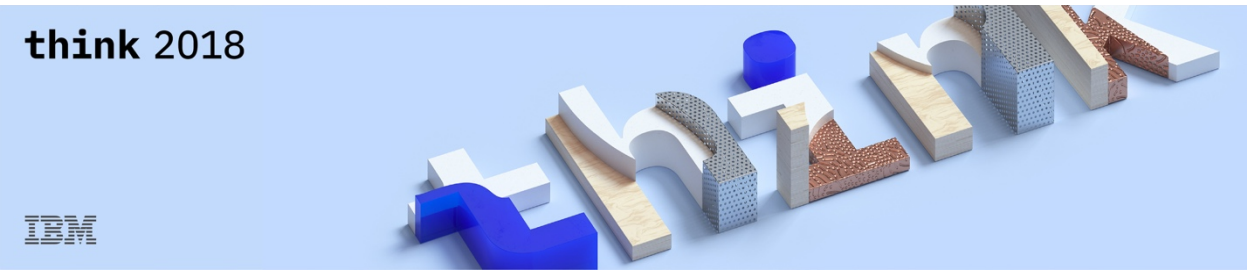


Lab Center – Hands-on Lab

Session 7232

Hands-on Hybrid Management with IBM Cloud Private

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

Table of Contents

Disclaimer

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.

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

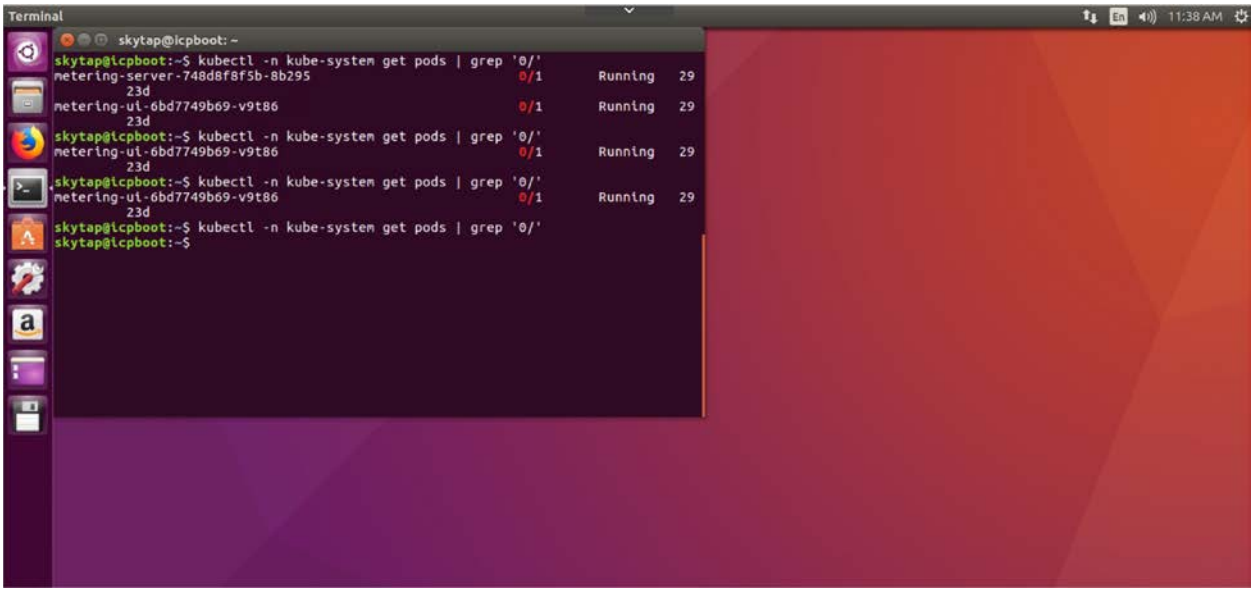
The lab environment consists of two VMs, named *Boot/Master Node* and *Ubuntu16x64*. The *Boot/Master Node* VM contains an installation of IBM Cloud Private 2.1.0.1, kubectl, and helm command-line clients, as well as an UrbanCode Deploy agent. The IBM Cloud Private cluster has two namespaces defined, *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.



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

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

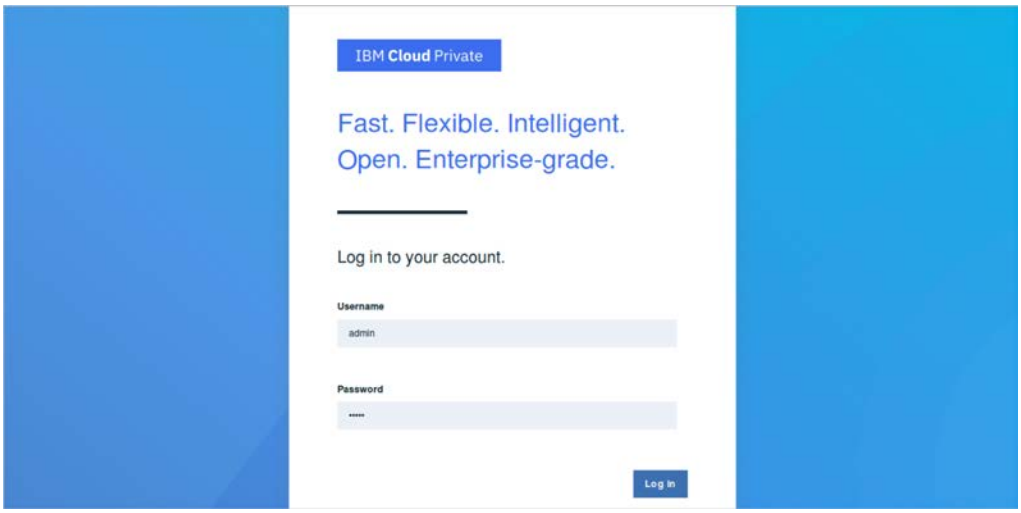
In this exercise, you complete the following tasks:

- 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

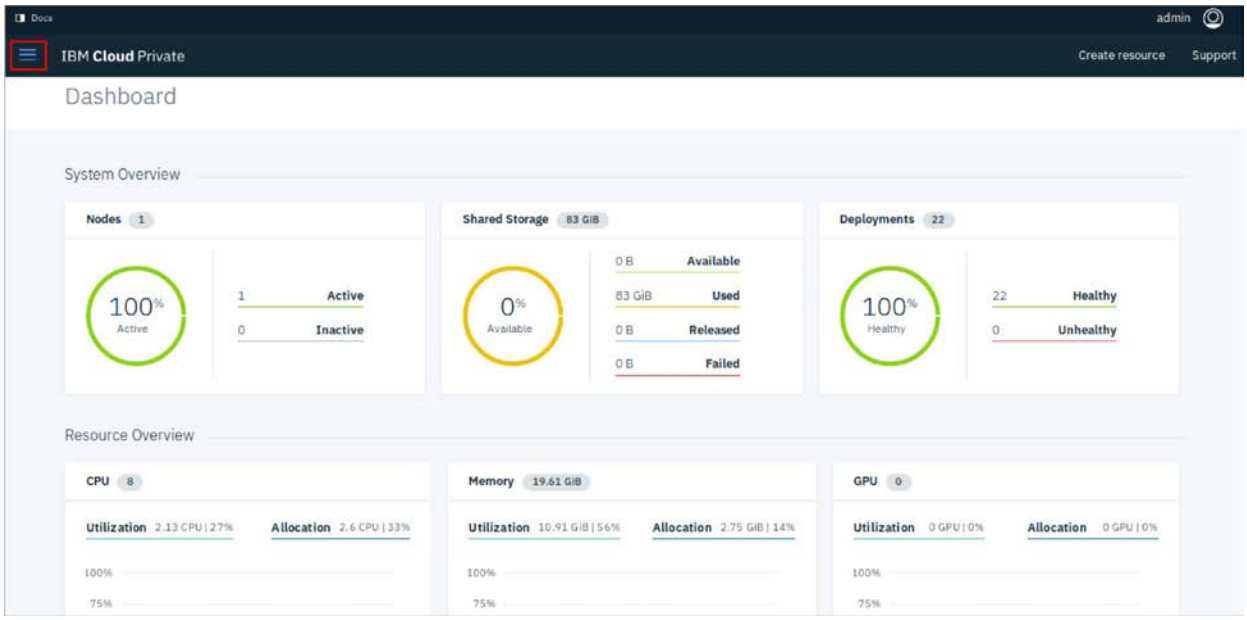
IBM Cloud Private is an application platform for developing and managing on-premises, containerized applications. It is an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image repository, a management console, and monitoring frameworks.

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**.

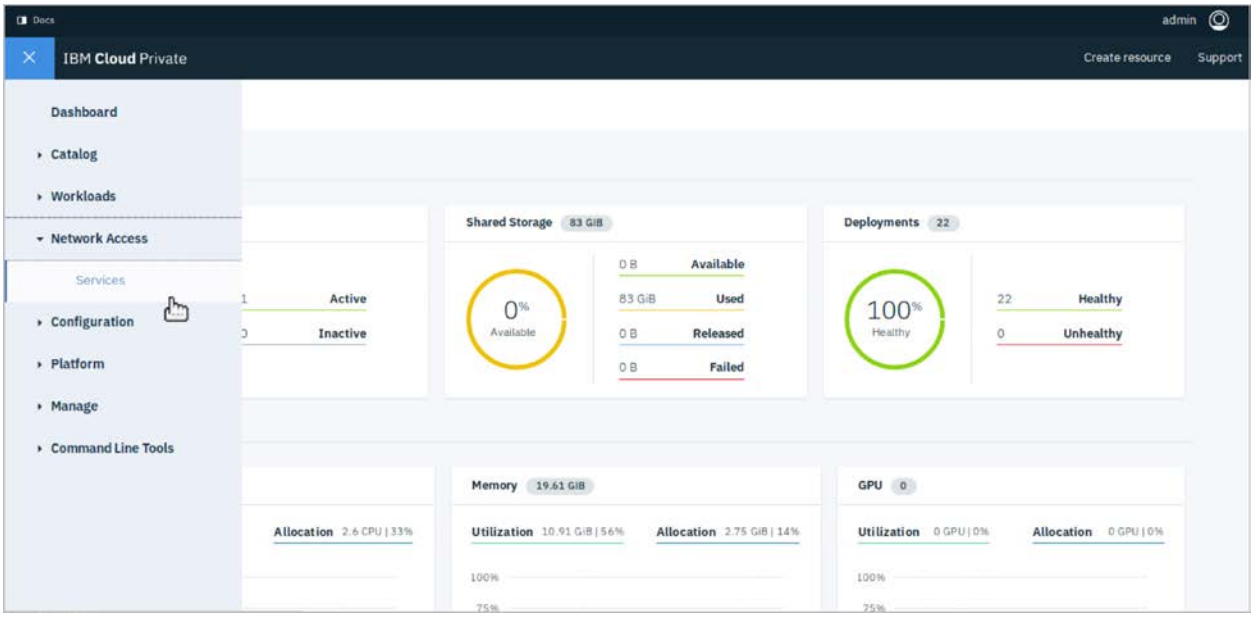


The IBM Cloud Private dashboard displays.

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



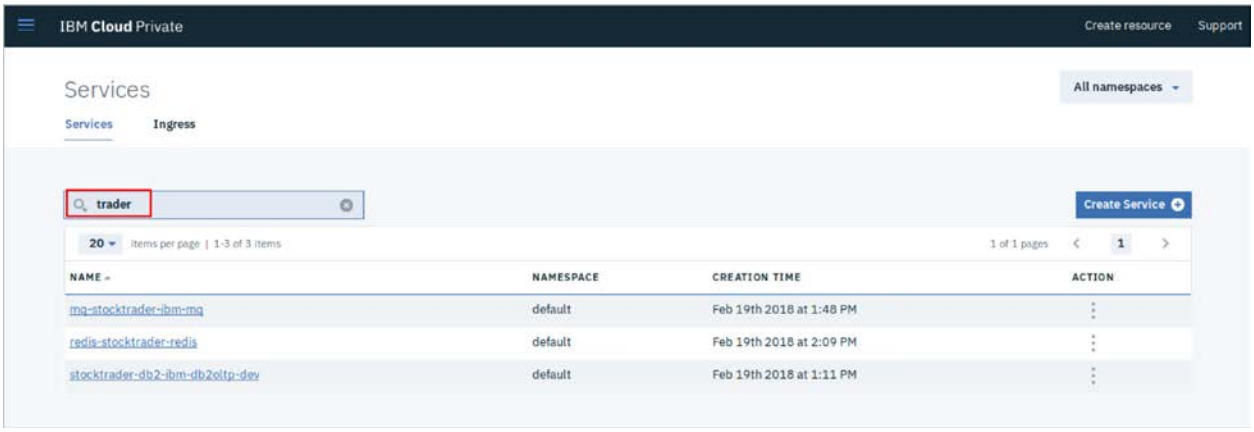
4. From the menu, click **Network Access**, and then click **Services**.



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.



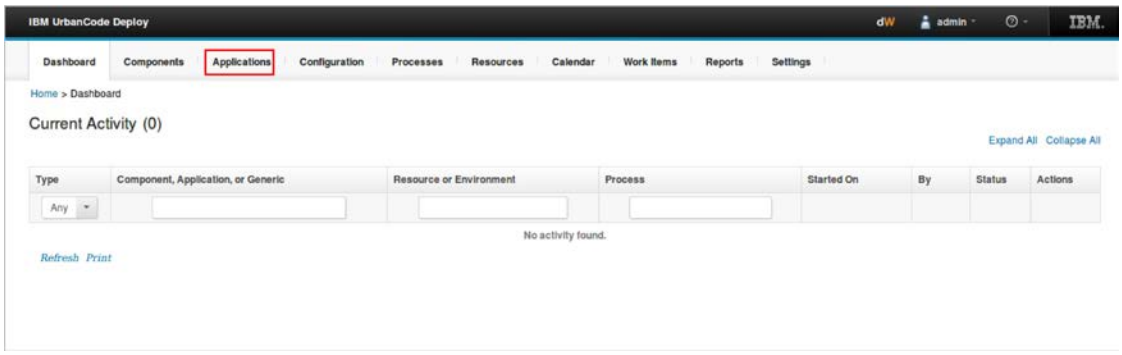
View the StockTrader application environments

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.

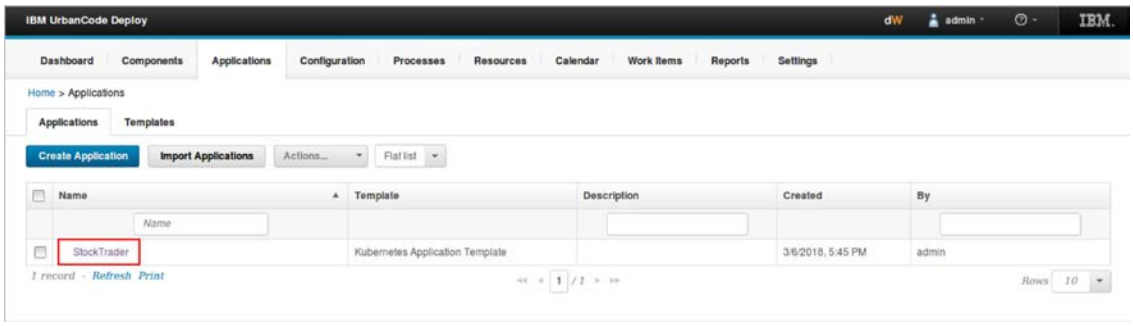
1. Open a new browser tab, and click the **IBM UrbanCode Deploy** bookmark.
2. If prompted for the username and password, type `admin` in both fields, and then click **Log in**.

The UrbanCode Deploy dashboard displays.

