8
19   JAVA_OPTS=-Dfile.encoding=UTF-8 -Dconsole.encoding=UTF-8
20   PLUGIN_HOME=/opt/ibm-ucd/agent/var/plugins/com.urbanocode.air.plugin.uDeployApplication_77_f8343362996bc
21   UCD_USE_ENCRYPTED_PROPERTIES=true
22   UD_DIALOGUE_ID=0a87fd1e-5a61-4e90-alc8-8295788c8a3e
23   WE_ACTIVITY_ID=1621672a-cadc-2efa-a035-34dbfbd938c6
24
25
26 Snapshot TestsPassed-2018-03-11T19:07:29.314+0000 created.
27
  
```

3. Close the output log.
4. Click the **StockTrader** breadcrumb link, and then click the **Snapshots** tab.

The snapshot is listed.

Snapshot	Description	Created	By	Actions
TestsPassed-2018-03-11T19:07:29.314+0000	All performance tests passed	3/11/2018, 12:07 PM	admin	Export Compare Edit Delete

- Open the snapshot by clicking the name.
- Click the **Component Versions** tab. All the component versions are listed.

IBM UrbanCode Deploy

Home > Applications > StockTrader > Snapshots > Snapshot: TestsPassed-2018-03-11T19:07:29.314+0000

Snapshot: TestsPassed-2018-03-11T19:07:29.314+0000 (show details)

Dashboard Component Versions Configuration Calendar

Select For All... Revert Lock Versions

Component	Versions
AutomatedTests	Add...
Database	V1.0 Add...
StockTrader-loyalty-level	latest Add...
StockTrader-notification	latest Add...
StockTrader-portfolio	latest Add...
StockTrader-stock-quote	latest Add...
StockTrader-trader	latest Add...
stocktrader-all-in-one.yaml	V1.0 Add...

8 records 1 / 1 Rows 10

- To see the deployed components, return to the **Environments** tab by clicking the **StockTrader** breadcrumb link.
- Expand the TEST environment.

Notice that versions 1 for the database and YAML file were deployed. Also notice that five StockTrader images components were created. The Kubernetes plugin step, *Process YAML File*, created them automatically based on the image specifications found in the application description YAML file.

IBM UrbanCode Deploy

Home > Applications > StockTrader

Application: StockTrader (show details)

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Create Environment Compare Environments

Drag environments by their names to re-order them. 2 Environments

Search by Name or Search by Blueprint Expand All Collapse All

TEST Snapshot: TestsPassed-2018-03-11T19:07:29 Blueprint: Kubernetes Resource Template Compliance 7/7

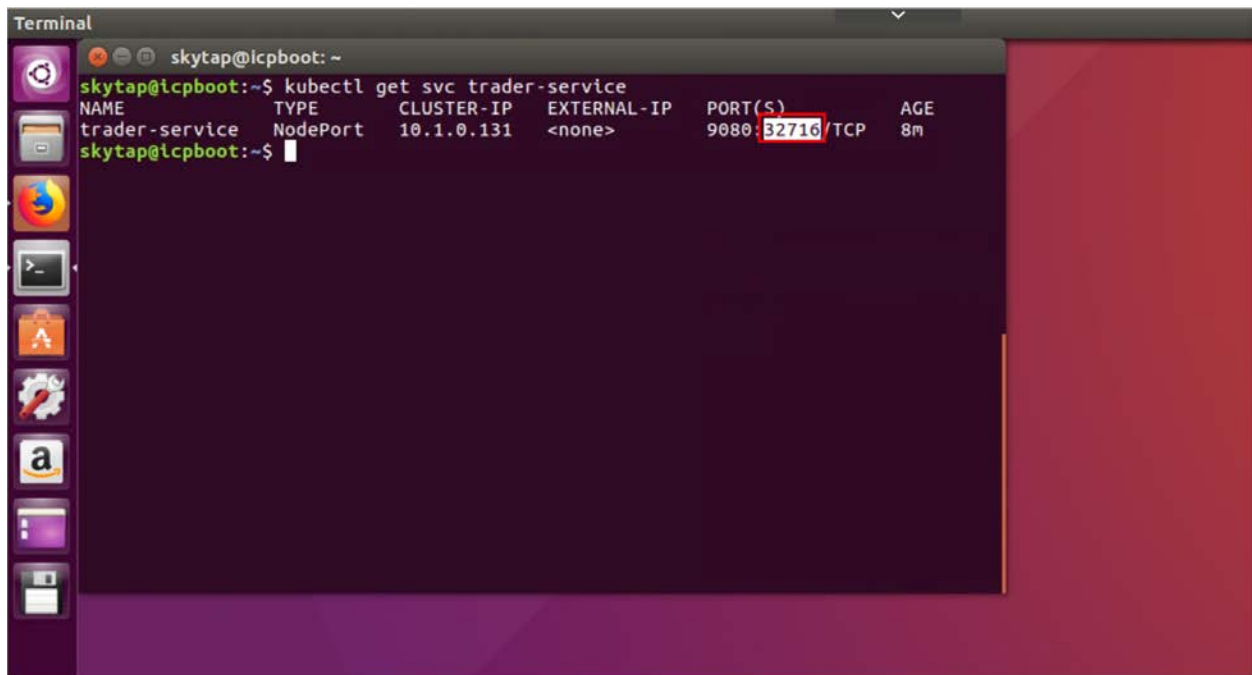
Component	Version	Date Deployed	Compliance	Actions
StockTrader-trader	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-stock-quote	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-portfolio	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-notification	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-loyalty-level	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
Database	V1.0 (View Details)	3/11/2018, 12:02 PM	Compliant (1/1)	View Request
stocktrader-all-in-one.yaml	V1.0 (View Details)	3/11/2018, 12:02 PM	Compliant (1/1)	View Request

Refresh Print

Verify the application in the test environment

Now that the StockTrader application is deployed to the test environment, you can view the web UI that's in IBM Cloud Private. First, you need retrieve the port number for the trader-service.

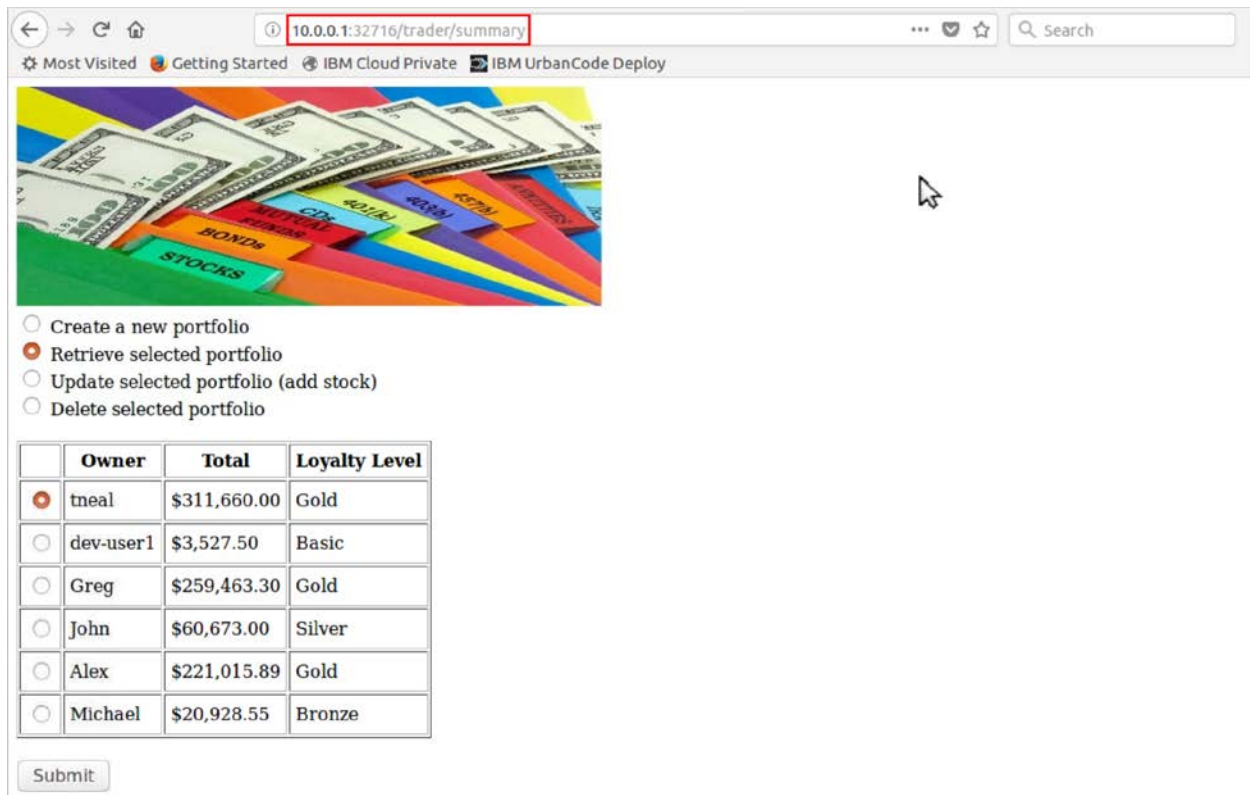
1. Open the command-line prompt in the Boot/Master Node VM, and type `kubectl get svc trader-service`, and press Enter. This command finds the port that the StockTrader application is listening at.
2. Copy the port number.



```
Terminal
skytap@lcpboot: ~
skytap@lcpboot:~$ kubectl get svc trader-service
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
trader-service NodePort    10.1.0.131   <none>        9080:32716/TCP   8m
skytap@lcpboot:~$
```

3. Open a new browser tab, and type the URL `http://10.0.0.1:<port number>/trader/summary`, pasting your port number in the appropriate spot. For example, `http://10.0.0.1:32716/trader/summary`.