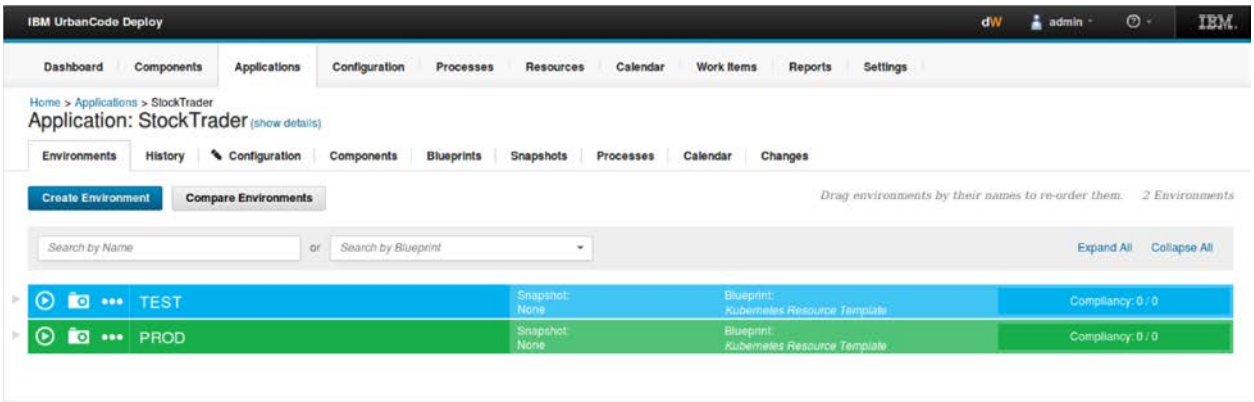
3. From the menu along the top of the page, click the **Applications** tab.



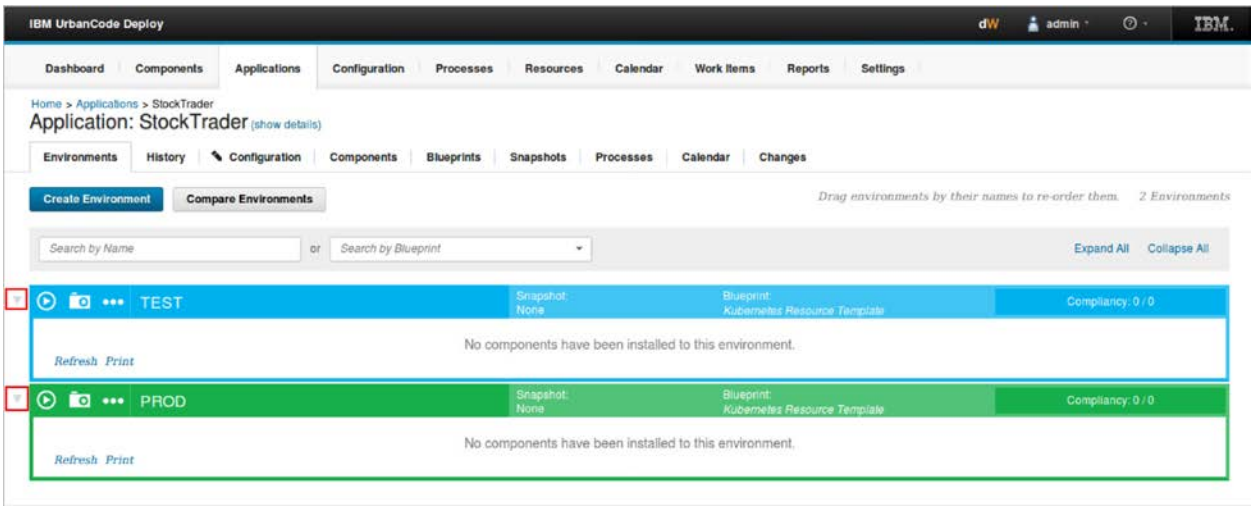
4. Click the **StockTrader** application name.



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.

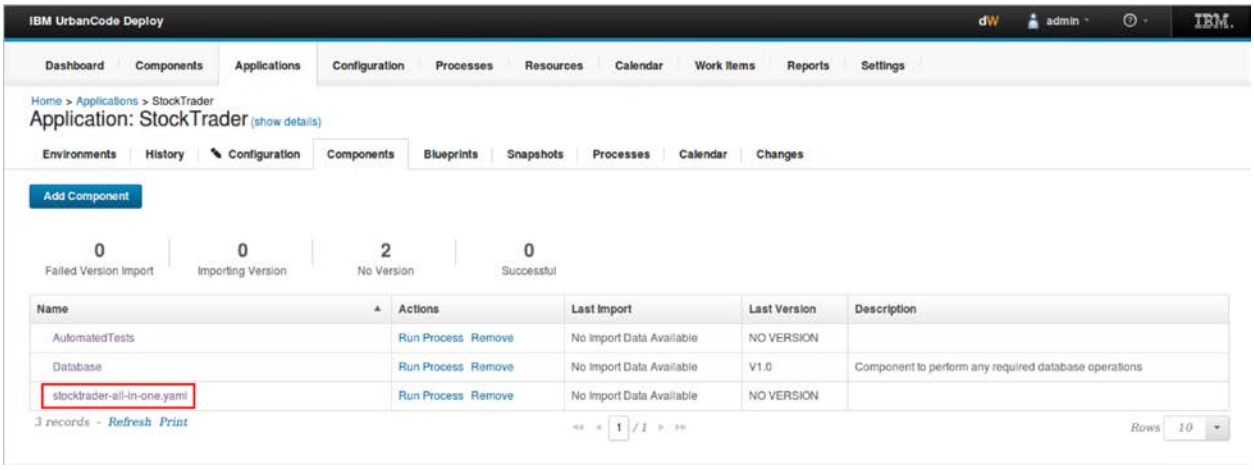


Currently, nothing is deployed to either environment.

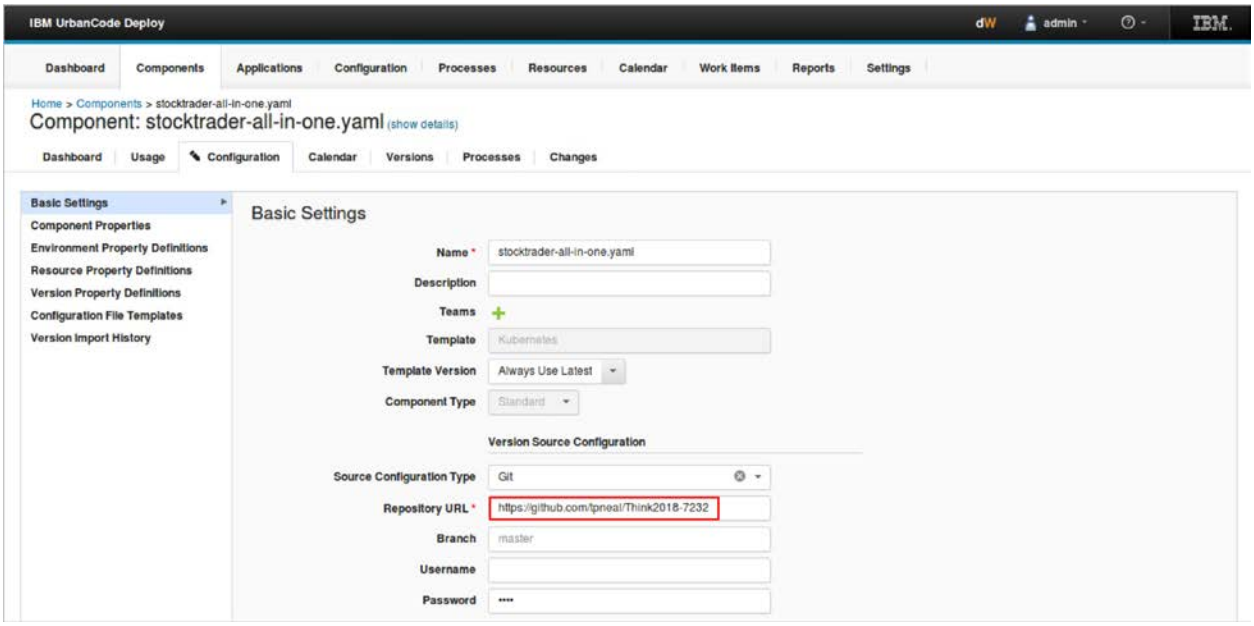
6. Click the **Components** tab.

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**.



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



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.

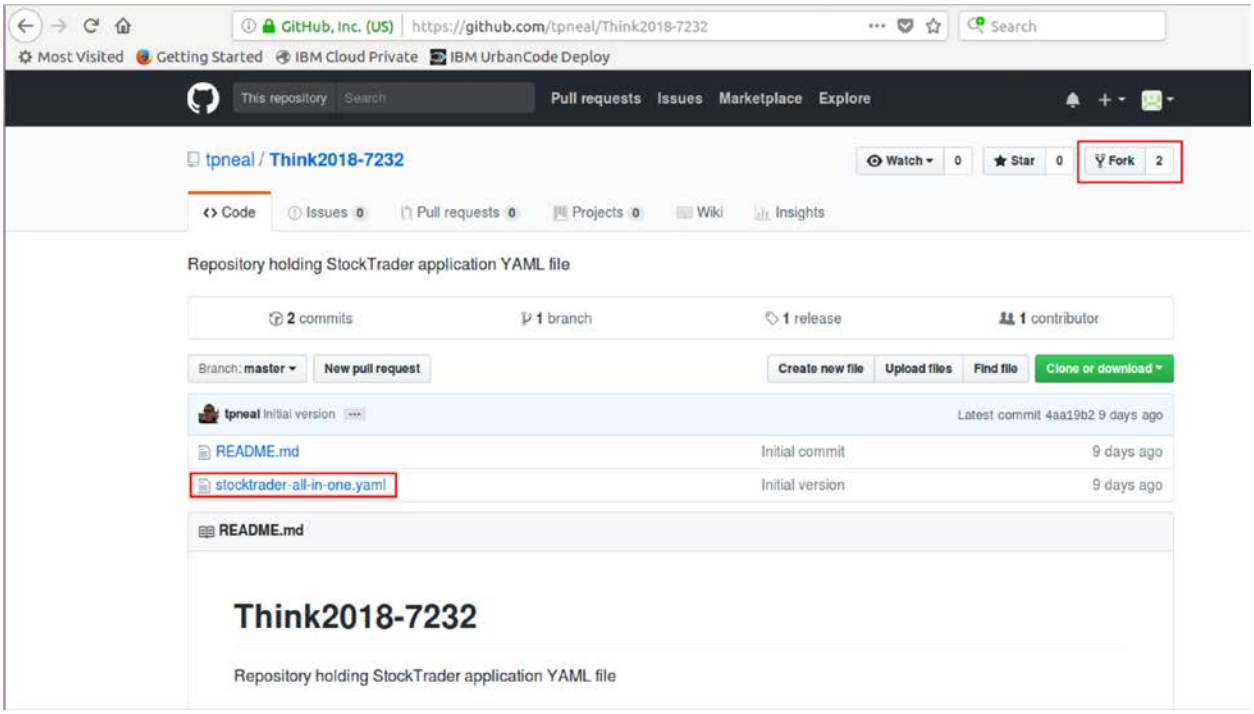
Import the StockTrader version

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

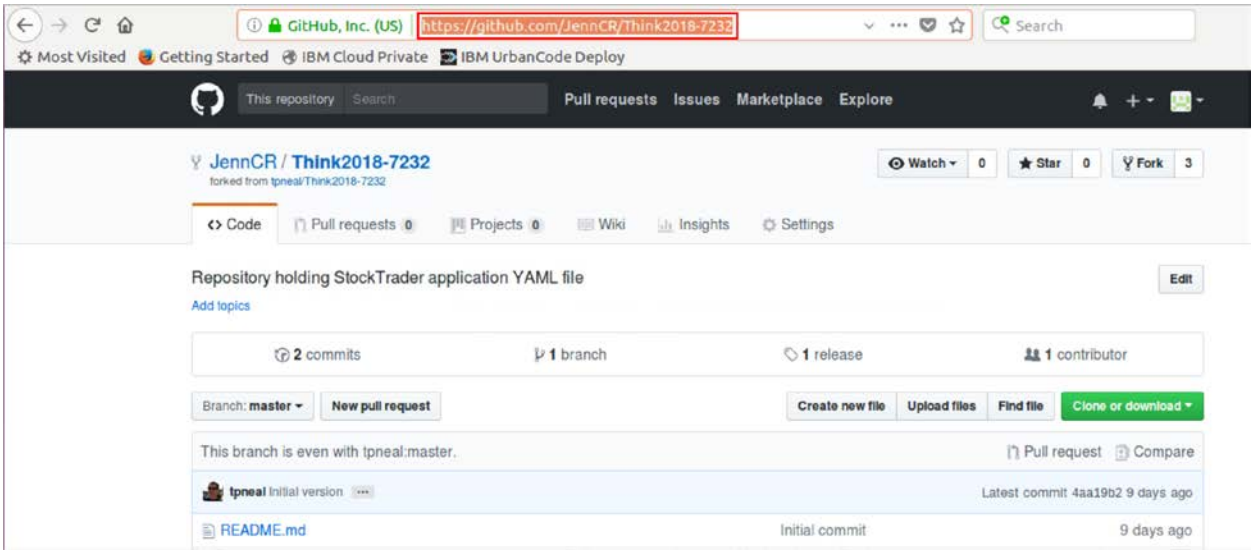
Fork the GitHub StockTrader project

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**.

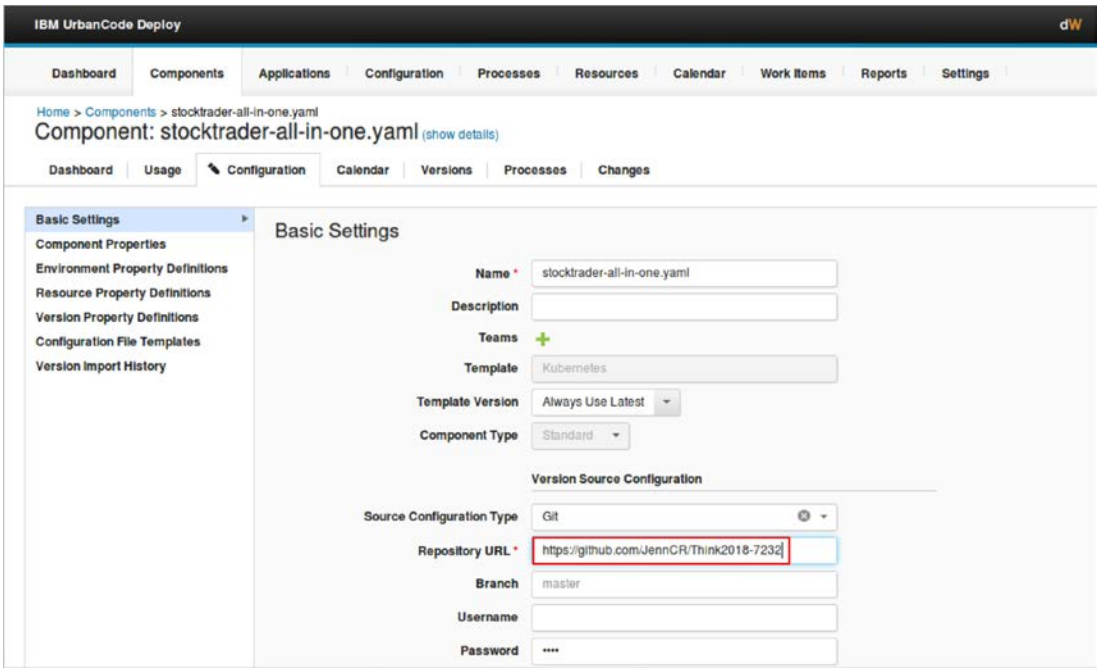


3. Copy your URL for the Think2018-7232 project.

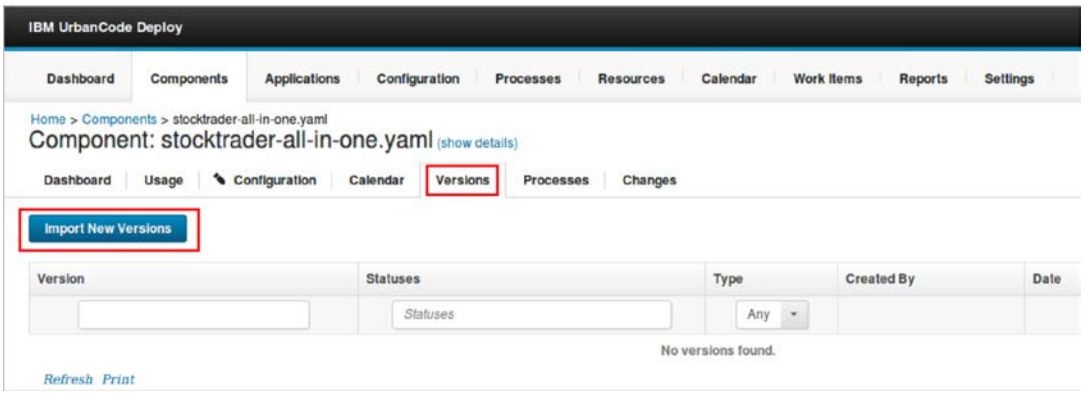


Update the configuration and import the new version

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



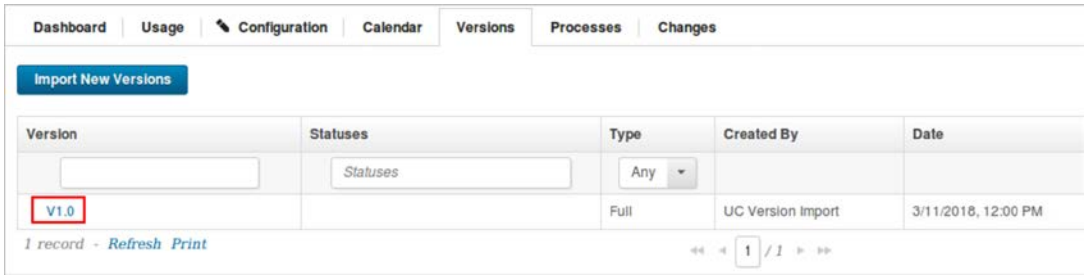
2. Click the **Versions** tab, and then click **Import New Versions**.



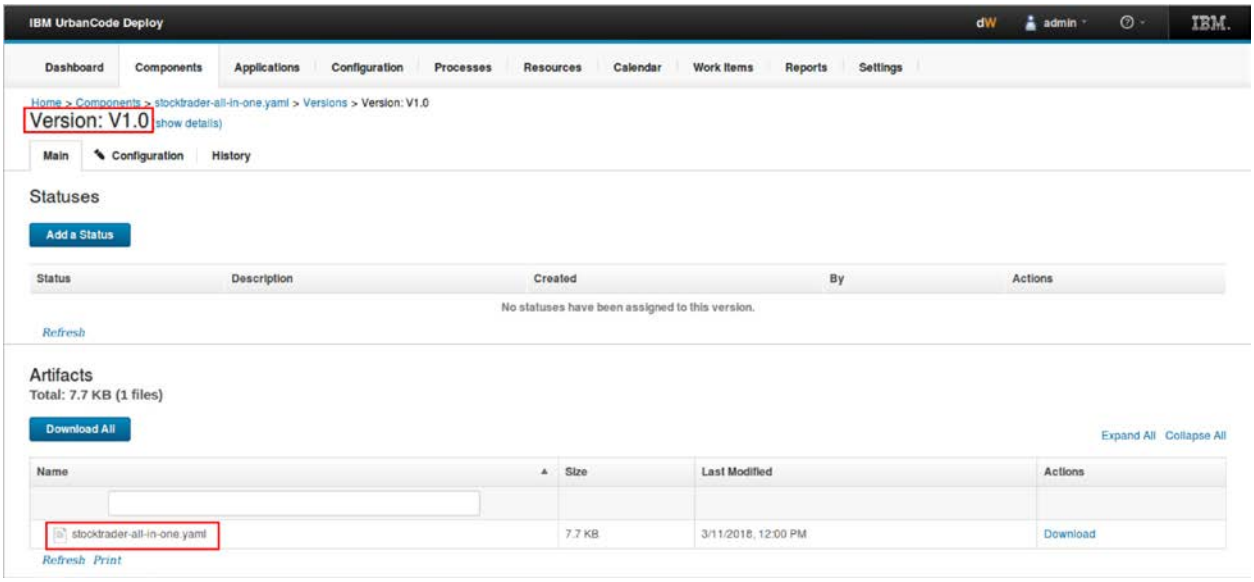
3. In the *Import New Versions* window, leave the fields blank, and click **Save**.

After a few moments, version 1.0 of the YAML file is imported. (You might need to click **Refresh**.)

4. Click **V1.0** to see what files are in this version.



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



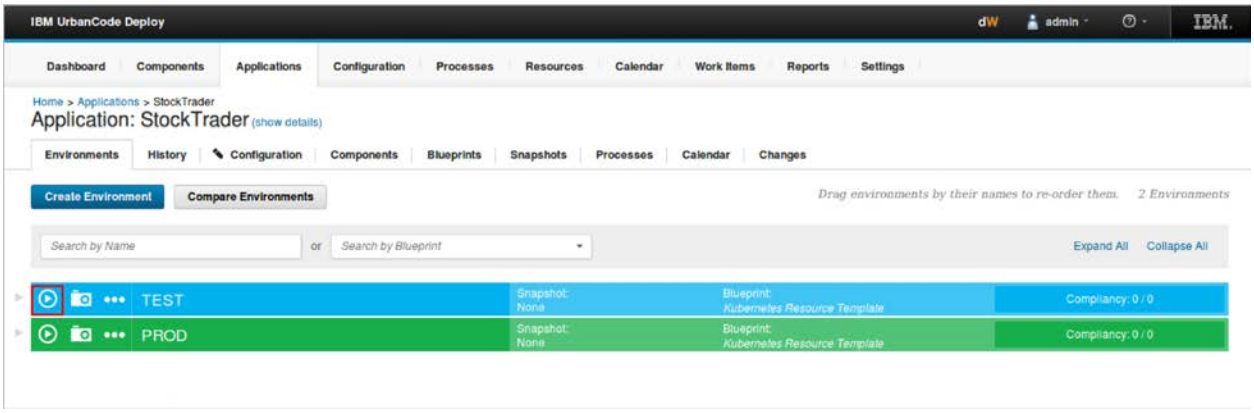
Deploy the application to the test environment

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

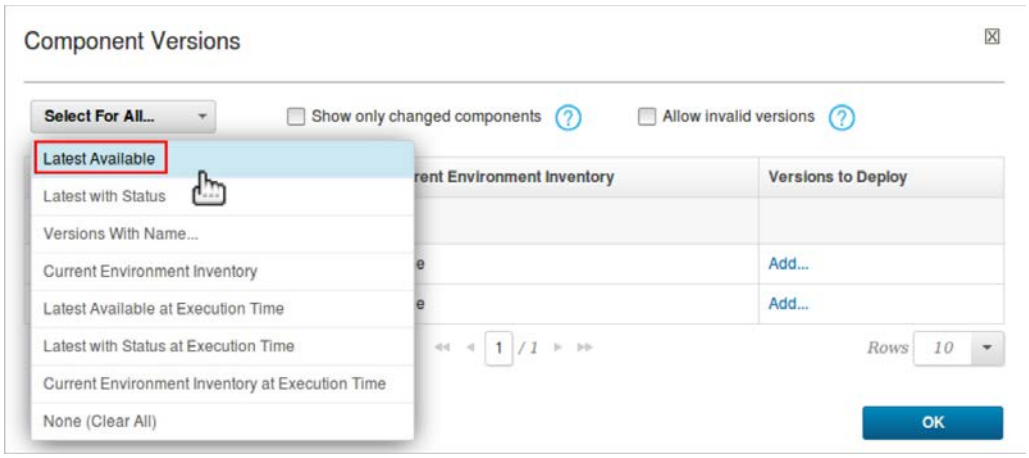
Choose the component versions and run the application process

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

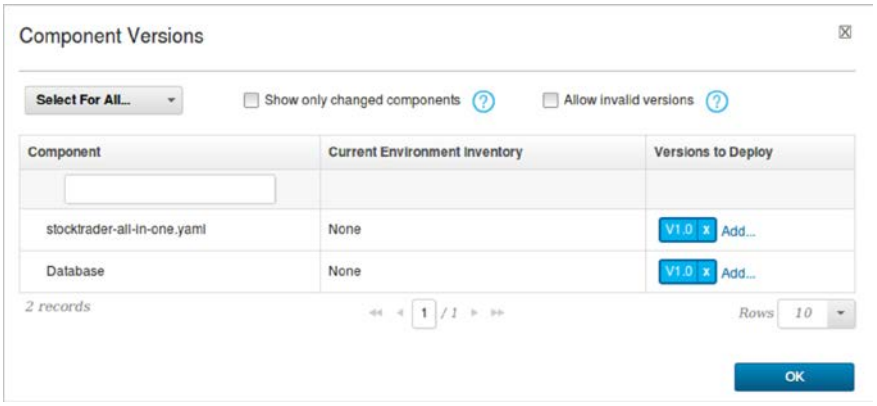
1. Click **Applications**, and then click **StockTrader**.
2. Click the play button to the left of the TEST environment:



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**.



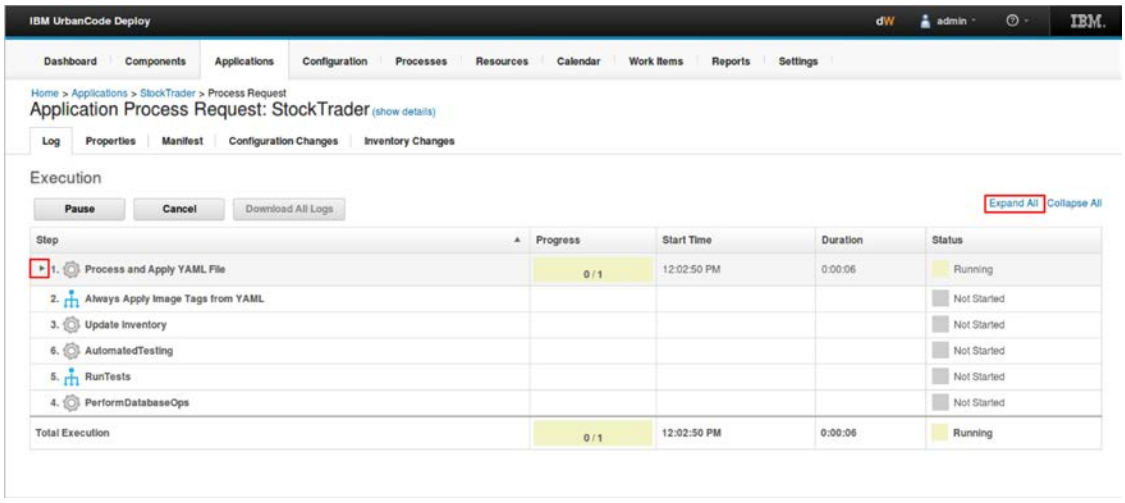
The latest versions are added for deployment.



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.

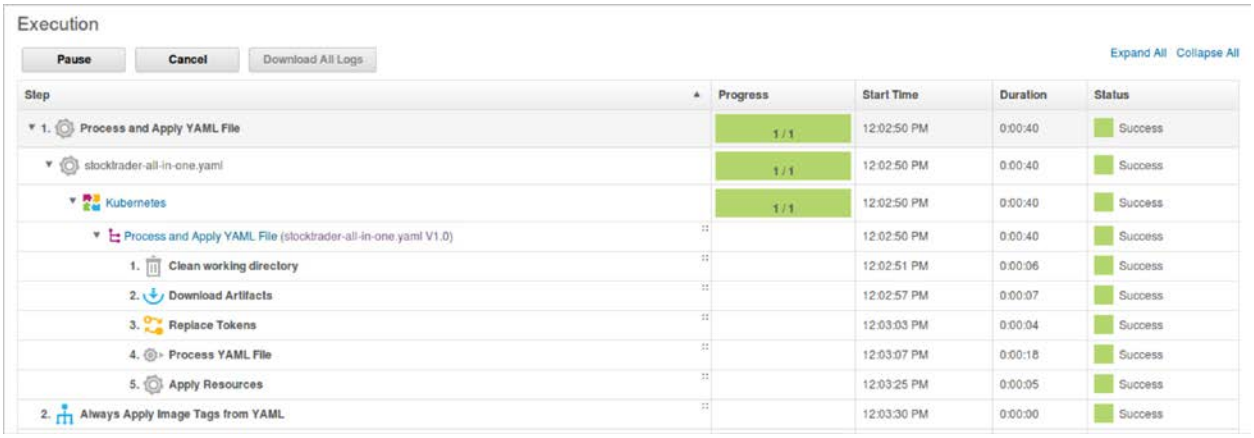
Note: Steps might display out of order, but they still process in the correct order.

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.



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

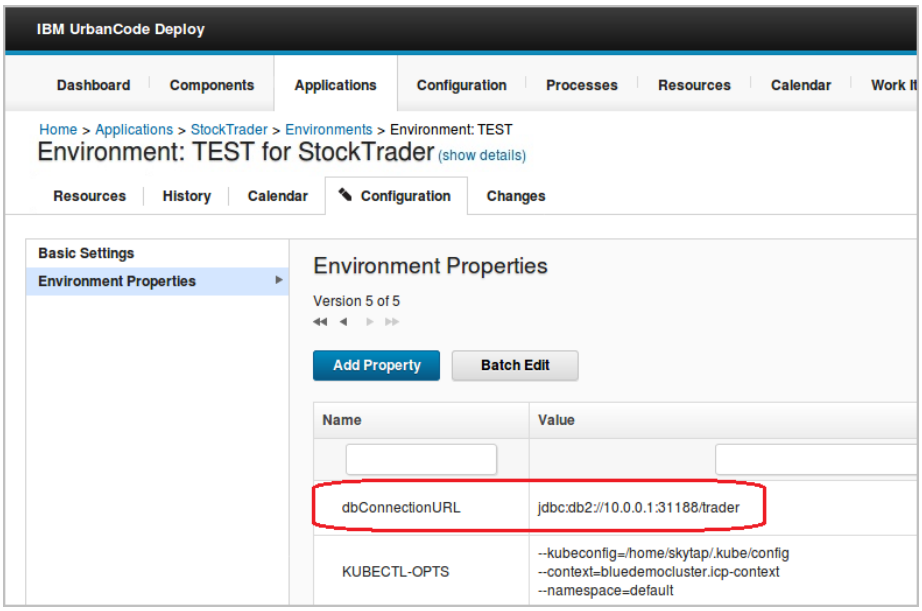
In the output log, notice the property named, *connectionString*. This property is used to determine the database to connect to.