4. Press Enter.

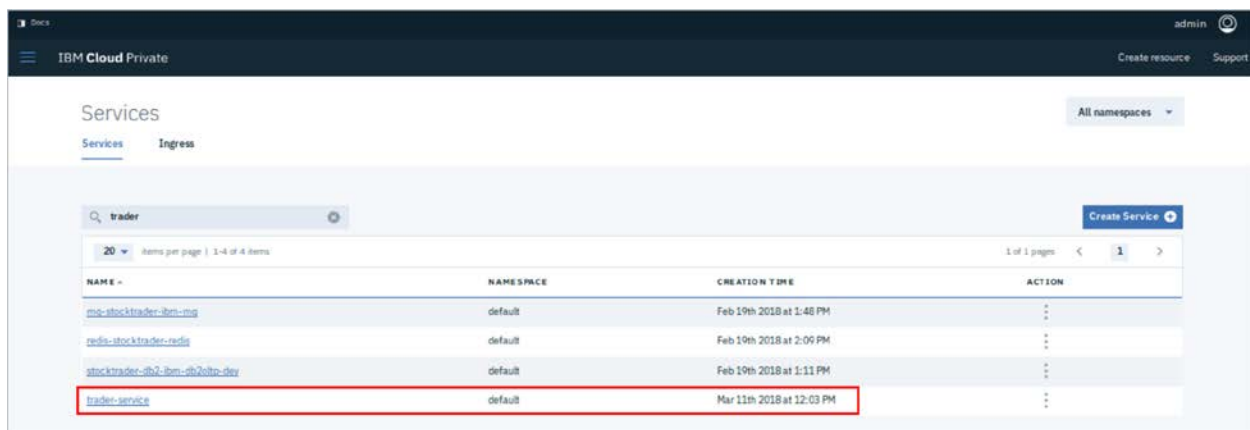
The StockTrader application displays in the browser. Test the application by creating a portfolio, adding stock to an account, and retrieving portfolio information.



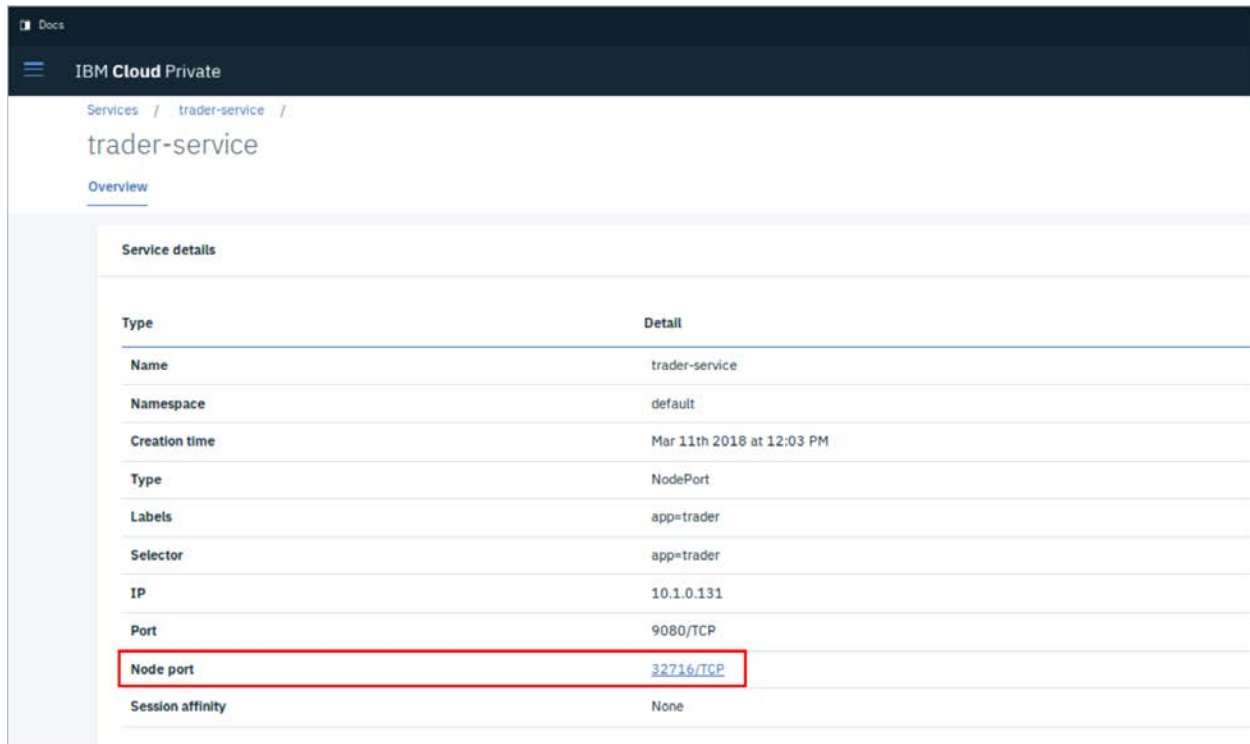
[View the microservice in IBM Cloud Private](#)

Return to IBM Cloud Private to see the StockTrader microservice.

1. Click the **IBM Cloud Private** tab.
2. If needed, from the menu, click **Network Access**, and then click **Services**.
3. In the **Filter** field, type `trader`. The trader service now displays for the default/test namespace.
4. Click **trader-service**.



Notice the **Node port** field. This is the same port number that you retrieved with the `kubectl` command.



The screenshot shows the IBM Cloud Private console interface. At the top, there's a dark blue header with 'IBM Cloud Private' and a navigation menu. Below the header, the breadcrumb 'Services / trader-service /' is visible, followed by the title 'trader-service' and a sub-tab 'Overview'. The main content area is titled 'Service details' and contains a table with the following data:

Type	Detail
Name	trader-service
Namespace	default
Creation time	Mar 11th 2018 at 12:03 PM
Type	NodePort
Labels	app=trader
Selector	app=trader
IP	10.1.0.131
Port	9080/TCP
Node port	32716/TCP
Session affinity	None

The 'Node port' row is highlighted with a red rectangular border, and the value '32716/TCP' is a blue hyperlink.

Exercise 2: Use properties to set the deployment environment

In this exercise, you complete the following tasks:

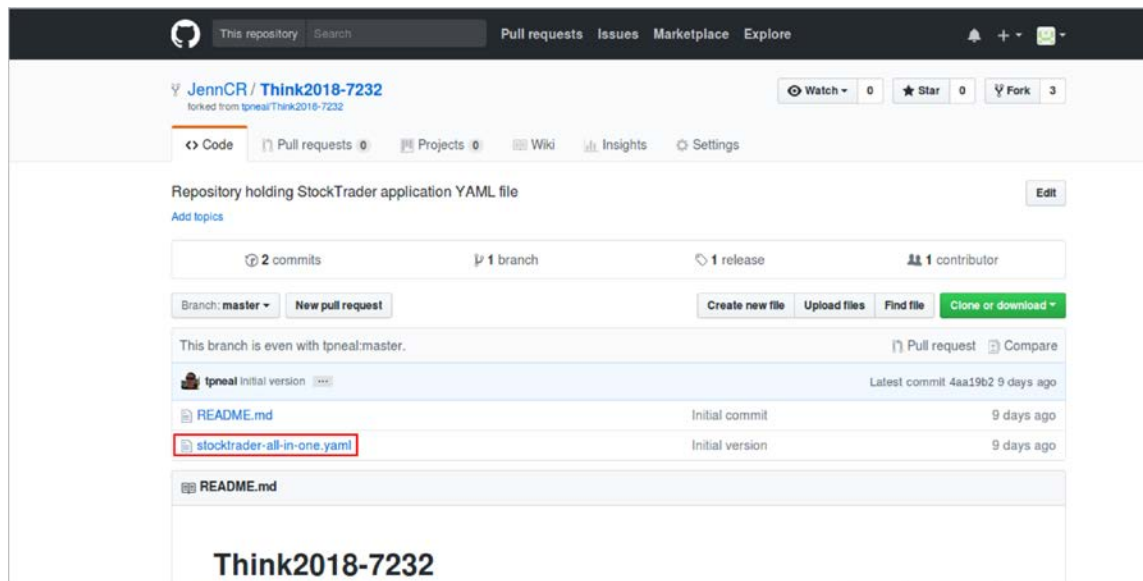
- Modify the application description YAML file to use tokens for the registry host and repository part of the image specification
- Create a new version and publish a release in GitHub
- Deploy the parameterized file to the test environment

Create a new version and publish a release

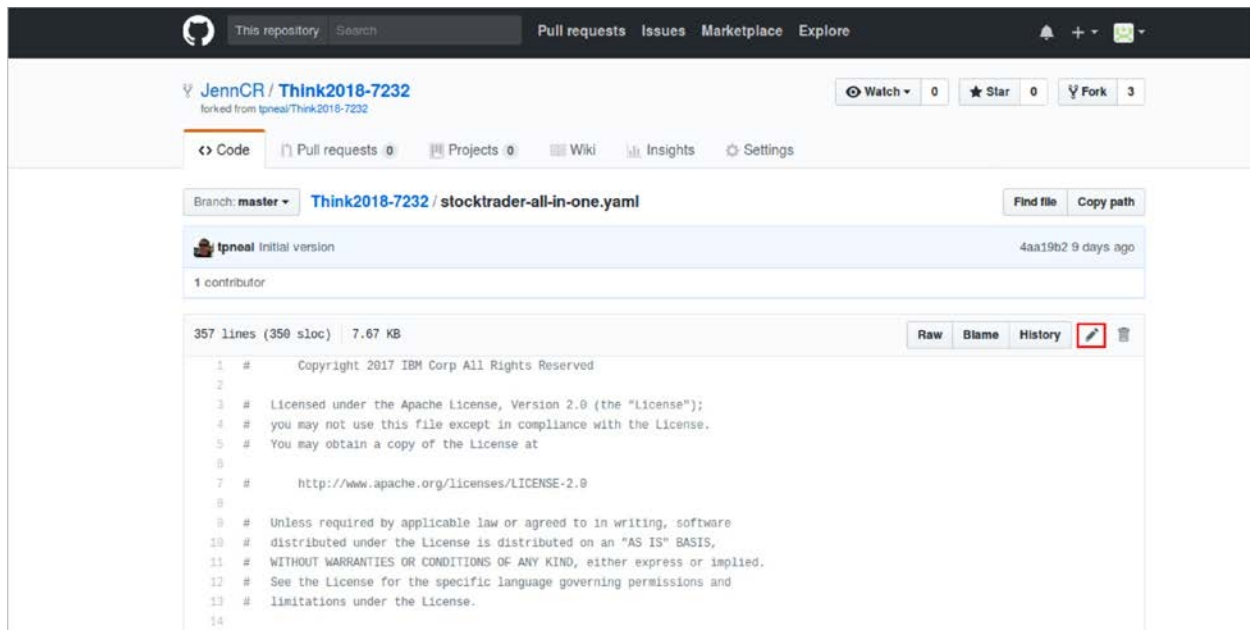
After you modify the application description YAML file in GitHub, you publish a release with those changes. Then, you import the new version into UrbanCode Deploy.