Output Log

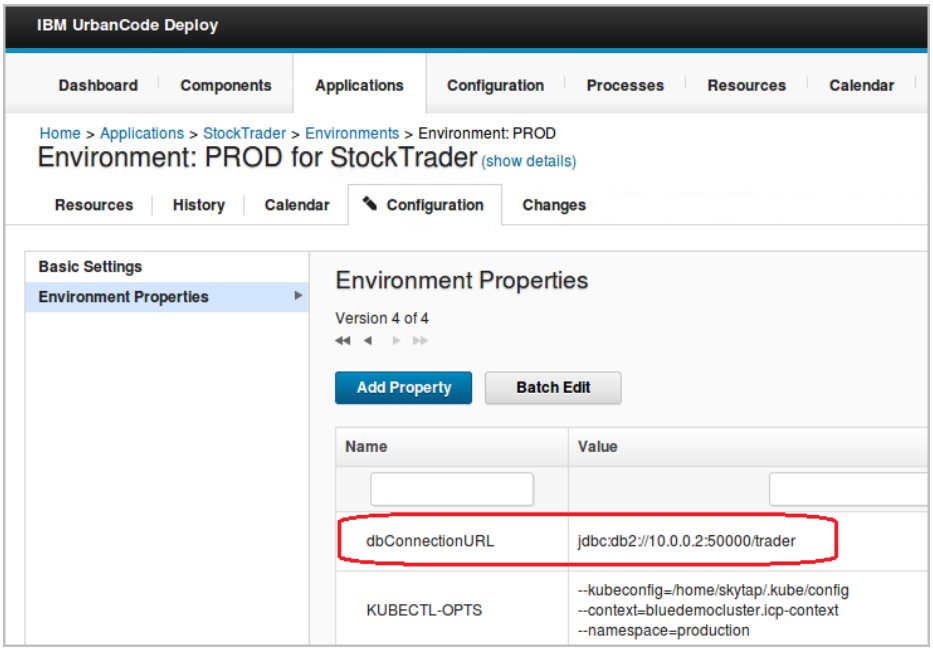
Working Directory /opt/ibm-ucd/agent/var/work//Database

```
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.



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



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 and 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:

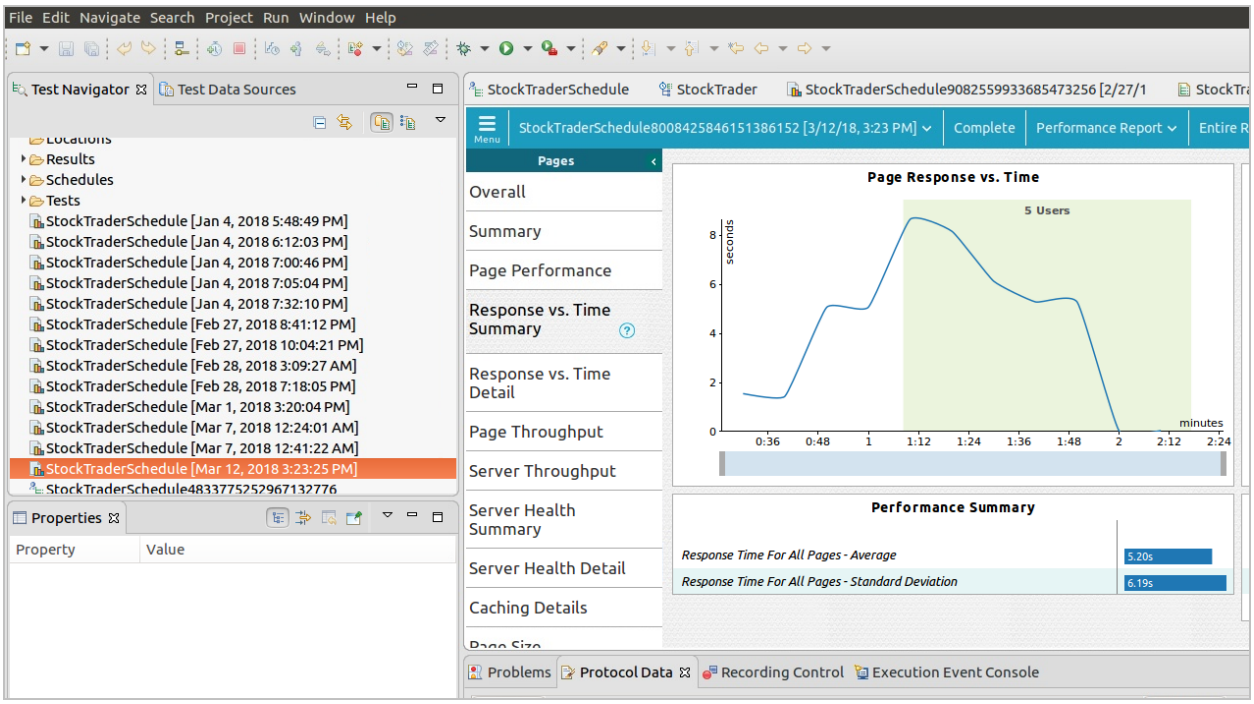
   ```
   . .bash_profile
   eclipse &
   ```

   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.

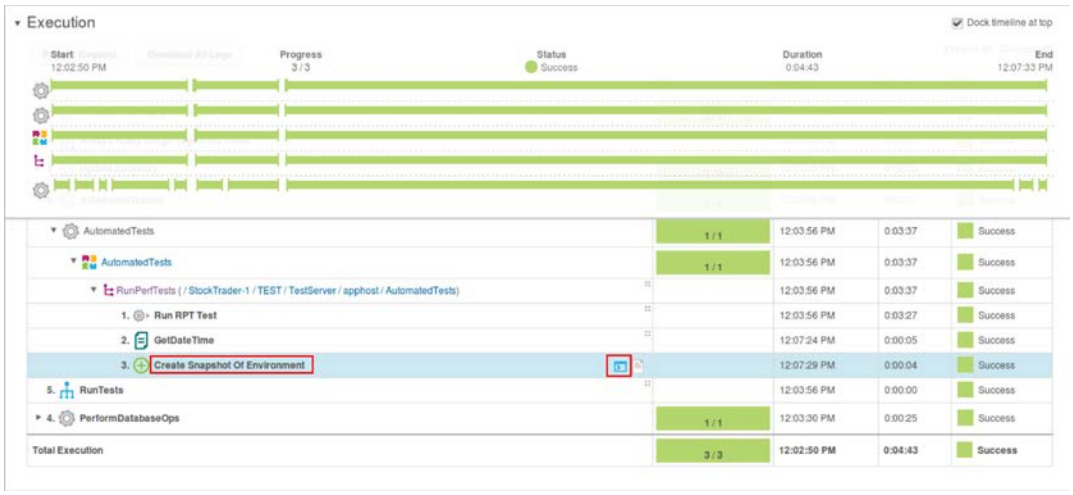
Note: When you are done looking at the results, be sure to select **File** > **Exit** to exit the Rational Performance Tester workbench; otherwise, when UrbanCode Deploy attempts to do another test run, it will fail because the workbench is already open or busy.



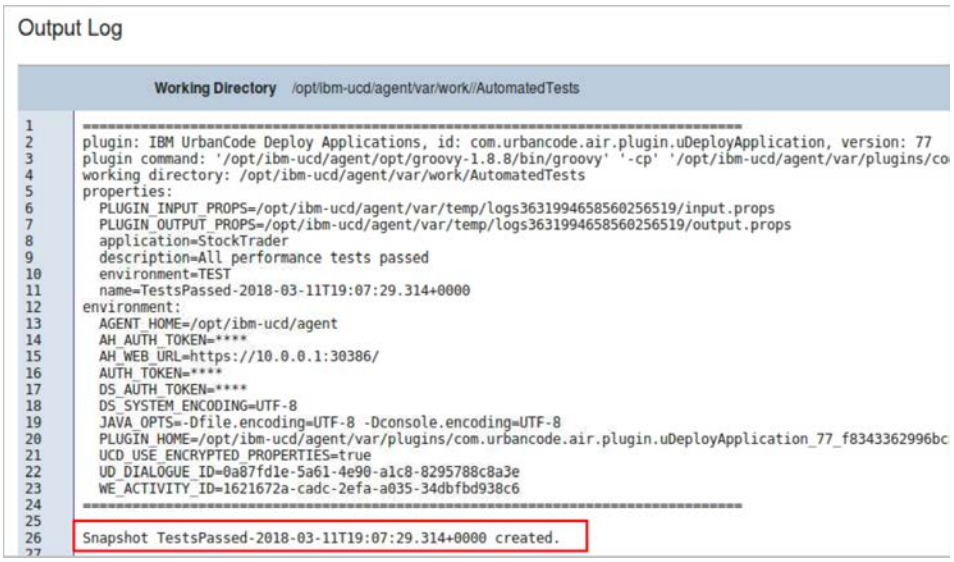
Locate the snapshot and view the deployed components

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.

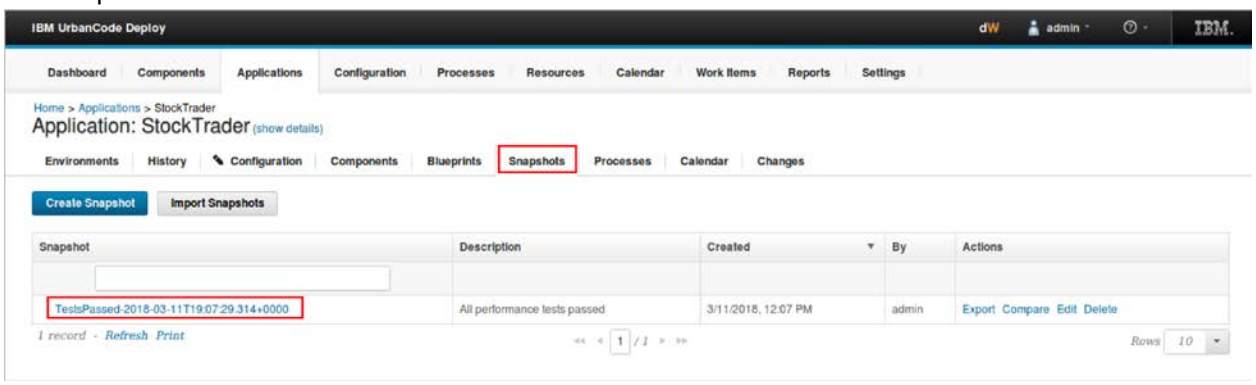


The output log shows that the snapshot was created:

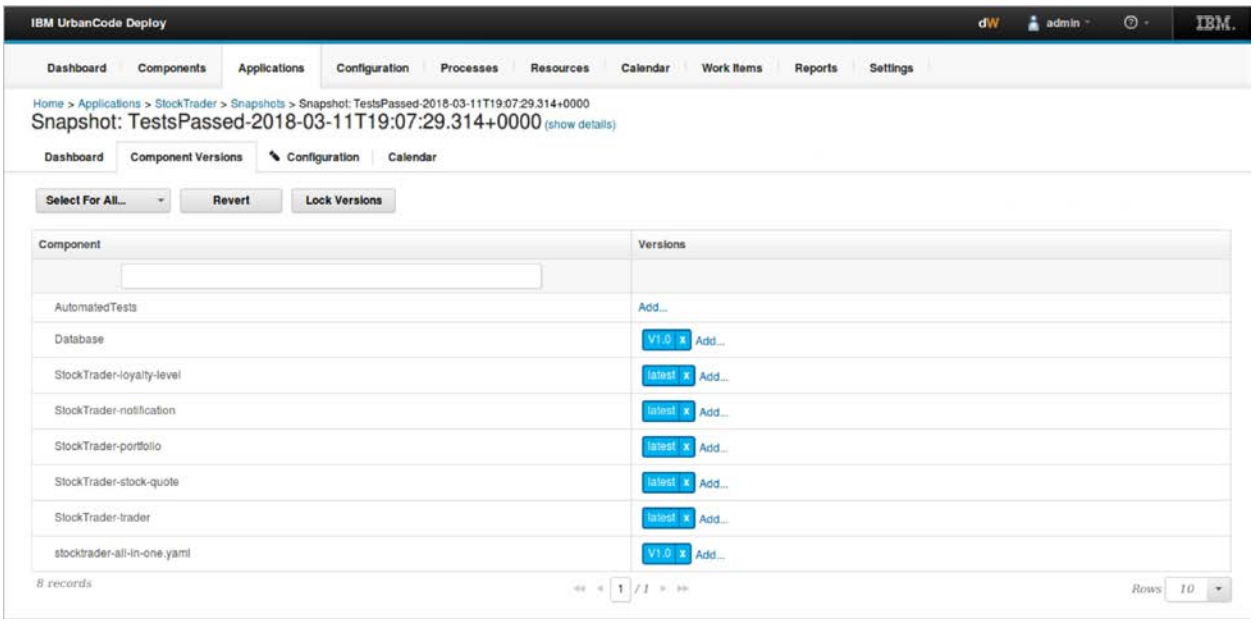


3. Close the output log.
4. Click the **StockTrader** breadcrumb link, and then click the **Snapshots** tab.

The snapshot is listed.

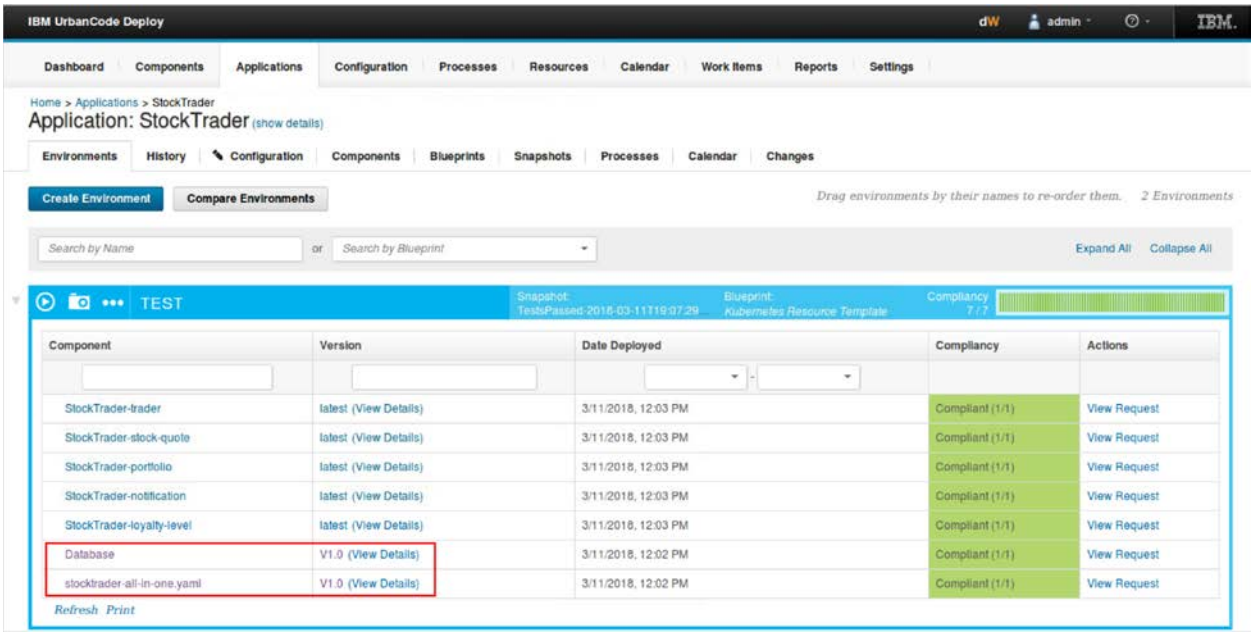


5. Open the snapshot by clicking the name.
6. Click the **Component Versions** tab. All the component versions are listed.



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

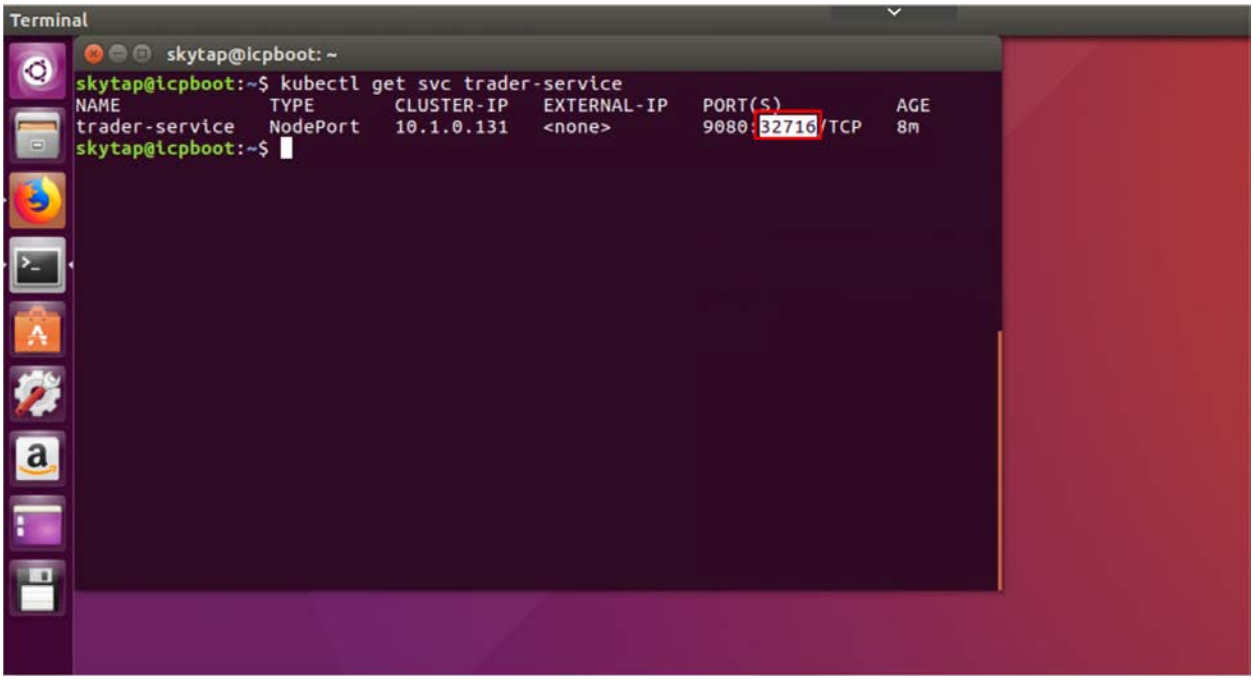
Notice that versions 1 for the database and YAML file were deployed. Also notice that five StockTrader images components were created. The Kubernetes plugin step, *Process YAML File*, created them automatically based on the image specifications found in the application description YAML file.



Verify the application in the test environment

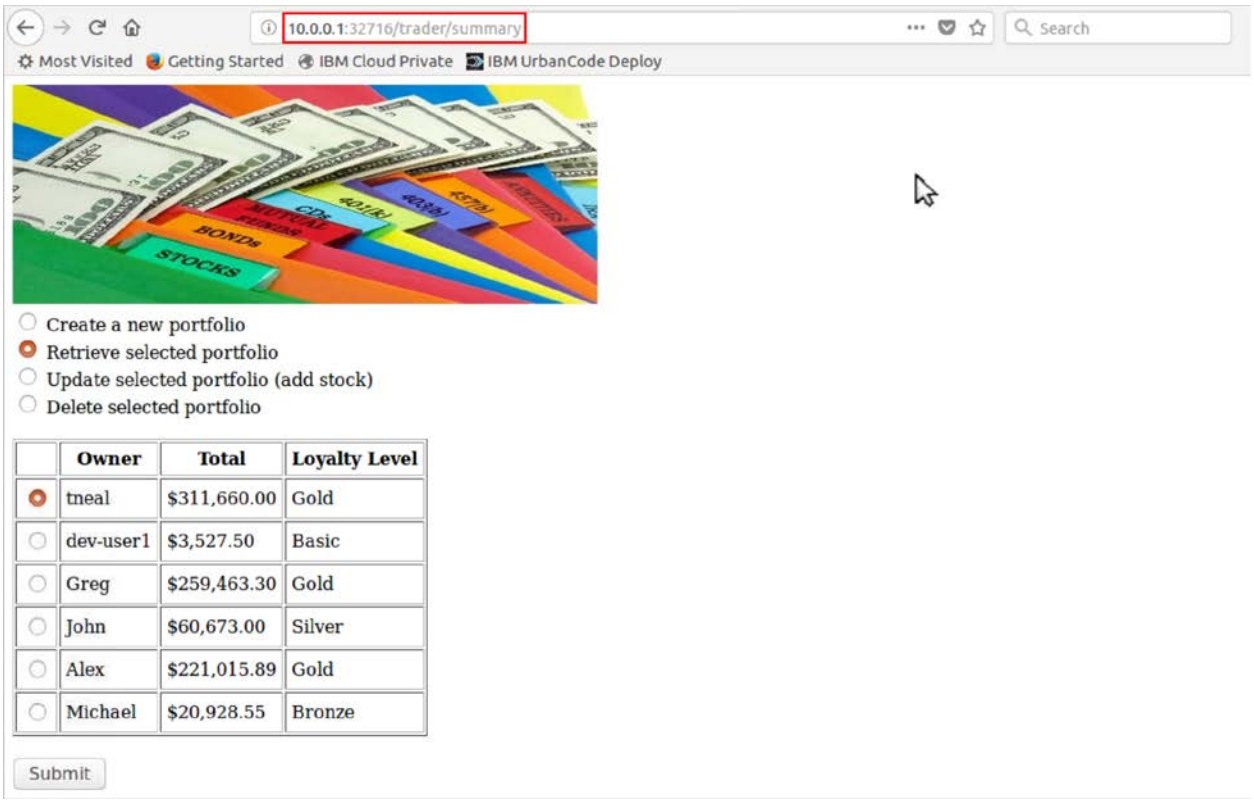
Now that the StockTrader application is deployed to the test environment, you can view the web UI that's in IBM Cloud Private. First, you need retrieve the port number for the trader-service.