Replace hard-coded environment specifications with tokens

1. Click the **GitHub** tab with your open Think2018-7232 project, and click the **stocktrader-all-in-one.yaml** file.



2. Click the **Edit** icon in the right corner of the open document.



For each of the five StockTrader images, you substitute the registry host and repository with a token. Adding a token parameterizes the application description YAML file, which makes it simpler to deploy to different environments without changing the YAML file each time. Instead of hard-coding the image specification, UrbanCode Deploy can pass parameters from the environment to identify where to pull the image from.

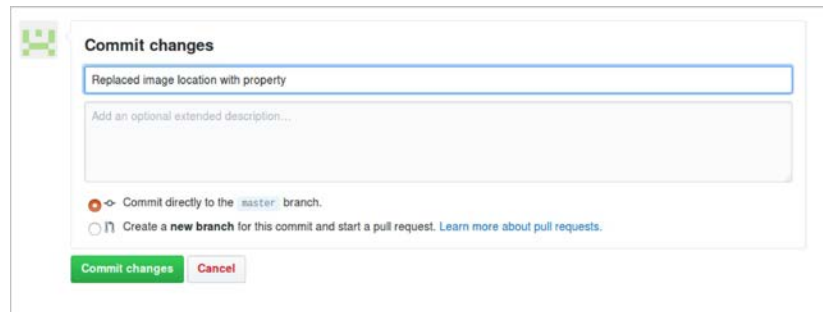
3. For every instance of `bluedemocluster.icp:8500/default` that you find in the YAML file, replace it with `@registryHostAndRepo@`.

```

7 # http://www.apache.org/licenses/LICENSE-2.0
8
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the license for the specific language governing permissions and
13 # limitations under the License.
14
15 #Deploy the pod
16 apiVersion: extensions/v1beta1
17 kind: Deployment
18 metadata:
19   name: loyalty-level
20 spec:
21   replicas: 1
22   template:
23     metadata:
24       labels:
25         app: loyalty-level
26     spec:
27       containers:
28       - name: loyalty-level
29         image: @registryHostAndRepo@/loyalty-level:latest
30         env:
31         - name: MQ_ID
32           valueFrom:
33             secretKeyRef:
34               name: mq
35               key: id

```

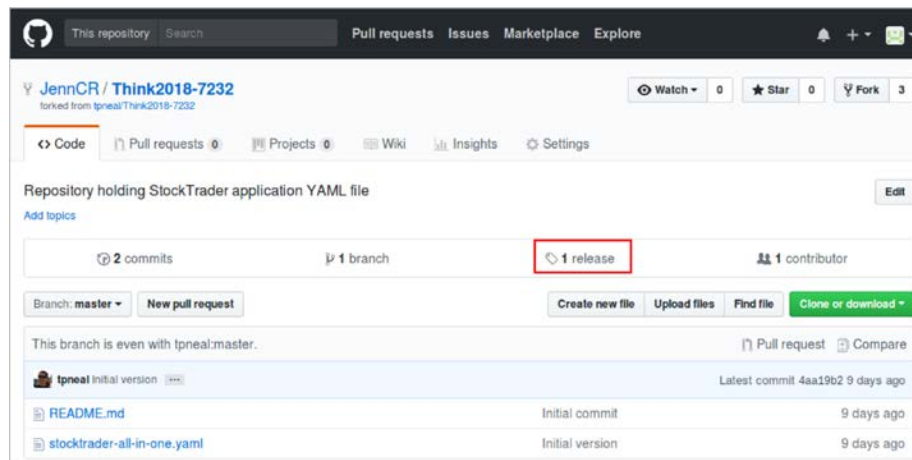
4. In the *Commit changes* section at the bottom of the page, type a comment in the field, and click **Commit changes**.



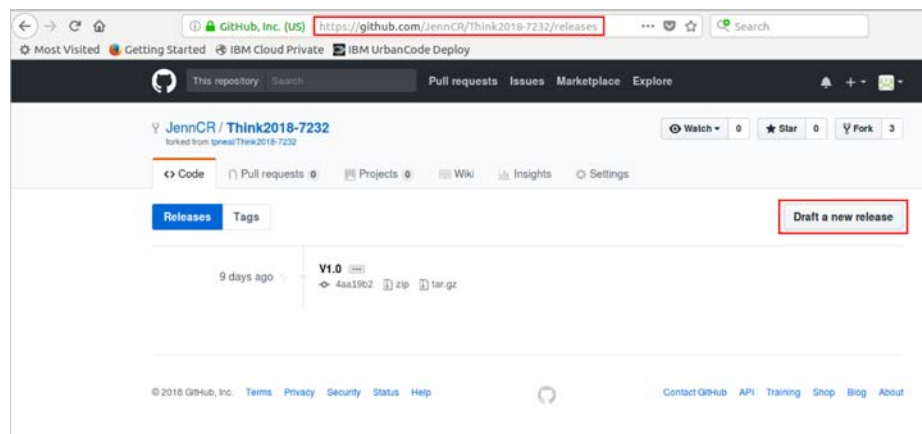
The screenshot shows the 'Commit changes' dialog box in GitHub. It has a title bar with the GitHub logo and the text 'Commit changes'. Below the title bar is a text input field containing the text 'Replaced image location with property'. Below the input field is a larger text area with the placeholder text 'Add an optional extended description...'. Below the text area are two radio buttons. The first radio button is selected and is labeled 'Commit directly to the master branch.'. The second radio button is labeled 'Create a new branch for this commit and start a pull request. Learn more about pull requests.'. At the bottom of the dialog box are two buttons: 'Commit changes' (green) and 'Cancel' (grey).

Create and publish a release

1. Click **Release** to open the releases page.



2. Click **Draft a new release**.



3. In the fields, type `v2.0` as the version number and `Version 2.0` as the name. Then, click **Publish release**.

This repository: `JennCR / Think2018-7232`

Watch 0 Star 0 Fork 3

Code Pull requests 0 Projects 0 Wiki Insights Settings

Releases Tags

v2.0 @ Target: `master`

Excellent! This tag will be created from the target when you publish this release.

Version 2.0

Write Preview Markdown supported

Describe this release

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Attach binaries by dropping them here or selecting them.

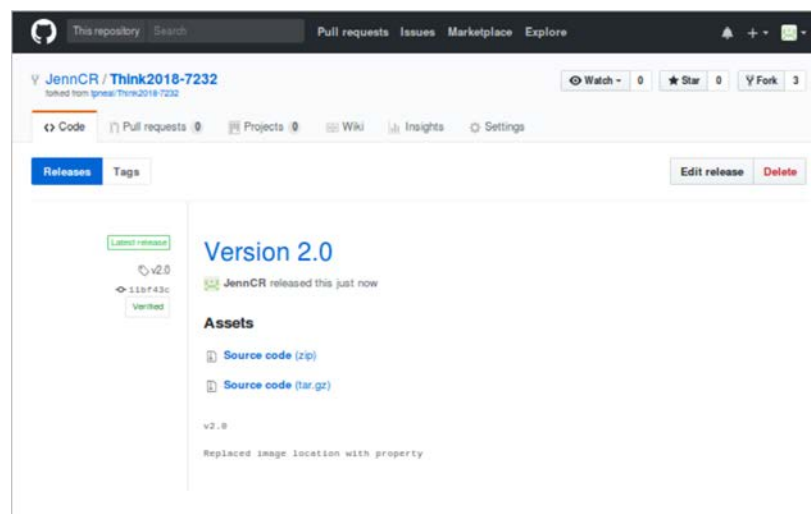
☐ This is a pre-release
We'll point out that this release is identified as non-production ready.

Publish release Save draft

Tagging suggestions
It's common practice to prefix your version names with the letter `v`. Some good tag names might be `v1.0` or `v2.3.4`.
If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be `vs.2-alpha` or `vs.9-beta.3`.

Semantic versioning
If you're new to releasing software, we highly recommend reading about [semantic versioning](#).

Now you have a new version that incorporates the updated parameter.



Import the parameterized version into UrbanCode Deploy

1. Click the **IBM UrbanCode Deploy** tab.
2. From the *stocktrader-all-in-one.yaml* component, click the **Versions** tab, and then click **Import New Versions**.
3. In the *Import New Versions* window, leave the fields blank, and click **Save**.

The new version is created with the updated parameter.

IBM UrbanCode Deploy

Home > Components > stocktrader-all-in-one.yaml (show details)

Component: stocktrader-all-in-one.yaml

Dashboard Usage Configuration Calendar **Versions** Processes Changes

Import New Versions

Version	Statuses	Type	Created By	Date	Description	Actions
<input type="text"/>	<input type="text"/>	Any				
v2.0		Full	UC Version Import	3/11/2018, 12:27 PM		Compare Delete Copy
V1.0		Full	UC Version Import	3/11/2018, 12:00 PM		Compare Delete Copy

2 records - Refresh Print

1 / 1

Rows 10

Currently Running Version Imports

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

4. Compare the changes between files by clicking **Compare** next to a version.

IBM UrbanCode Deploy

Home > Components > stocktrader-all-in-one.yaml (show details)

Component: stocktrader-all-in-one.yaml

Dashboard Usage Configuration Calendar **Versions** Processes Changes

Import New Versions

Version	Statuses	Type	Created By	Date	Description	Actions
v2.0		Full	UC Version Import	3/11/2018, 12:27 PM		Compare Delete Copy
V1.0		Full	UC Version Import	3/11/2018, 12:00 PM		Compare Delete Copy

3 records - Refresh Print

1 / 1

Rows 10

5. In the *Compare versions* window, select the version to compare, and then click **Submit**.

Compare Versions

Version *

6. In the *File Difference Report* section, click **Compare**.

Compare Versions

Version * V1.0

Submit

File Difference Report for stocktrader-all-in-one.yaml

Path	Version: v2.0 (Last Modified)	Version: V1.0 (Last Modified)	Actions
stocktrader-all-in-one.yaml	3/11/2018, 12:27 PM (Download)	3/11/2018, 12:00 PM (Download)	Compare

1 record - Refresh Print

1 / 1

Rows 10

Name	Left	Right
No records found.		

View the changes. You can see the substitutions you made in Version 2.0.

File Differences	
Version V1.0	Version v2.0
261 containers: 27 - name: loyalty-level 28 image: bluedemocluster.icp:8500/default/loyalty-level:latest 29 env: 30 - name: MQ ID	261 containers: 27 - name: loyalty-level 28 image: @registryHostAndRepo/loyalty-level:latest 29 env: 30 - name: MQ ID
112 containers: 113 - name: notification 114 image: bluedemocluster.icp:8500/default/notification:latest 115 env: 116 - name: MQ ID	112 containers: 113 - name: notification 114 image: @registryHostAndRepo/notification:latest 115 env: 116 - name: MQ ID
183 containers: 184 - name: portfolio 185 image: bluedemocluster.icp:8500/default/portfolio:latest 186 env: 187 - name: JDBC HOST	183 containers: 184 - name: portfolio 185 image: @registryHostAndRepo/portfolio:latest 186 env: 187 - name: JDBC HOST
261 containers: 262 - name: stock-quote 263 image: bluedemocluster.icp:8500/default/stock-quote:latest 264 env:	261 containers: 262 - name: stock-quote 263 image: @registryHostAndRepo/stock-quote:latest 264 env:

7. Close the *File Differences* window.

View the Replace Tokens step in the component process

Processes are lists of steps and connections between those steps. Each step is an individual command that runs on a target computer. Steps can manipulate files, run system commands, download files, and run programs.

1. Click the **Processes** tab, and then click **Process and Apply YAML file**.

IBM UrbanCode Deploy dW

Dashboard Components Applications Configuration Processes Resources Calendar Work Items Reports Settings

Home > Components > stocktrader-all-in-one.yaml

Component: stocktrader-all-in-one.yaml (show details)

Dashboard Usage Configuration Calendar Versions Processes Changes

Create Process

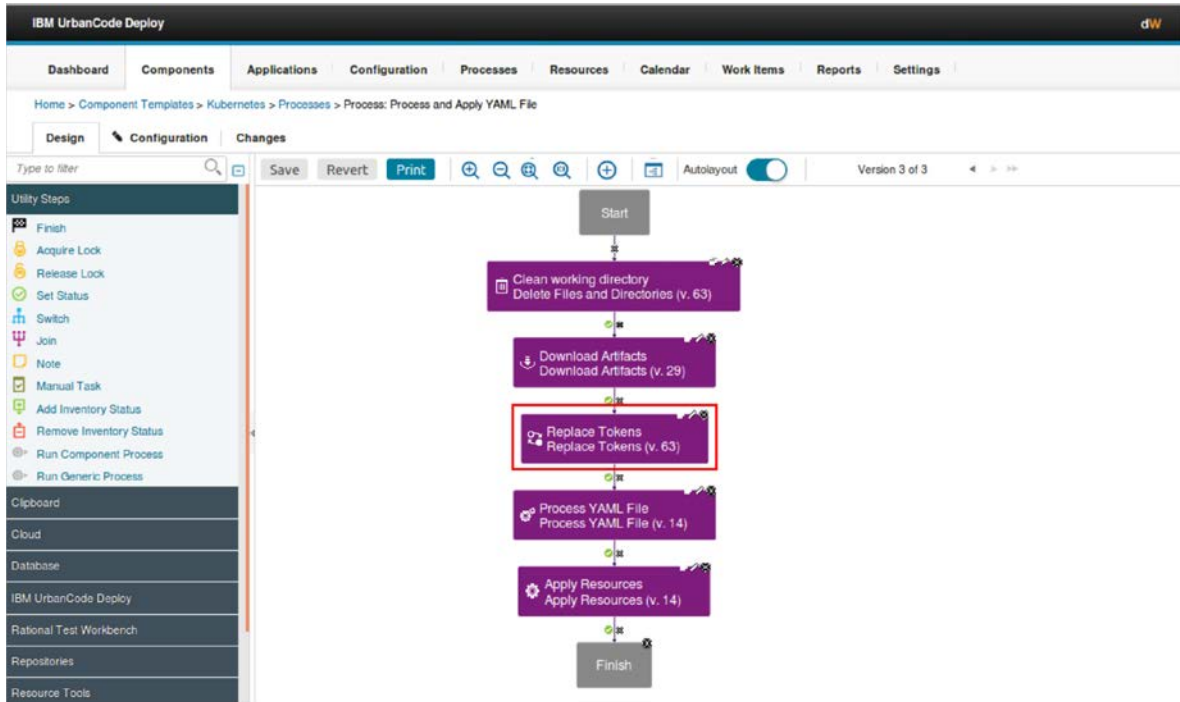
Process	Description	Actions
Process and Apply YAML File (Kubernetes)		Edit Copy Delete

1 record - Refresh Print

1 / 1

The process displays in the process editor. The process editor allows you to organize the steps in a process, specify their properties, and connect them to each other.

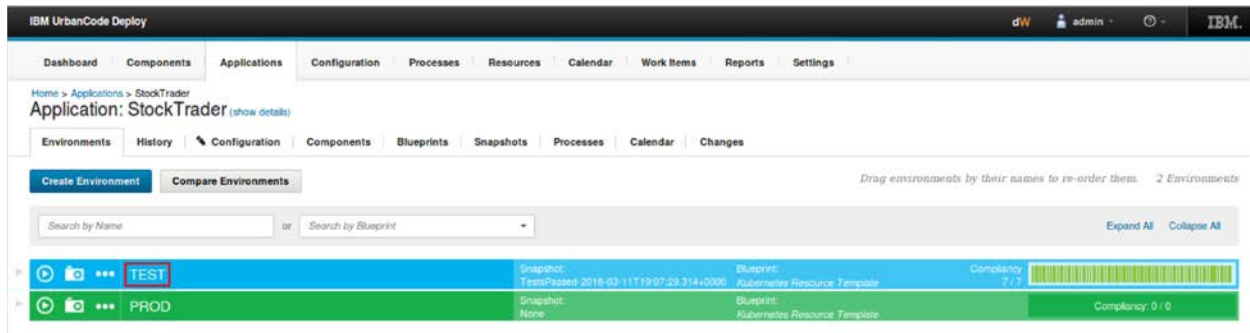
2. Click the **Edit** icon (looks like a small pencil) on the upper-right corner of the *Replace Tokens* step.



3. In the properties window for the step, view the *Start Token Delimiter* and *End Token Delimiter* fields. The ampersands (@) are identifying the beginning and end of the token.