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.



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.

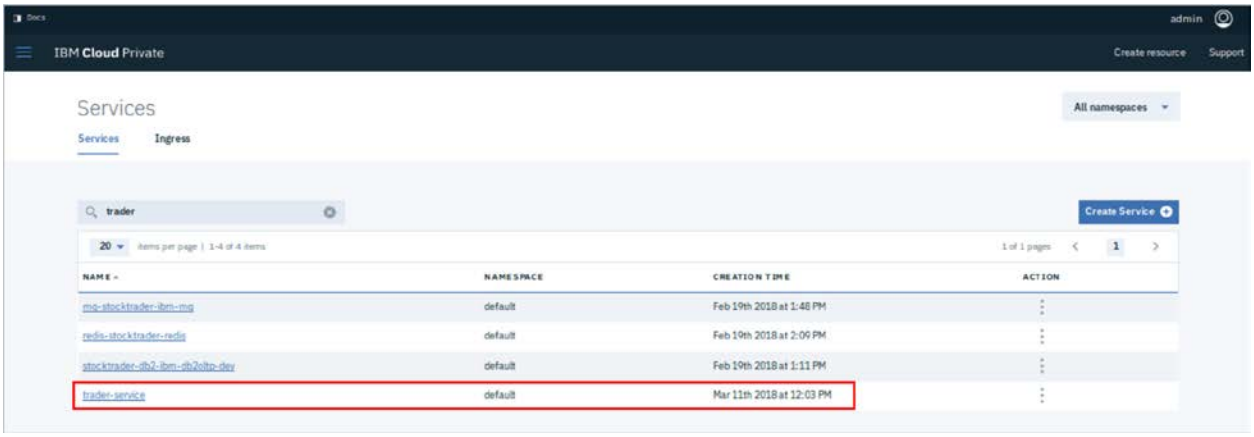
The StockTrader application displays in the browser. Test the application by creating a portfolio, adding stock to an account, and retrieving portfolio information.



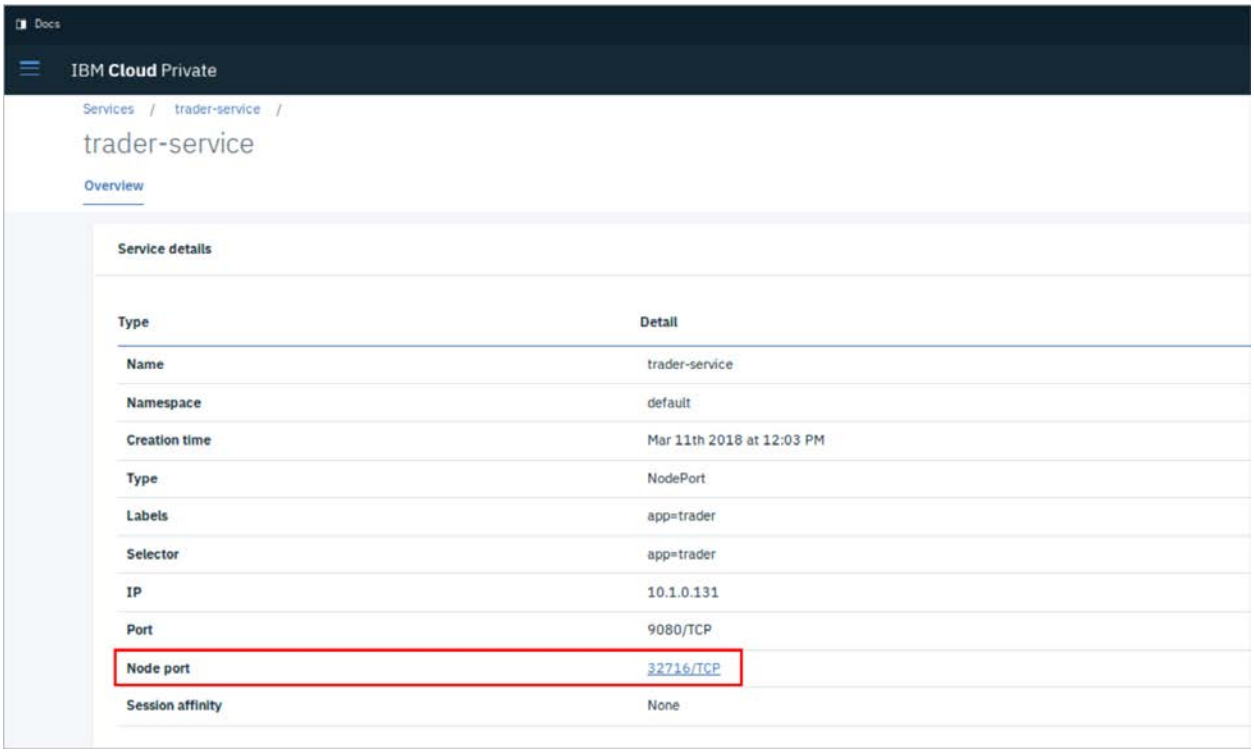
View the microservice in IBM Cloud Private

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.



Exercise 2: Use properties to set the deployment environment

In this exercise, you complete the following tasks:

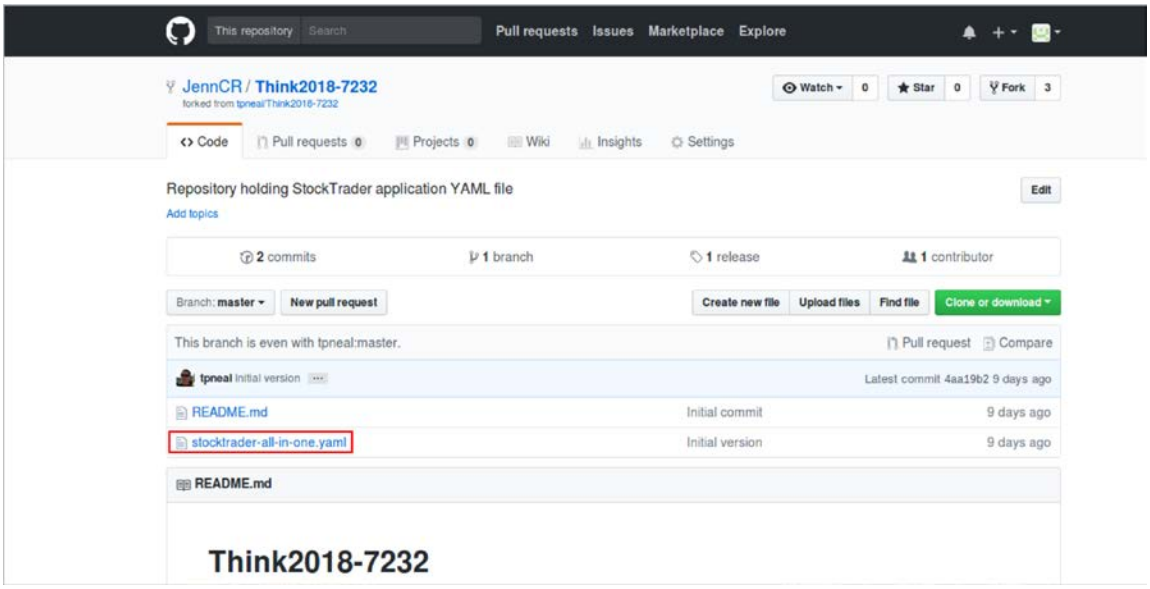
- Modify the application description YAML file to use tokens for the registry host and repository part of the image specification
- Create a new version and publish a release in GitHub
- Deploy the parameterized file to the test environment

Create a new version and publish a release

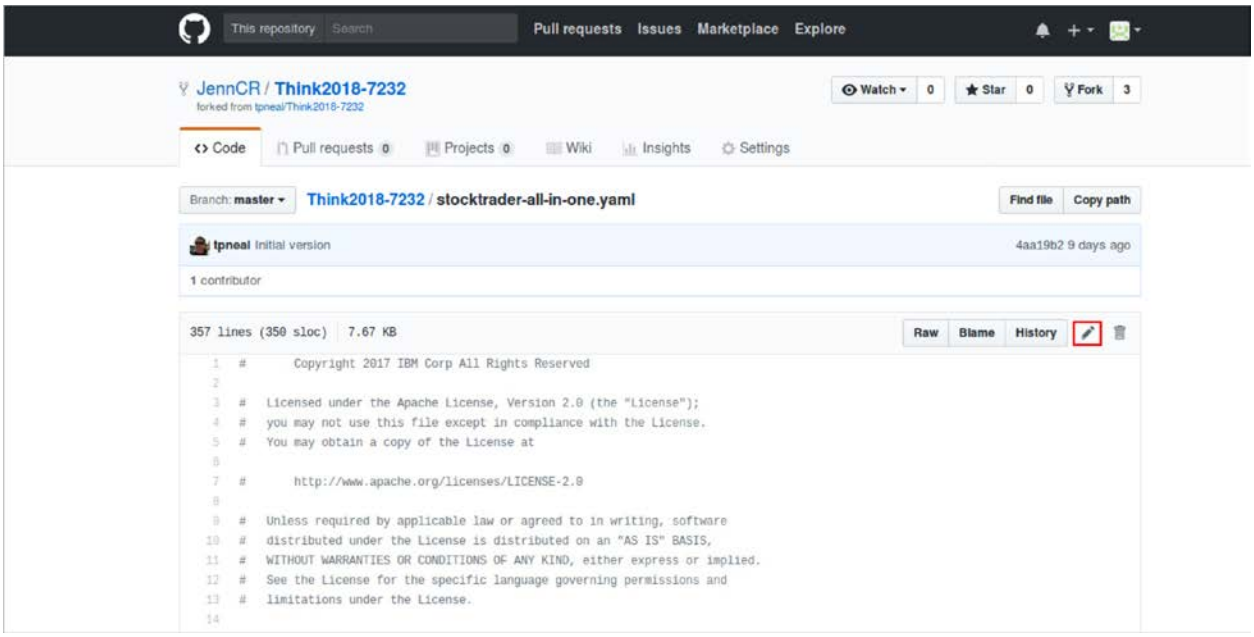
After you modify the application description YAML file in GitHub, you publish a release with those changes. Then, you import the new version into UrbanCode Deploy.

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.

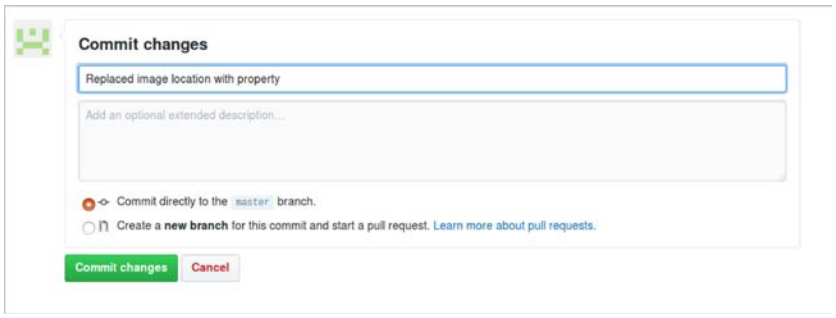


For each of the five StockTrader images, you substitute the registry host and repository with a token. Adding a token parameterizes the application description YAML file, which makes it simpler to deploy to different environments without changing the YAML file each time. Instead of hard-coding the image 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@`.

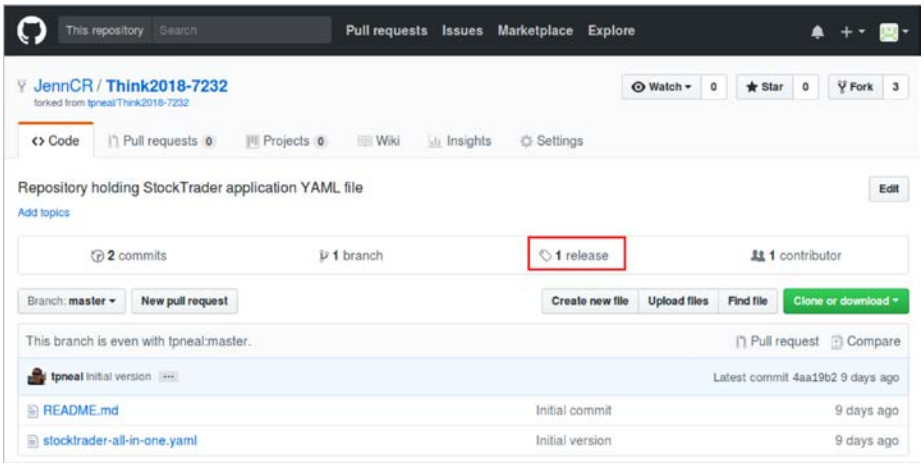
```
7   #       http://www.apache.org/licenses/LICENSE-2.0
8
9   #   Unless required by applicable law or agreed to in writing, software
10  #   distributed under the License is distributed on an "AS IS" BASIS,
11  #   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12  #   See 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**.