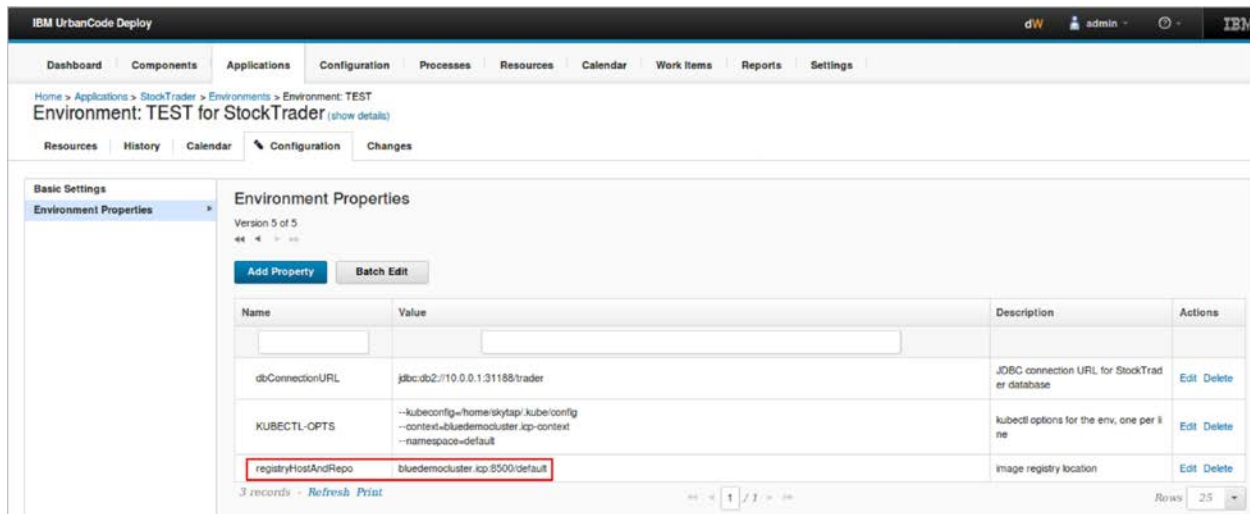
The screenshot shows the 'Edit Properties for Replace Tokens' dialog box. The 'Name' field is set to 'Replace Tokens'. The 'Include Files' field contains '***'. The 'Exclude Files' field is empty. The 'Property List' field contains '\$(p.environmentalProperties)'. The 'Property Prefix' field is empty. The 'Explicit Tokens' field is empty. The 'Start Token Delimiter' and 'End Token Delimiter' fields are both set to '@' and are highlighted with a red box. The 'Property File Name' field is set to 'replace_tokens.properties'. The 'Directory Offset' field is empty. The 'Custom Encoding' field is empty. The 'Working Directory' field is empty. The 'Post Processing Script' field is set to 'Step Default'. The 'Precondition' field is empty. The 'OK' and 'Cancel' buttons are at the bottom right.

- Click **Cancel** to exit the window.
- Return to the application by clicking the **Applications** tab, and then clicking **StockTrader**.
- From the **Environments** tab, click **TEST** to view the environment information.



- Click the **Configuration** tab, and then click **Environment Properties**.

Notice the `registryHostAndRepo` property that you substituted in the application description YAML file. For the TEST environment, it points to `bluedemocluster.icp:8500/default`.



- Return to the **Environments** tab, and click **PROD** to view the environment information.
- Click the **Configuration** tab, and then click **Environment Properties**. Notice the `registryHostAndRepo` property points to `bluedemocluster.icp:8500/production`. (The namespace is different.)

IBM UrbanCode Deploy

Home > Applications > StockTrader > Environments > Environment: PROD

Environment: PROD for StockTrader (show details)

Resources | History | Calendar | Configuration | Changes

Basic Settings | Environment Properties

Environment Properties

Version 4 of 4

Add Property Batch Edit

Name	Value	Description	Actions
dbConnectionURL	jdbc:db2://10.0.0.2:50000/trader	JDBC connection URL for StockTrader database	Edit Delete
KUBECTL_OPTS	--kubeconfig=/home/isyapi/kubeconfig --context=bluedemocluster.icp.context --namespace=production	kubectl options for the env, one per line	Edit Delete
registryHostAndRepo	bluedemocluster.icp:8500/production	image registry location	Edit Delete

3 records Refresh Print

Rows 25

Deploy the parameterized YAML file to the test environment

In this next deployment, you run the same processes to deploy the application, but you choose the new version of the application description YAML file.

Choose the component versions to deploy and run the application process

Deploy to the TEST environment again, but this time, the property within the token identifies the namespace.

1. Return to the **Environment** tab.
2. Click the play button to the left of the TEST environment.

IBM UrbanCode Deploy

Home > Applications > StockTrader

Application: StockTrader (show details)

Environments | History | Configuration | Components | Blueprints | Snapshots | Processes | Calendar | Changes

Create Environment Compare Environments

Search by Name or Search by Blueprint

Expand All Collapse All

TEST

Snapshot: TestsPassed: 2018-03-11T18:57:29-314-0000 Blueprint: Kubernetes Resource Template Compliance: 7/7

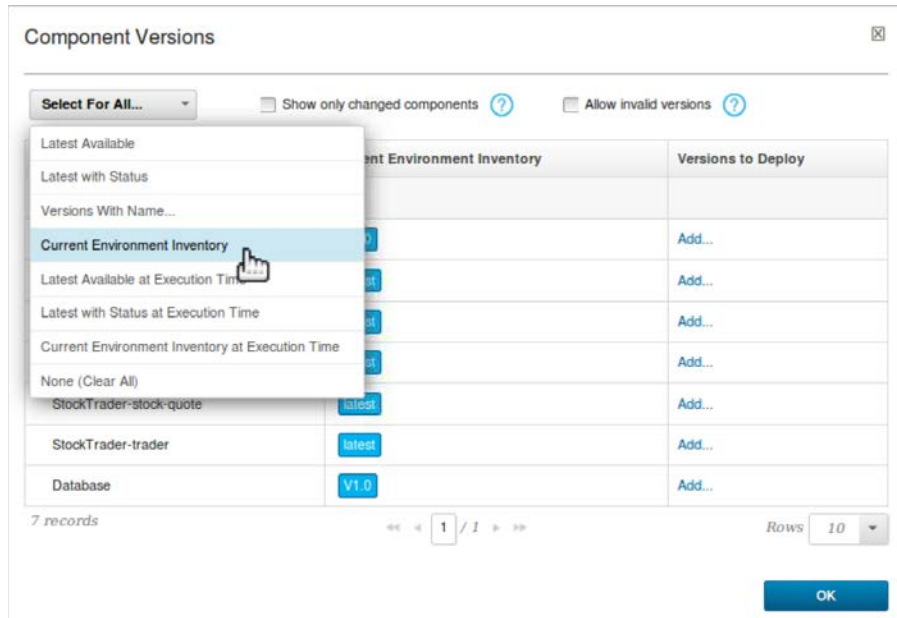
Component	Version	Date Deployed	Compliance	Actions
StockTrader-trader	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-stock-quote	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-portfolio	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-notification	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
StockTrader-loyalty-level	latest (View Details)	3/11/2018, 12:03 PM	Compliant (1/1)	View Request
Database	V1.0 (View Details)	3/11/2018, 12:02 PM	Compliant (1/1)	View Request
stocktrader-all-in-one.yaml	V1.0 (View Details)	3/11/2018, 12:02 PM	Compliant (1/1)	View Request

Refresh Print

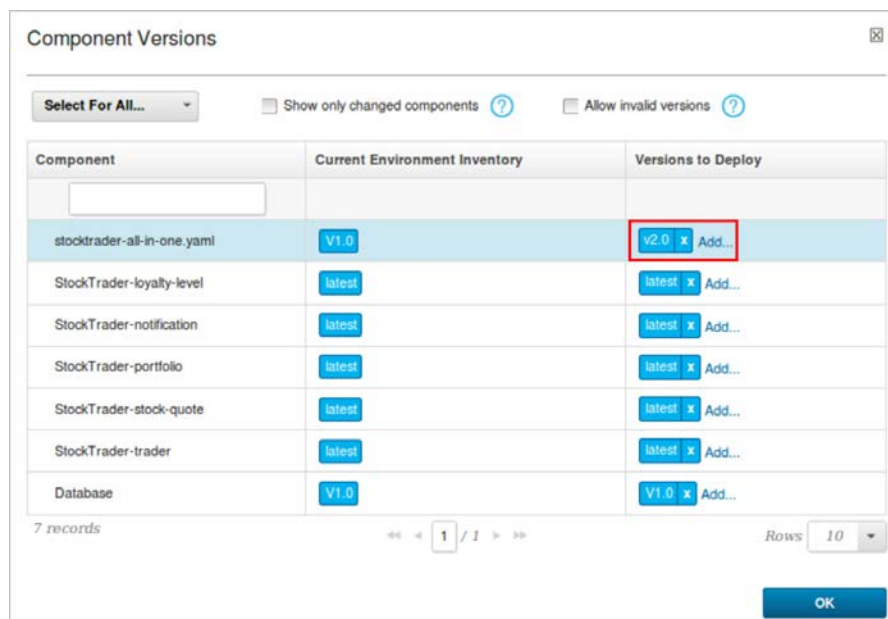
PROD

Snapshot: None Blueprint: Kubernetes Resource Template Compliance: 0/0

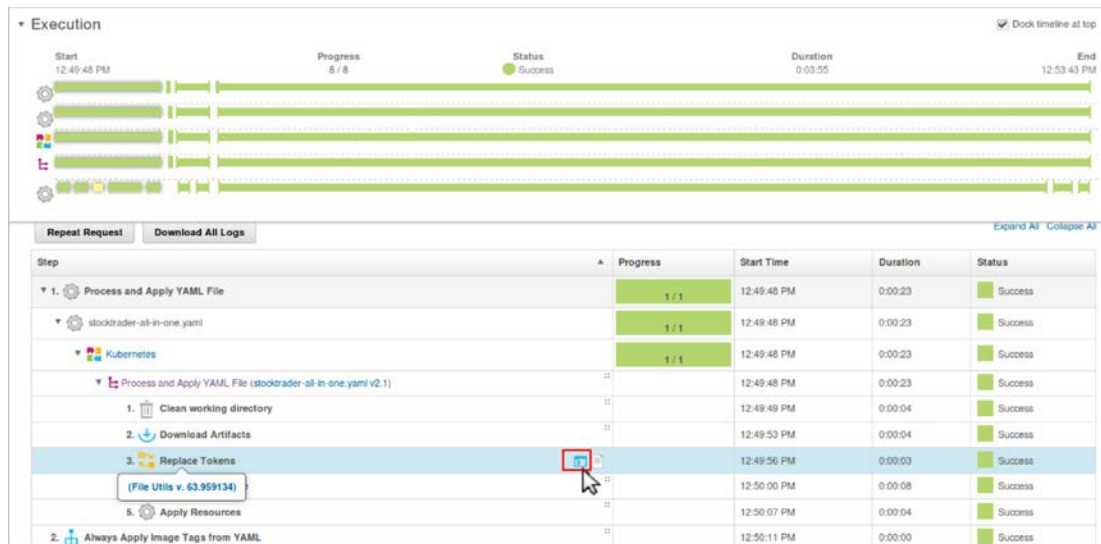
3. In the *Run Process on TEST* window, click **Choose Versions**.
4. In the *Component Versions* window, click **Select For All**, and then click **Current Environment Inventory**.



- Under the *Versions to Deploy* column in the *stocktrader-all-in-one.yaml* row, delete **v1.0**, and add **v2.0**.