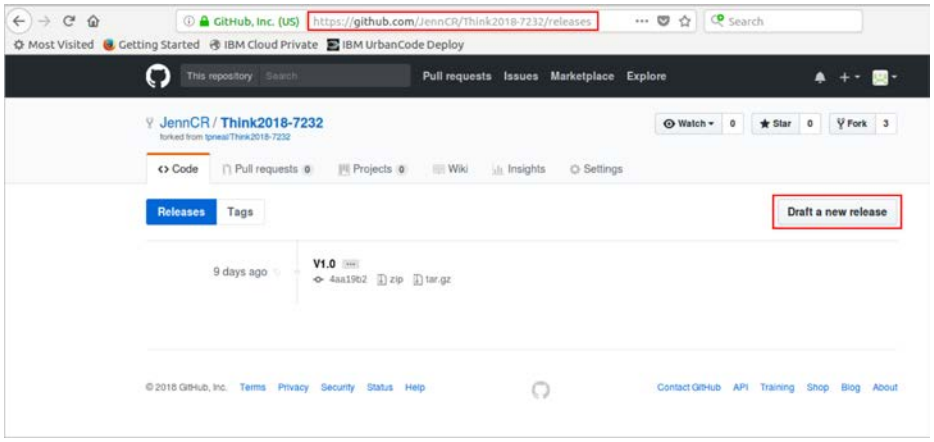


Create and publish a release

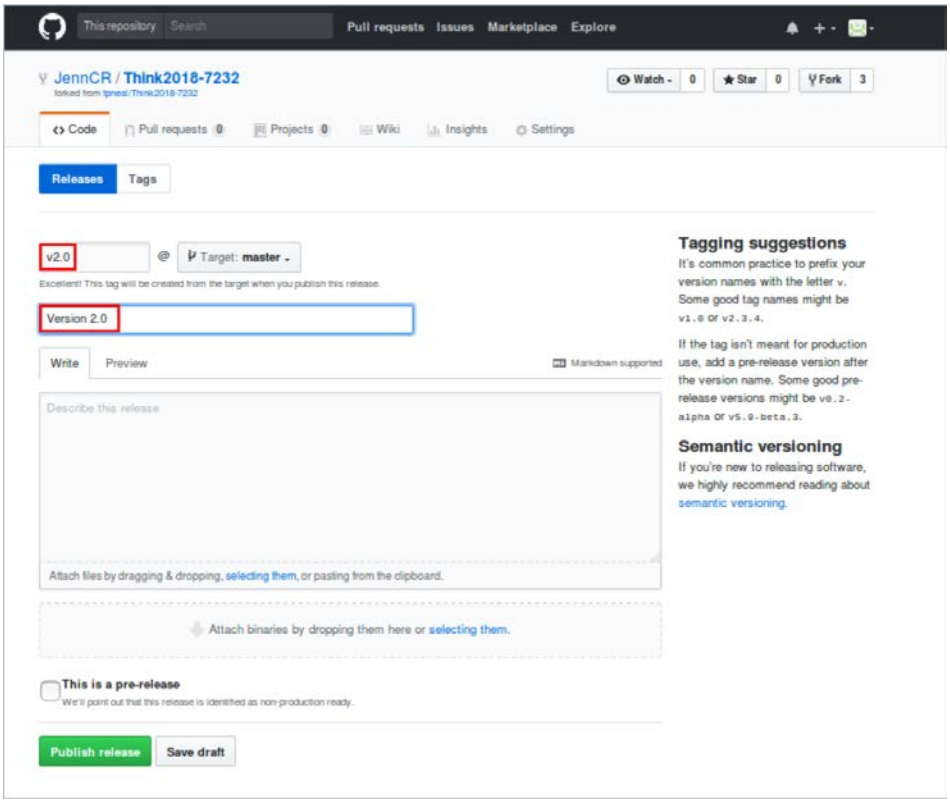
1. Click **Release** to open the releases page.



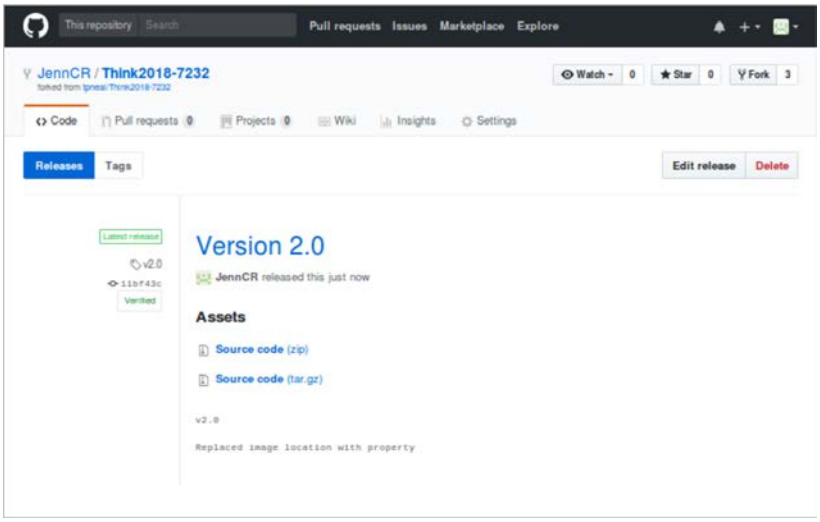
2. Click **Draft a new release**.



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



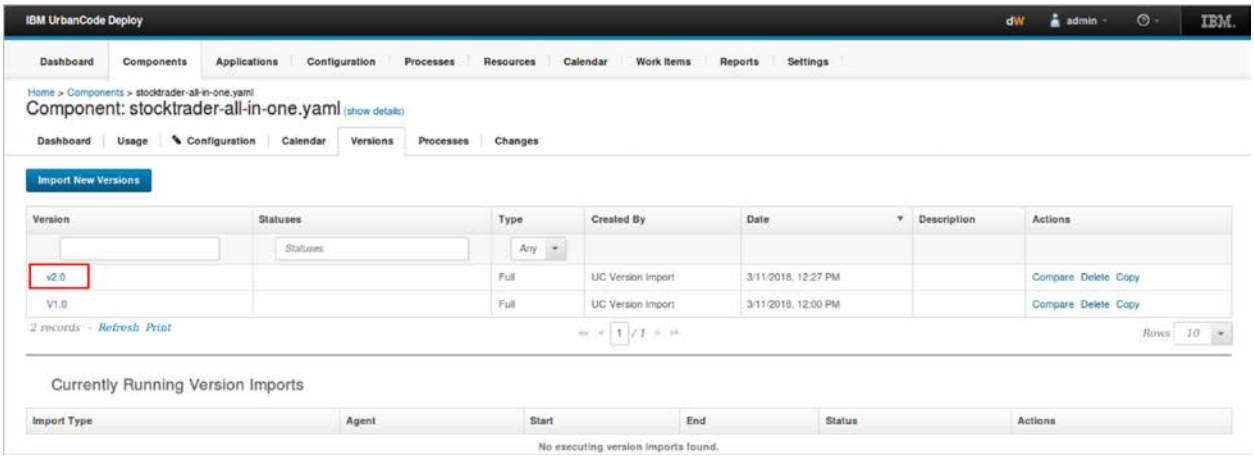
Now you have a new version that incorporates the updated parameter.



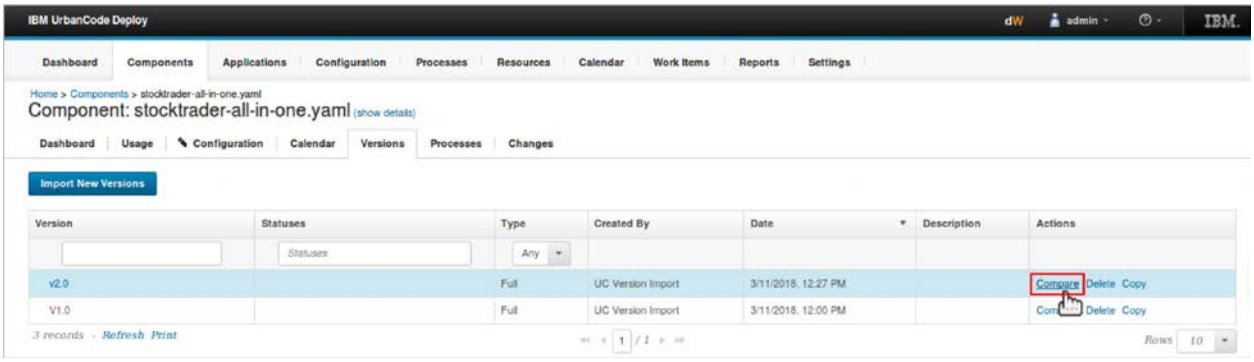
Import the parameterized version into UrbanCode Deploy

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**.

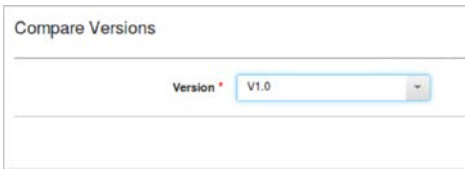
The new version is created with the updated parameter.



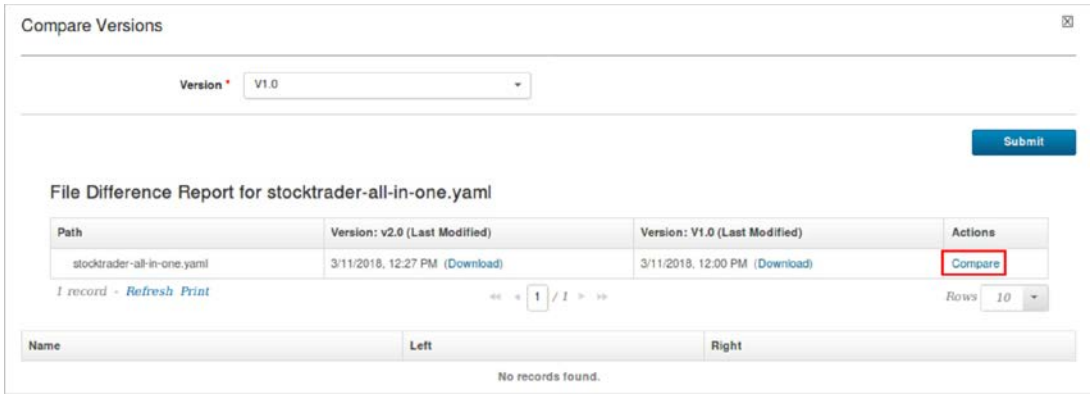
4. Compare the changes between files by clicking **Compare** next to a version.



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



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



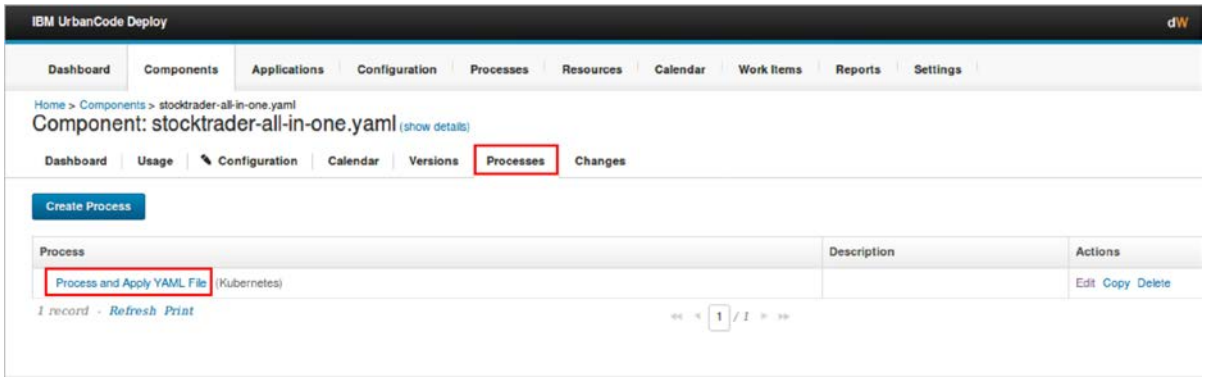
View the changes. You can see the substitutions you made in Version 2.0.

7. Close the *File Differences* window.

View the Replace Tokens step in the component process

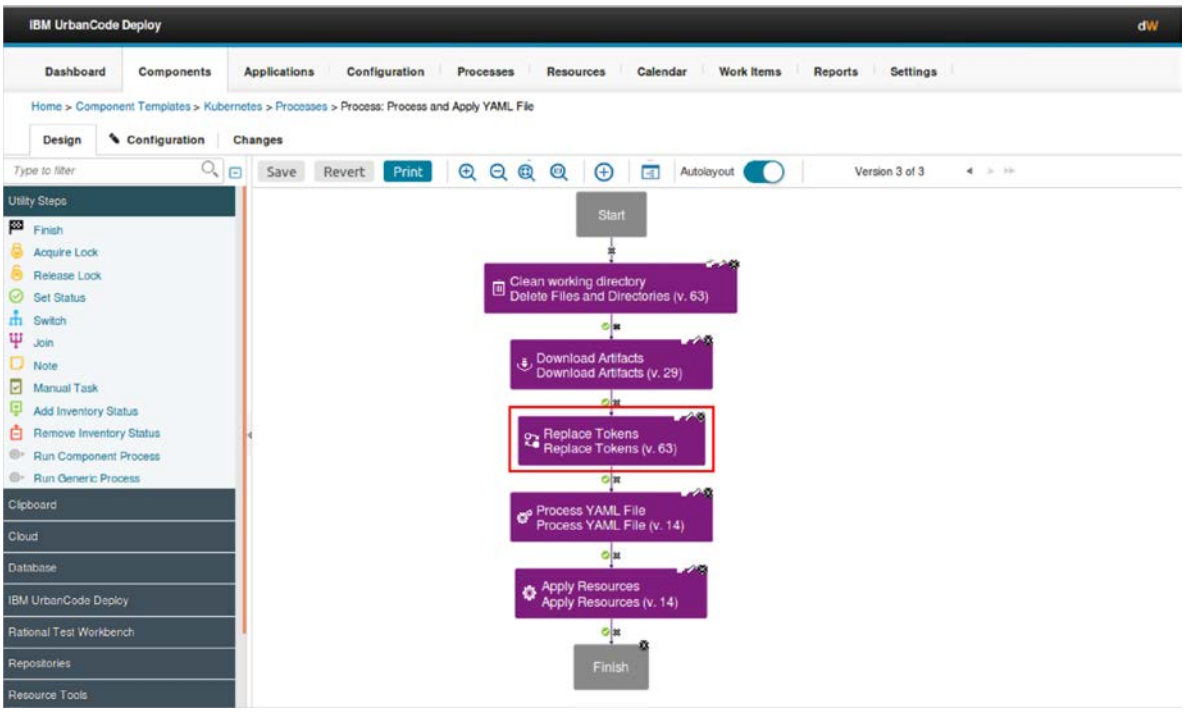
Processes are lists of steps and connections between those steps. Each step is an individual command that runs on a target computer. Steps can manipulate files, run system commands, download files, and run programs.