- Click **OK**, and then click **Submit**. The same deployment runs.
- When the *Replace Tokens* step completes, click the **Output Log** icon.



The *Replace Tokens* step looks for properties in between the two ampersands (@). You can see it found several tokens, including the one you added, @registryHostAndRepo@, which replaced the property with the test location of bluedemocluster.icp:8500/default.

```

Output Log

Working Directory /opt/ibm-ucd/agent/var/work/stocktrader-all-in-one.yaml

8 customEncoding=
9 dir=
10 endDelimiter=@
11 envPropValues=alwaysApplyVersionsFromYaml=false,dbConnectionURL=jdbc:db2://10.0.0.1:31188/trader,r
12 --context=bluedemocluster.icp-context
13 --namespace=default
14 excludes=
15 explicitTokens=
16 includes=**/*
17 propFile=replace_tokens.properties
18 propertyPrefix=
19 startDelimiter=@
20 environment:
21 AGENT_HOME=/opt/ibm-ucd/agent
22 AH_AUTH_TOKEN=***
23 AH_WEB_URL=https://10.0.0.1:30386/
24 AUTH_TOKEN=***
25 DS_AUTH_TOKEN=***
26 DS_SYSTEM_ENCODING=UTF-8
27 JAVA_OPTS=-Dfile.encoding=UTF-8 -Dconsole.encoding=UTF-8
28 PLUGIN_HOME=/opt/ibm-ucd/agent/var/plugins/com.urbancode.air.plugin.FileUtils_63_0f62f8486e00beab0
29 UCD_USE_ENCRYPTED_PROPERTIES=true
30 UD_DIALOGUE_ID=e4365e2f-e658-416d-8d38-ac92231d984b
31 WE_ACTIVITY_ID=162169ca-d2c0-2b9c-541b-0d7f52c05dc5
32
33 added: @alwaysApplyVersionsFromYaml@:false
34 added: @registryHostAndRepo@:bluedemocluster.icp:8500/default
35 added: @dbConnectionURL@:jdbc:db2://10.0.0.1:31188/trader
36 added: @KUBECTL_OPTS@:--kubeconfig=/home/skytap/.kube/config
37 --context=bluedemocluster.icp-context
38 --namespace=default
39 Replacing tokens in files located in /opt/ibm-ucd/agent/var/work/stocktrader-all-in-one.yaml
40 [replace] Replaced 9 occurrences in 2 files.
41 Temporary property file deleted.
42
Download Log

```

- Close the output log.
- Return to the **Environment** tab by clicking the **StockTrader** breadcrumb link.
- Expand the test environment.

Notice the latest StockTrader images and the new version of the YAML file that was deployed.

IBM UrbanCode Deploy

Home > Applications > StockTrader

Application: StockTrader (show details)

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Create Environment Compare Environments

Search by Name or Search by Blueprint

Expand All Collapse All

TEST

Snapshot: TestsPassed-2018-03-11T19:53:40.070+0000

Blueprint: Kubernetes Resource Template

Compliance: 7/7

Component	Version	Date Deployed	Compliance	Actions
StockTrader-trader	latest (View Details)	3/11/2018, 12:50 PM	Compliant (1/1)	View Request
StockTrader-stock-quote	latest (View Details)	3/11/2018, 12:50 PM	Compliant (1/1)	View Request
StockTrader-portfolio	latest (View Details)	3/11/2018, 12:50 PM	Compliant (1/1)	View Request
StockTrader-loyalty-level	latest (View Details)	3/11/2018, 12:50 PM	Compliant (1/1)	View Request
StockTrader-notification	latest (View Details)	3/11/2018, 12:49 PM	Compliant (1/1)	View Request
Database	V1.0 (View Details)	3/11/2018, 12:49 PM	Compliant (1/1)	View Request
stocktrader-all-in-one.yaml	v2.0 (View Details)	3/11/2018, 12:49 PM	Compliant (1/1)	View Request

Refresh Print

11. Click the **Snapshots** tab. A new snapshot is listed. This is the snapshot that you will deploy to PROD.

12. Click the name of the snapshot, and then click **Component Versions**.

IBM UrbanCode Deploy

Home > Applications > StockTrader

Application: StockTrader (show details)

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Create Snapshot Import Snapshots

Snapshot	Description	Created	By	Actions
TestsPassed-2018-03-11T19:53:40.070+0000	All performance tests passed	3/11/2018, 12:53 PM	admin	Export Compare Edit Delete
TestsPassed-2018-03-11T19:07:29.314+0000	All performance tests passed	3/11/2018, 12:07 PM	admin	Export Compare Edit Delete

2 records - Refresh Print

Notice that version 2 of the *stocktrader-all-in-one.yaml* file is captured in the snapshot.

IBM UrbanCode Deploy

Home > Applications > StockTrader > Snapshots > Snapshot: TestsPassed-2018-03-11T19:53:40.070+0000

Snapshot: TestsPassed-2018-03-11T19:53:40.070+0000 (show details)

Dashboard Component Versions Configuration Calendar

Select For All... Revert Lock Versions

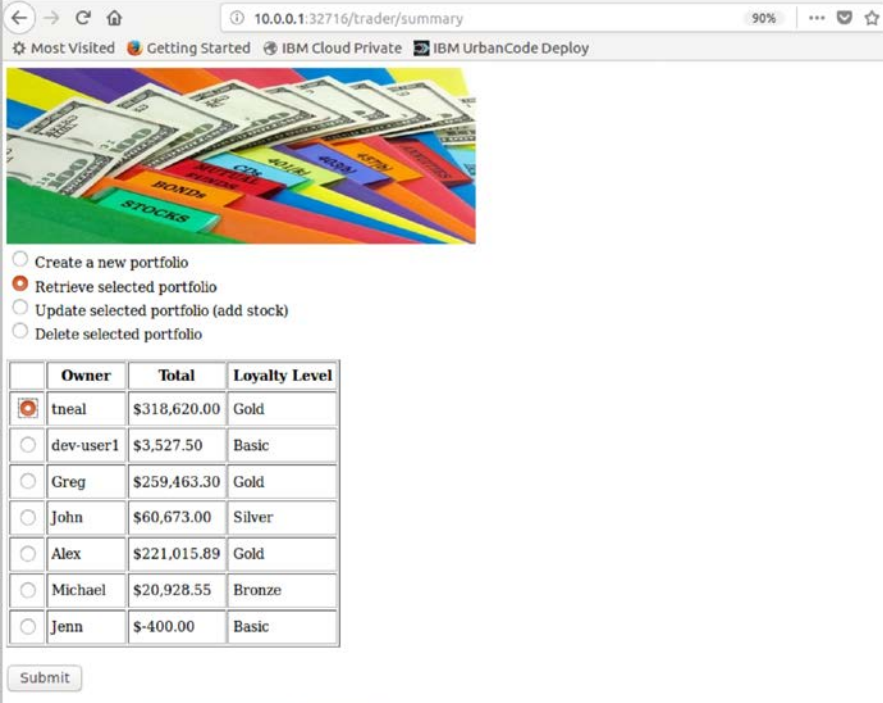
Component	Versions
AutomatedTests	Add...
Database	V1.0 Add...
StockTrader-loyalty-level	latest Add...
StockTrader-notification	latest Add...
StockTrader-portfolio	latest Add...
StockTrader-stock-quote	latest Add...
StockTrader-trader	latest Add...
stocktrader-all-in-one.yaml	v2.0 Add...

8 records

Verify the second deployment to the test environment

The second deployment to the test environment uses the same port number.

1. Return to the **StockTrader** browser tab to verify the deployment.
2. Refresh the browser.



	Owner	Total	Loyalty Level
<input checked="" type="radio"/>	tneal	\$318,620.00	Gold
<input type="radio"/>	dev-user1	\$3,527.50	Basic
<input type="radio"/>	Greg	\$259,463.30	Gold
<input type="radio"/>	John	\$60,673.00	Silver
<input type="radio"/>	Alex	\$221,015.89	Gold
<input type="radio"/>	Michael	\$20,928.55	Bronze
<input type="radio"/>	Jenn	\$-400.00	Basic

Submit

Test the application, as you did before, by creating a portfolio, adding stock to an account, and retrieving portfolio information.

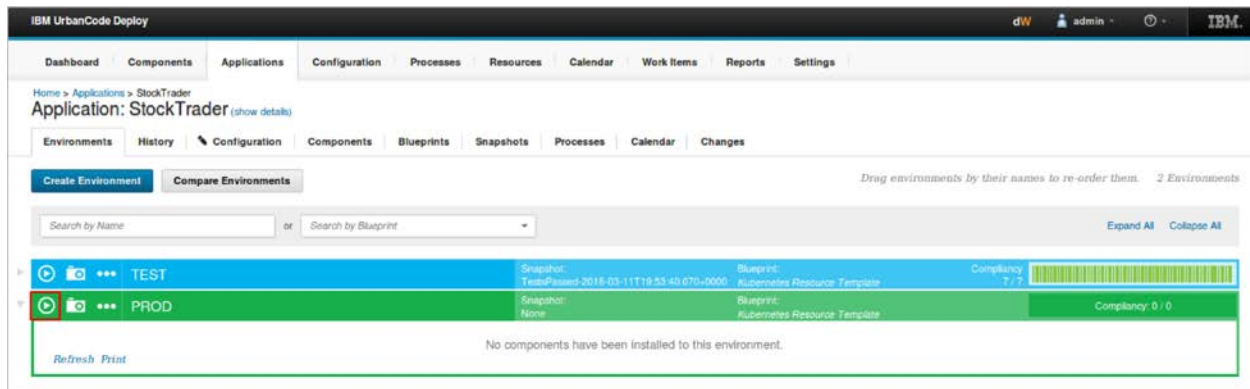
Exercise 3: Deploy the snapshot to the production environment

In this exercise, you complete the following tasks:

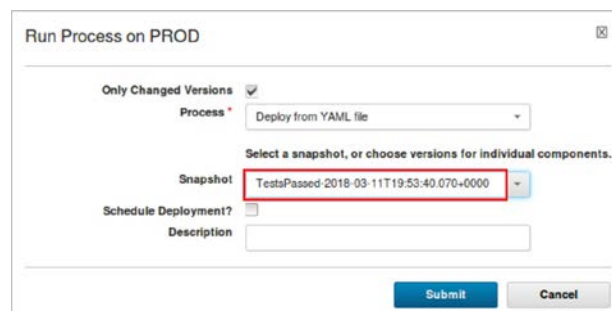
- Deploy the snapshot, include the parameterized version of the YAML file, to the production environment
- Verify the application in the web UI
- View the production version of the microservice in IBM Cloud Private

Deploy the snapshot to the production environment

1. Click the **UrbanCode Deploy** tab.
2. Click the **StockTrader** breadcrumb link.
3. Expand the **PROD** environment. Notice that nothing has been deployed.
4. Click the play button to the left of the PROD environment.



5. In the *Run Process on PROD* window, select the latest snapshot that you created, and then click **Submit**.



You are deploying a new instance of StockTrader into another Kubernetes namespace. You can separate environments by namespaces (like this) but you can also separate environments by cluster (one for test and one for production).

6. In the *Replace Tokens* step, click the **Output Log** icon.

Start	Progress	Status	Duration	End
12:58:57 PM	2 / 2	Success	0:00:48	12:59:45 PM
Kubernetes				
1 / 1				
1 / 1				
Process and Apply YAML File (stocktrader-all-in-one.yaml v2.1)				
1. Clean working directory		Success	0:00:25	12:58:57 PM
2. Download Artifacts		Success	0:00:03	12:58:57 PM
3. Replace Tokens		Success	0:00:05	12:59:01 PM
4. Process YAML File		Success	0:00:04	12:59:05 PM
5. Apply Resources		Success	0:00:08	12:59:09 PM
Always Apply Image Tags from YAML		Success	0:00:04	12:59:17 PM
		Success	0:00:00	12:59:21 PM

Notice the `@registryHostAndRepo@` token is replaced with a new namespace: `production`.

```

Working Directory /opt/ibm-ucd/agent/var/work/stocktrader-all-in-one.yaml
8 customEncoding=
9 dir=
10 endDelimiter=@
11 envPropValues=alwaysApplyVersionsFromYaml=false,dbConnectionURL=jdbc:db2://10.0.0.2:50000/trader,registryHostAndRepo=bluedemocluster.icp:81
12 --context=bluedemocluster.icp-context
13 --namespace=production
14 excludes=
15 explicitTokens=
16 includes=**/*
17 propFiles=replace_tokens.properties
18 propertyPrefix=
19 startDelimiter=@
20 environment:
21   AGENT_HOME=/opt/ibm-ucd/agent
22   AH_AUTH_TOKEN=****
23   AH_WEB_URL=https://10.0.0.1:30386/
24   AUTH_TOKEN=****
25   DS_AUTH_TOKEN=****
26   DS_SYSTEM_ENCODING=UTF-8
27   JAVA_OPTS=-Dfile.encoding=UTF-8 -Dconsole.encoding=UTF-8
28   PLUGIN_HOME=/opt/ibm-ucd/agent/var/plugins/com.urbancode.air.plugin.FileUtils_63_0f62f8486e00beab0cf3ad972e281440c83cbc56007cf955d3b73d1b9
29   UCD_USE_ENCRYPTED_PROPERTIES=true
30   UD_DIALOGUE_ID=ea8462dc-4808-4681-accf-1fdd22e6487f
31   WE_ACTIVITY_ID=16216a50-ac14-f8a6-72bc-96d50375d86f
32 =====
33 added: alwaysApplyVersionsFromYaml=false
34 added: @registryHostAndRepo@:bluedemocluster.icp:8500/production
35 added: @dbConnectionURL@:jdbc:db2://10.0.0.2:50000/trader
36 added: @KUBECTL_OPTS@:--kubeconfig=/home/skytap/.kube/config
37 --context=bluedemocluster.icp-context
38 --namespace=production
39 Replacing tokens in files located in /opt/ibm-ucd/agent/var/work/stocktrader-all-in-one.yaml
40 [replace] Replaced 9 occurrences in 2 files.
41 Temporary property file deleted.
42
  
```

7. Close the output log.
8. In the process request, notice that Step 6 did not run. Because you deployed to the production environment, the Rational Performance Test was skipped. This step only runs in a test environment.

Repeat Request		Download All Logs		Expand All Collapse All	
Step	Progress	Start Time	Duration	Status	
1. Process and Apply YAML File	1 / 1	12:58:57 PM	0:00:25	Success	
2. Always Apply Image Tags from YAML		12:59:21 PM	0:00:00	Success	
3. Update Inventory	0 / 0	12:59:21 PM	0:00:00	Success	
4. PerformDatabaseOps	1 / 1	12:59:22 PM	0:00:23	Success	
5. RunTests		12:59:45 PM	0:00:00	Success	
6. AutomatedTesting				Not Started	
Total Execution	2 / 2	12:58:57 PM	0:00:49	Success	

- When the deployment is finished, click the **StockTrader** breadcrumb link to return to the Environments tab.
- Expand the PROD environment to see the installed components:

IBM UrbanCode Deploy

Dashboard

Components

Applications

Configuration

Processes

Resources

Calendar

Work Items

Reports

Settings

Home > Applications > StockTrader

Application: StockTrader [\(show details\)](#)

Environments

History

Configuration

Components

Blueprints

Snapshots

Processes

Calendar

Changes

Create Environment

Compare Environments

Drag environments by their names to re-order them. 2 Environments

Search by Name

or

Search by Blueprint

Expand All

Collapse All

TEST

Snapshot: TestsPassed-2018-03-11T19:53:40.070+0000

Blueprint: Kubernetes Resource Template

Compliance: 7 / 7

PROD

Snapshot: TestsPassed-2018-03-11T19:53:40.070+0000

Blueprint: Kubernetes Resource Template

Compliance: 7 / 7

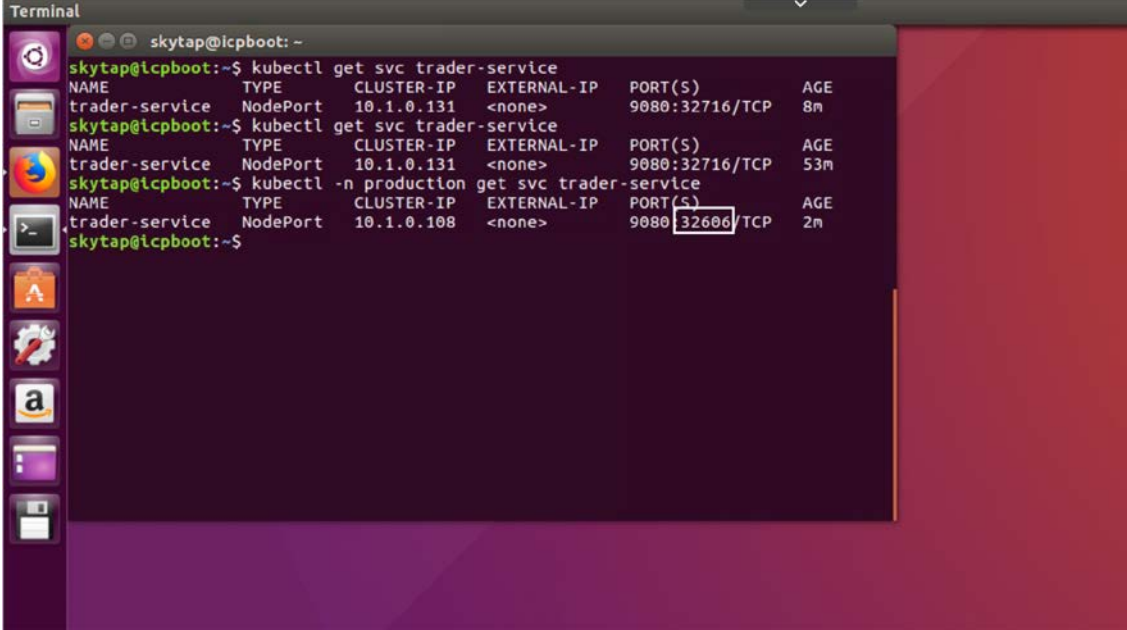
Component	Version	Date Deployed	Compliance	Actions
StockTrader-trader	latest (View Details)	3/11/2018, 12:59 PM	Compliant (1/1)	View Request
StockTrader-stock-quote	latest (View Details)	3/11/2018, 12:59 PM	Compliant (1/1)	View Request
StockTrader-portfolio	latest (View Details)	3/11/2018, 12:59 PM	Compliant (1/1)	View Request
StockTrader-notification	latest (View Details)	3/11/2018, 12:59 PM	Compliant (1/1)	View Request
StockTrader-loyalty-level	latest (View Details)	3/11/2018, 12:59 PM	Compliant (1/1)	View Request
Database	V1.0 (View Details)	3/11/2018, 12:58 PM	Compliant (1/1)	View Request
stocktrader-all-in-one.yaml	v2.0 (View Details)	3/11/2018, 12:58 PM	Compliant (1/1)	View Request

Refresh Print

Verify the application deployment to the production environment

Now that the StockTrader application is deployed to the production environment, you can view the web UI that's in IBM Cloud Private. First, you need to retrieve the port number.