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

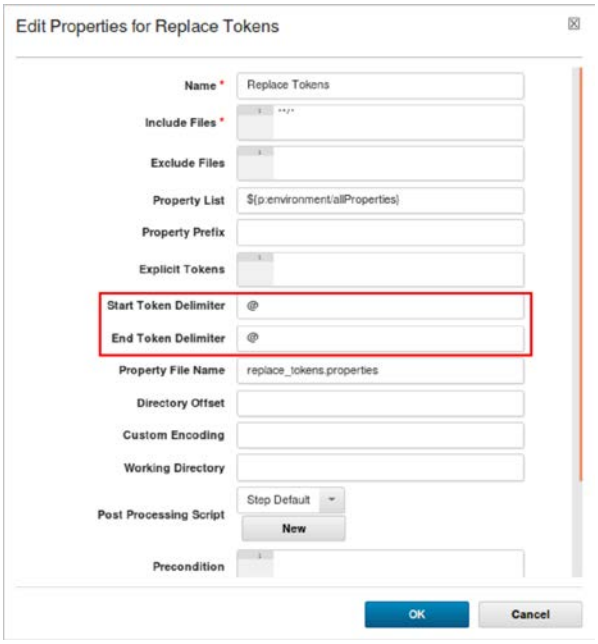


The process displays in the process editor. The process editor allows you to organize the steps in a process, specify their properties, and connect them to each other.

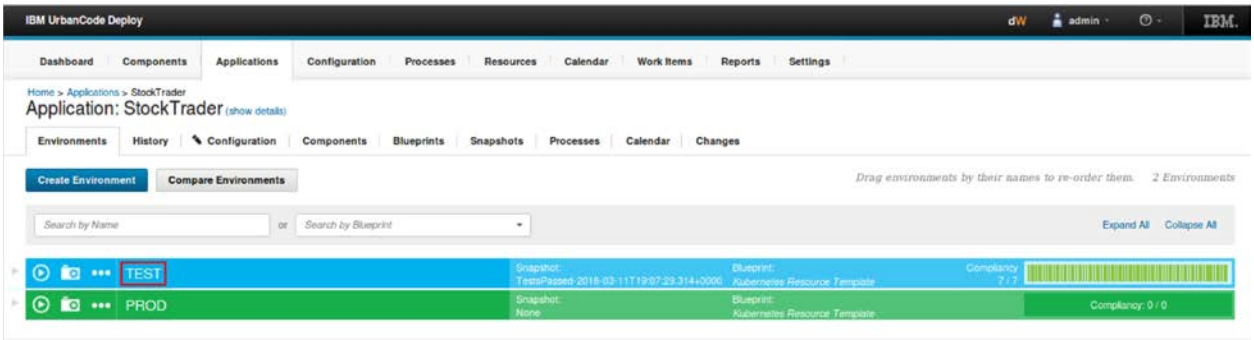
2. Click the **Edit** icon (looks like a small pencil) on the upper-right corner of the *Replace Tokens* step.



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

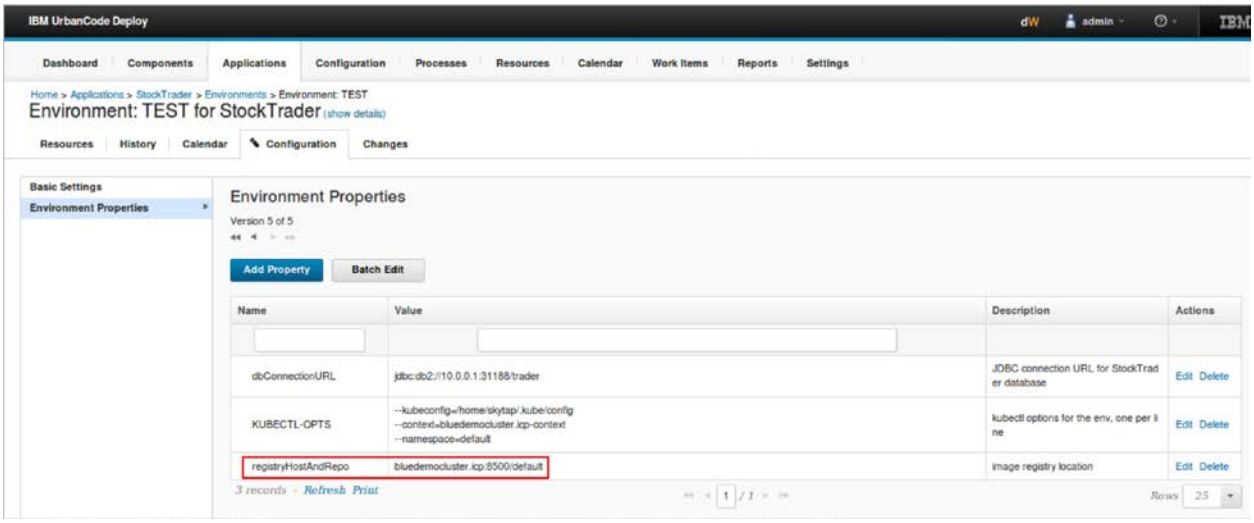


4. Click **Cancel** to exit the window.
5. Return to the application by clicking the **Applications** tab, and then clicking **StockTrader**.
6. From the **Environments** tab, click **TEST** to view the environment information.

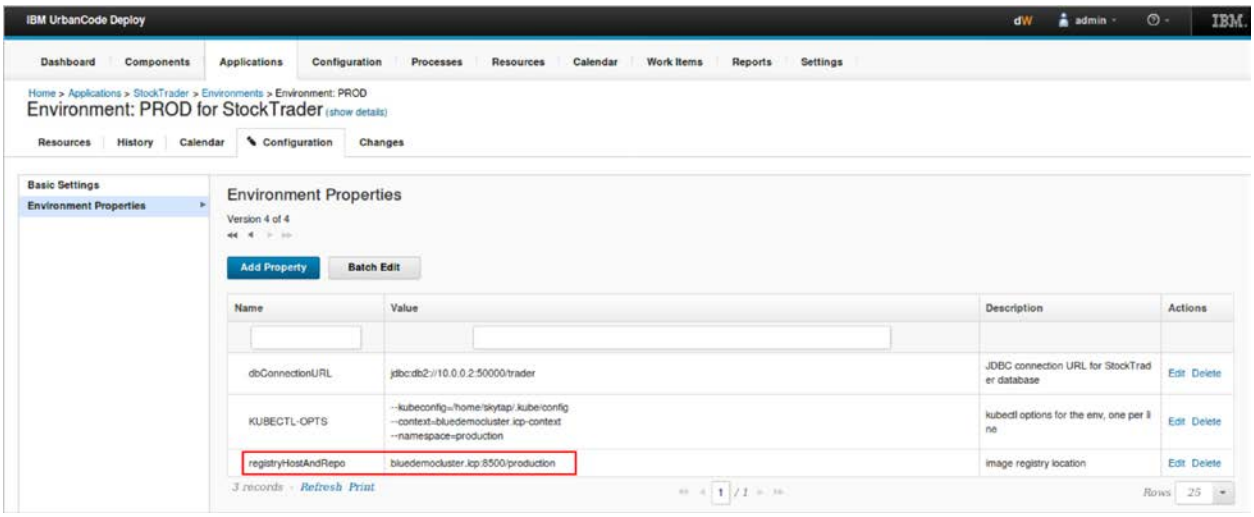


7. Click the **Configuration** tab, and then click **Environment Properties**.

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



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



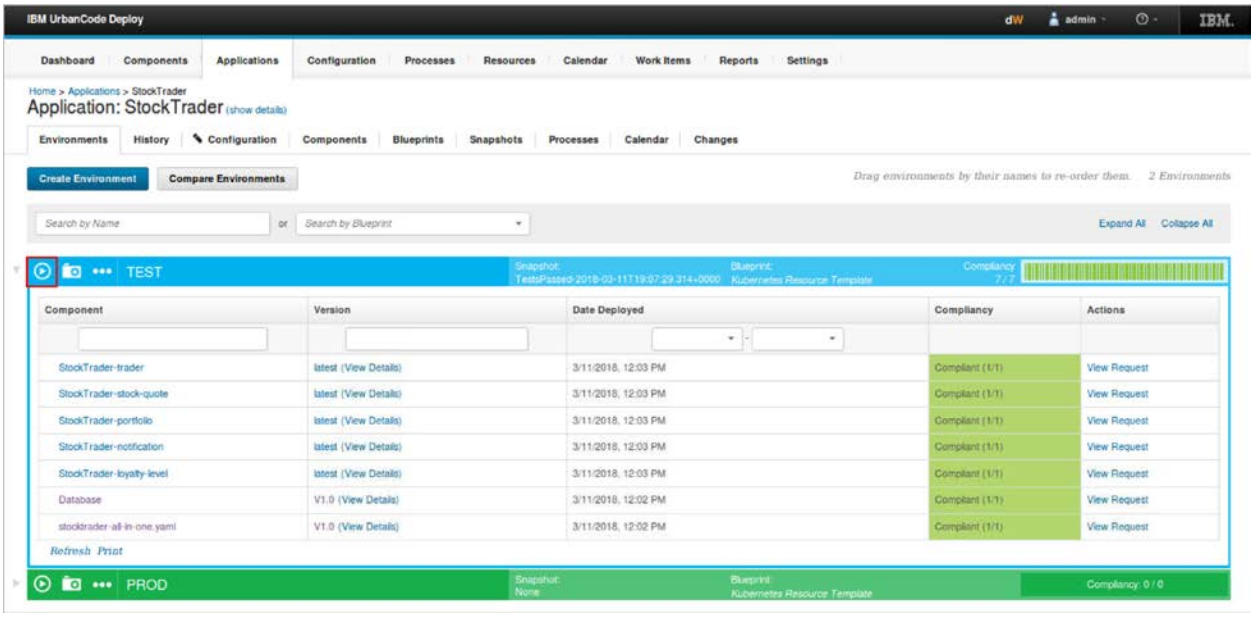
Deploy the parameterized YAML file to the test environment

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.

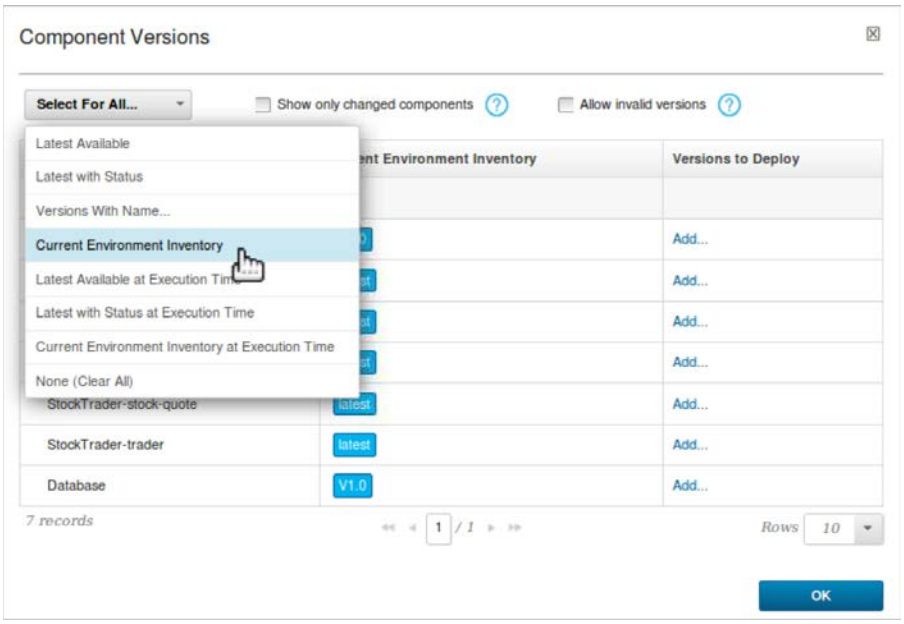
Choose the component versions to deploy and run the application process

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

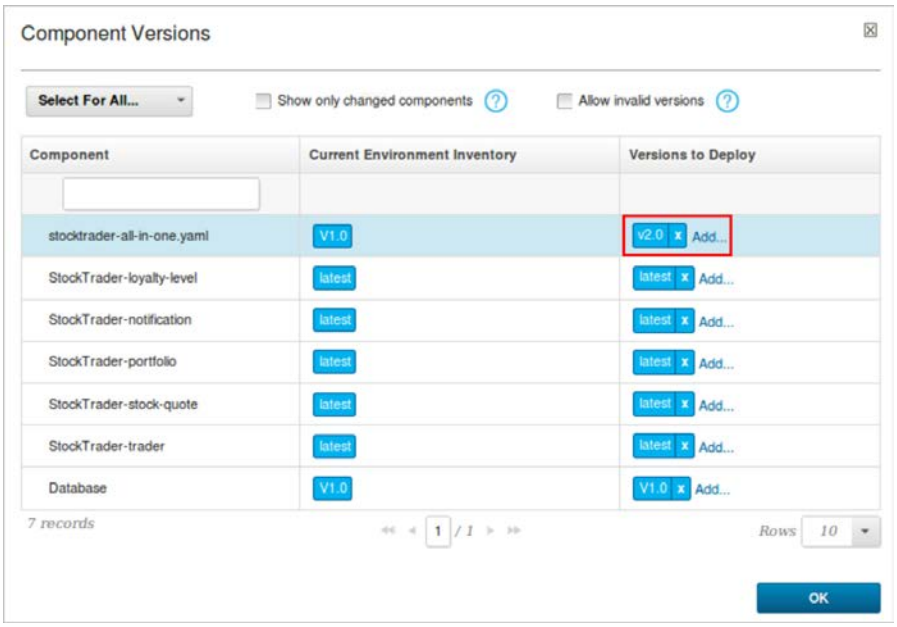
1. Return to the **Environment** tab.
2. Click the play button to the left of the TEST environment.



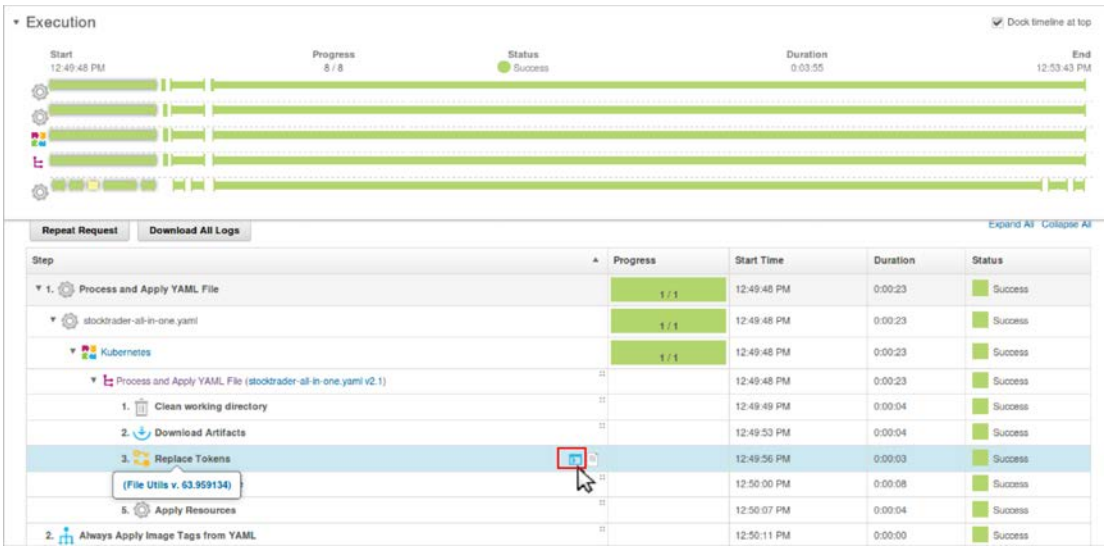
3. In the *Run Process on TEST* window, click **Choose Versions**.
4. In the *Component Versions* window, click **Select For All**, and then click **Current Environment Inventory**.