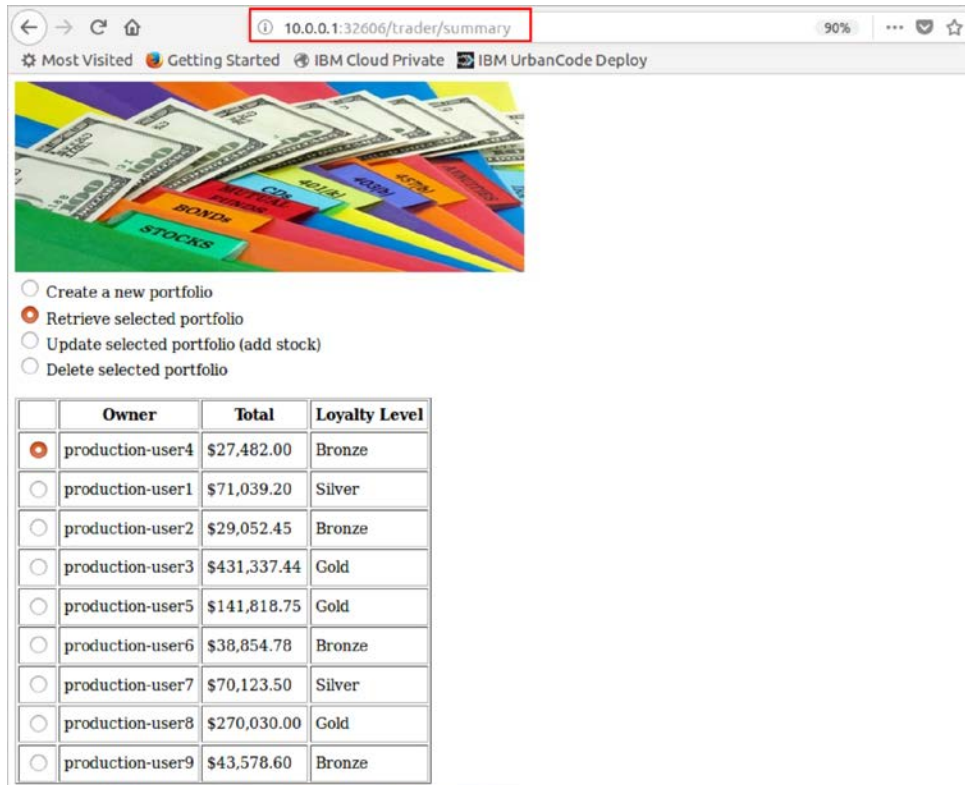
- Open the command-line prompt, and type `kubectl -n production get svc trader-service`, and press Enter. This command finds the port (in the production namespace) that the StockTrader application is listening at.
- Copy the port number.

A terminal window titled 'Terminal' with a dark purple background. The prompt is 'skytap@icpboot: ~'. The user enters 'kubectl get svc trader-service' and the output is a table with columns: NAME, TYPE, CLUSTER-IP, EXTERNAL-IP, PORT(S), and AGE. The first entry is 'trader-service' with type 'NodePort', cluster IP '10.1.0.131', and port '9080:32716/TCP', aged '8m'. The user then enters 'kubectl get svc trader-service' again, and the output is identical. Finally, the user enters 'kubectl -n production get svc trader-service' and the output is a table with columns: NAME, TYPE, CLUSTER-IP, EXTERNAL-IP, PORT(S), and AGE. The entry is 'trader-service' with type 'NodePort', cluster IP '10.1.0.108', and port '9080:32606/TCP', aged '2m'. The port '32606' is highlighted with a white box.

```
skytap@icpboot:~$ kubectl get svc trader-service
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
trader-service NodePort       10.1.0.131    <none>         9080:32716/TCP   8m
skytap@icpboot:~$ kubectl get svc trader-service
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
trader-service NodePort       10.1.0.131    <none>         9080:32716/TCP   53m
skytap@icpboot:~$ kubectl -n production get svc trader-service
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
trader-service NodePort       10.1.0.108    <none>         9080:32606/TCP   2m
skytap@icpboot:~$
```

3. Open a new browser tab, and type the URL `http://10.0.0.1:<port number>/trader/summary`, pasting your port number in the appropriate spot. For instance, <http://10.0.0.1:32606/trader/summary>.
4. Press Enter.

The StockTrader application displays in the browser. Notice that the production database contains different data than the test database. When deploying to the production environment, the StockTrader application uses a Kubernetes secret to control what database it's working with. In production, the secret points to an on-premise database that contains the customer data.

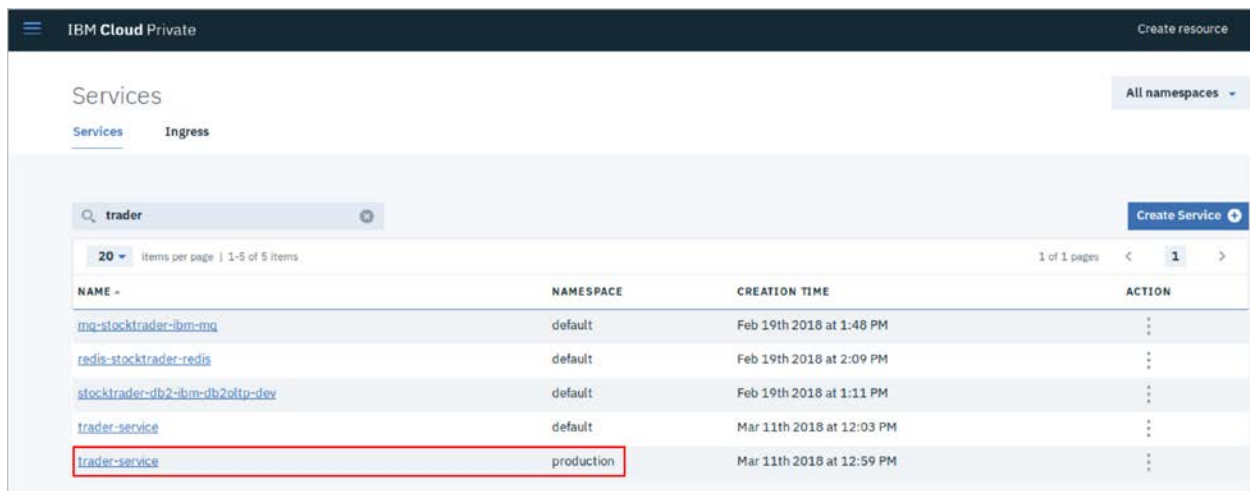


Test the application, as you did earlier, by creating a portfolio, adding stock to an account, and retrieving portfolio information.

View the microservice in IBM Cloud Private (production)

1. Click the **IBM Cloud Private** tab, and refresh the *Services* page (**Network Access > Services**).
2. If needed type `trader` in the **Filter** field.

The StockTrader service (trader-service) in the production namespace is listed.



Summary

Congratulations! You've just completed deploying an application that consists of multiple microservices into an IBM Cloud Private cluster using UrbanCode Deploy. You've seen how UrbanCode Deploy can accomplish these tasks:

- Integrate with public services like GitHub
- Deploy a containerized application into multiple namespaces and environments in IBM Cloud Private
- Perform any required operations on the database after deployment of the application, both with on-premise and cloud databases.
- Run a Rational Performance Tester schedule to verify the application was deployed successfully and passes all required tests

What's next

Are you interested in monitoring your cloud native applications? Come and check out Wednesday's lab and learn how to use IBM Application Performance Management (APM).

CLOUD NATIVE APPLICATION PERFORMANCE MANAGEMENT

Wednesday, 8:30 AM – 10:10 AM | Session ID: 1335A

Mandalay Bay South, Level 2, Oceanside, Think Academy | Lab 4

With the adoption of cloud-native architectures, auto-scaling applications, Docker, Kubernetes, microservices and more, managing these business applications has become increasingly challenging. In addition to the cloud-native architectures, many customers are running these applications on-premise or in a hybrid-cloud environment.

This session focuses on how IBM's own cloud-native solution can be run in a private cloud or as a SaaS offering to help customers monitor their applications. The IBM Application Performance Management (APM) solution leverages many of the same technologies that customers are using in their cloud-native business applications.

We Value Your Feedback!

- Don't forget to submit your Think 2018 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2018 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.