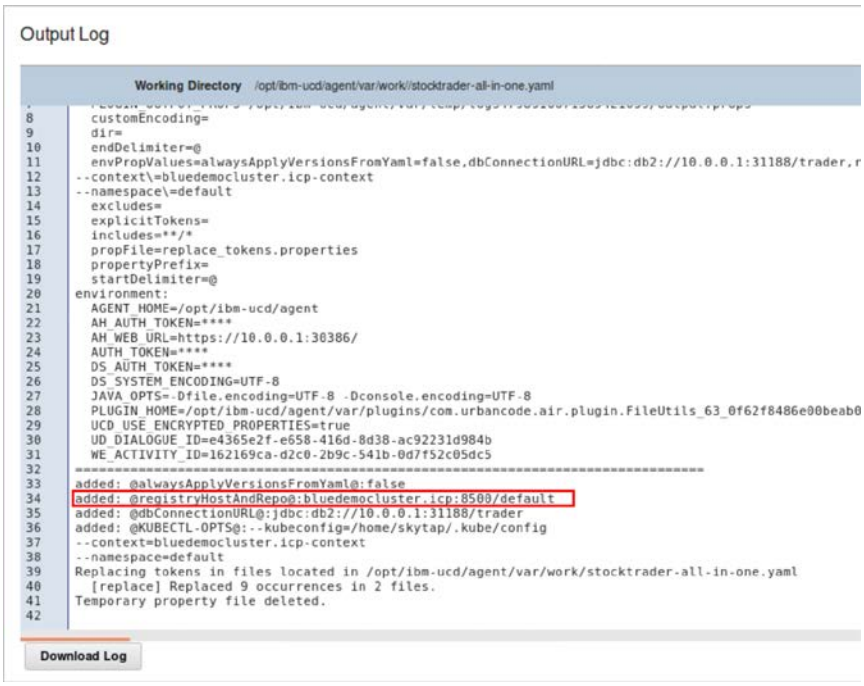
5. Under the *Versions to Deploy* column in the *stocktrader-all-in-one.yaml* row, delete **v1.0**, and add **v2.0**.



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

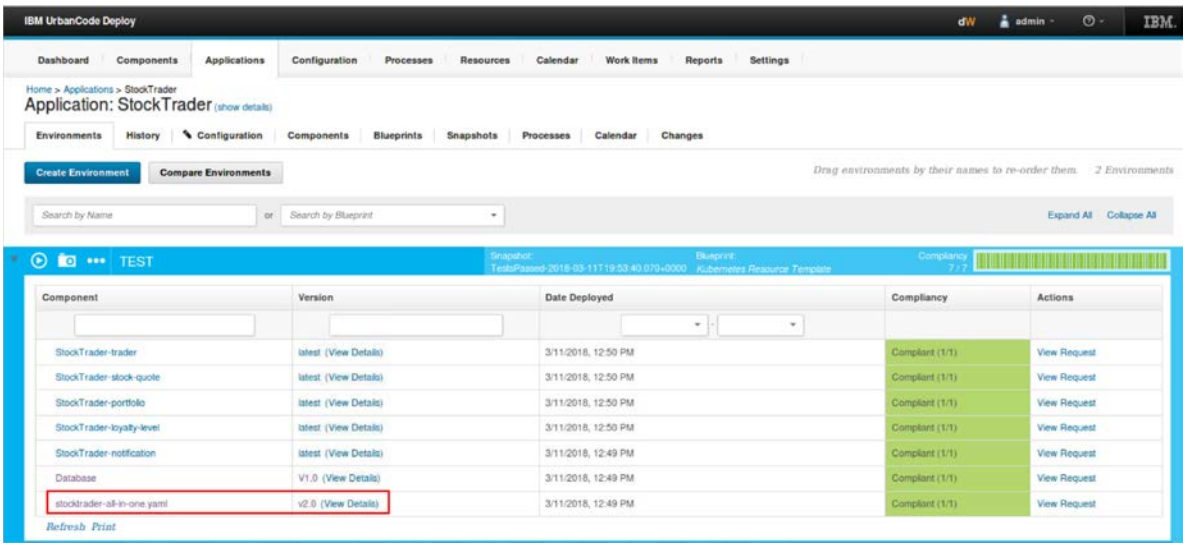


The *Replace Tokens* step looks for properties in between the two ampersands (@). You can see it found several tokens, including the one you added, `@registryHostAndRepo@`, which replaced the property with the test location of `bluedemocluster.icp:8500/default`.

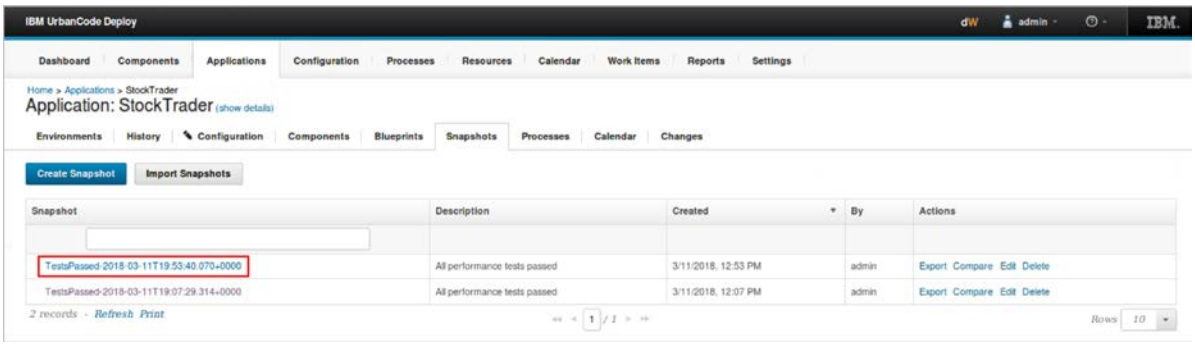


8. Close the output log.
9. Return to the **Environment** tab by clicking the **StockTrader** breadcrumb link.
10. Expand the test environment.

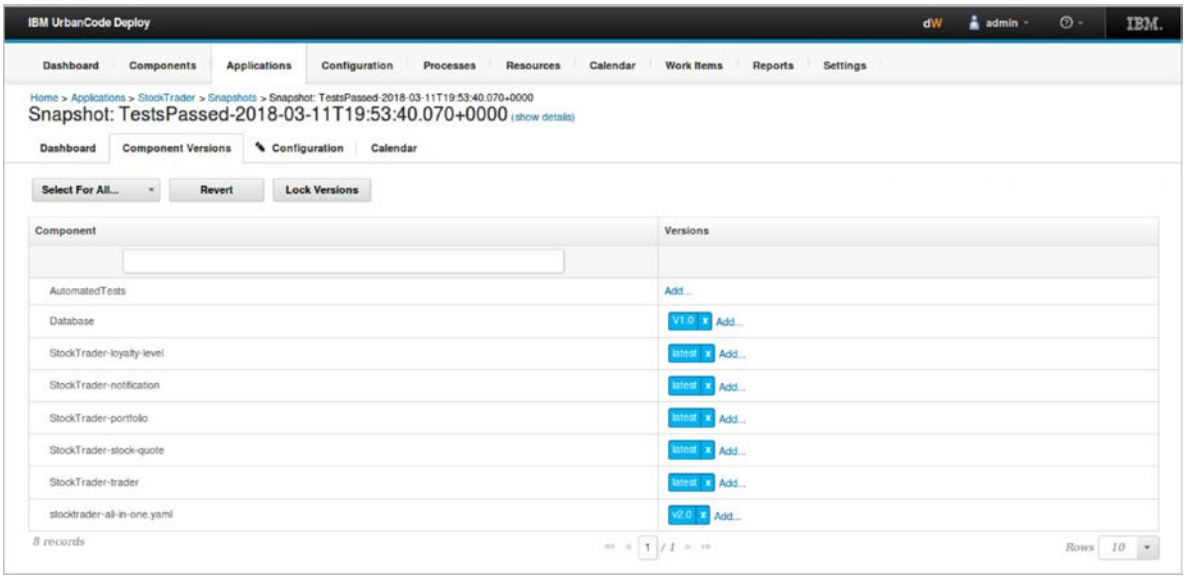
Notice the latest StockTrader images and the new version of the YAML file that was deployed.



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**.



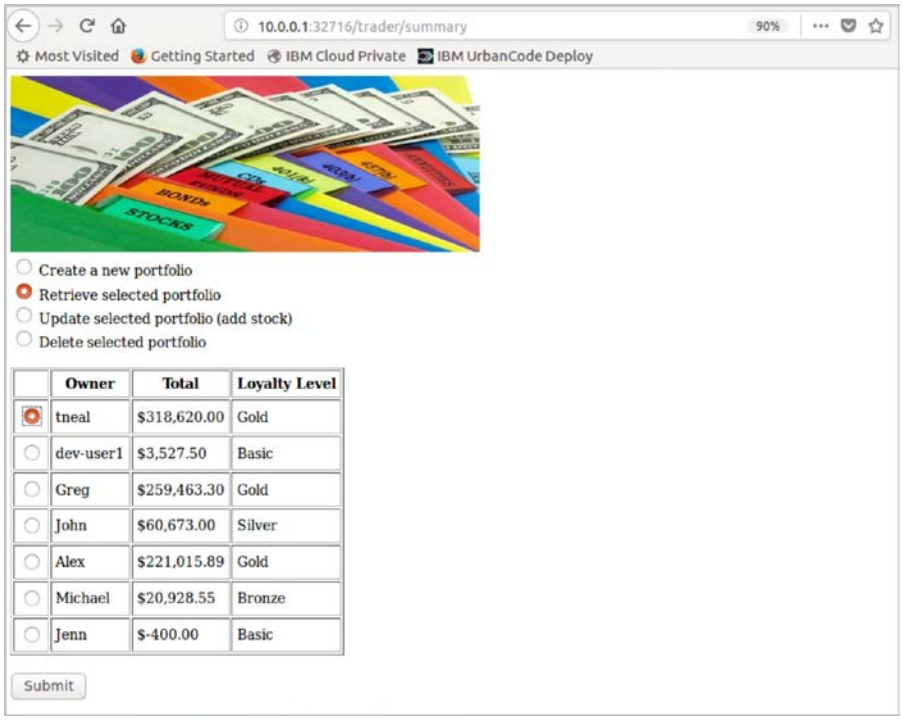
Notice that version 2 of the *stocktrader-all-in-one.yaml* file is captured in the snapshot.



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.



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

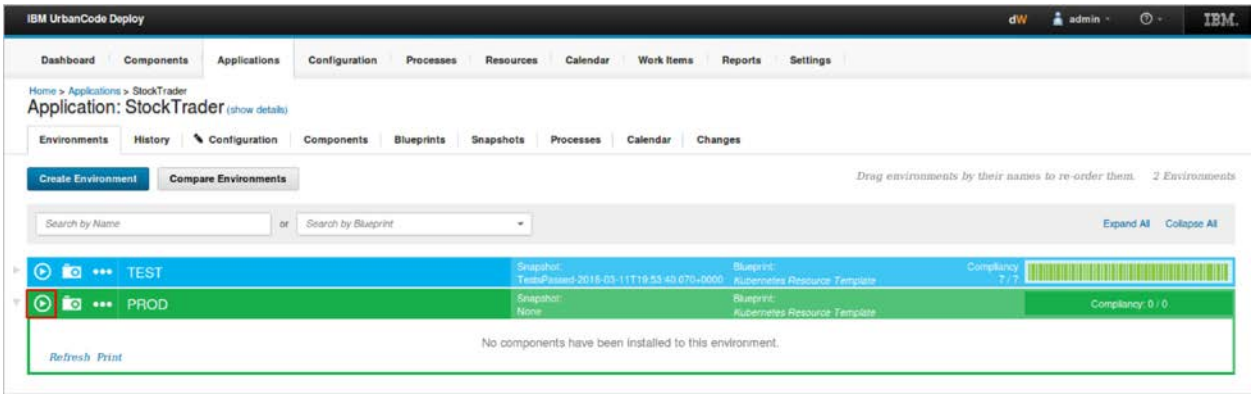
Exercise 3: Deploy the snapshot to the production environment

In this exercise, you complete the following tasks:

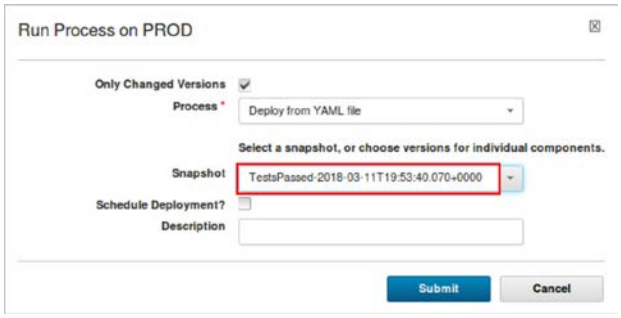
- Deploy the snapshot, include the parameterized version of the YAML file, to the production environment
- Verify the application in the web UI
- View the production version of the microservice in IBM Cloud Private

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.

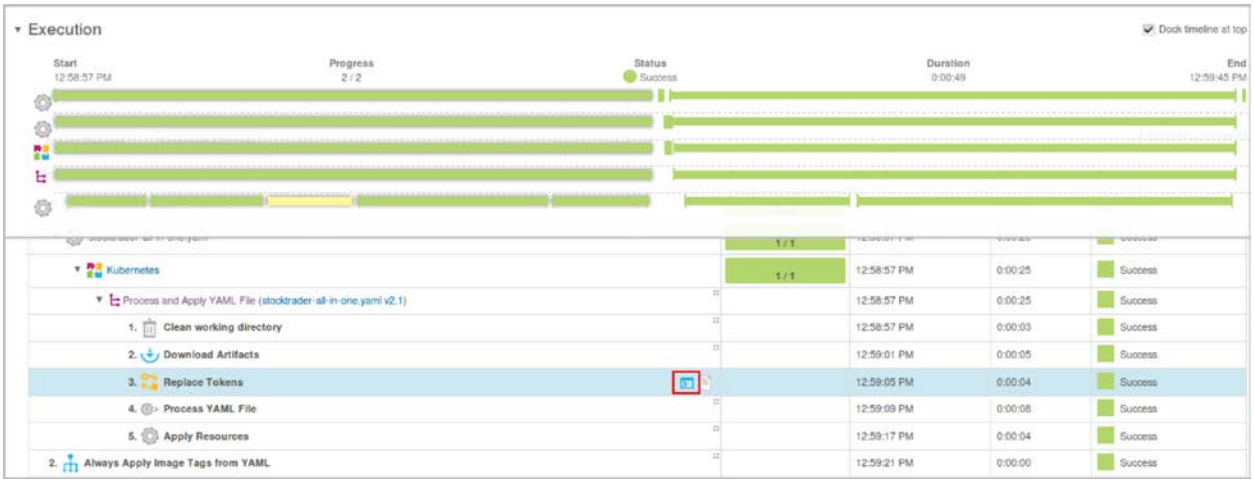


5. In the *Run Process on PROD* window, select the latest snapshot that you created, and then click **Submit**.



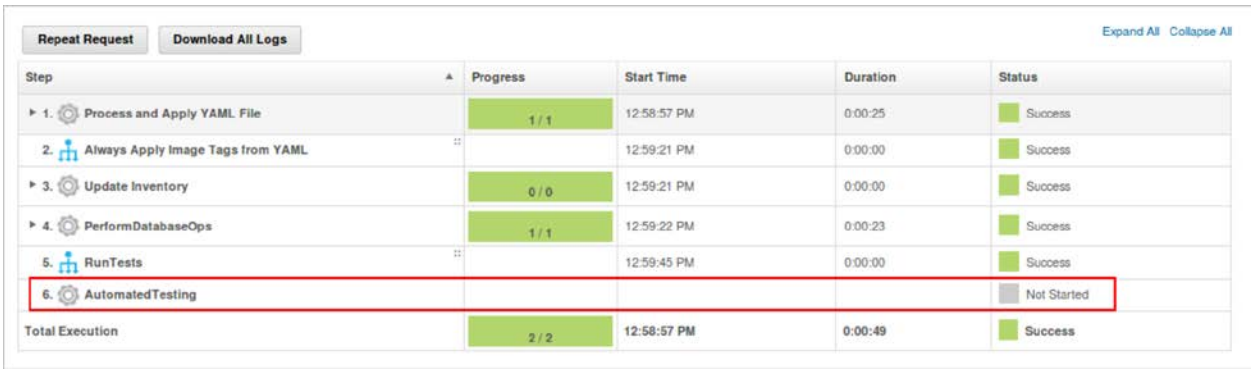
You are deploying a new instance of StockTrader into another Kubernetes namespace. You can separate environments by namespaces (like this) but you can also separate environments by cluster (one for test and one for production).

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

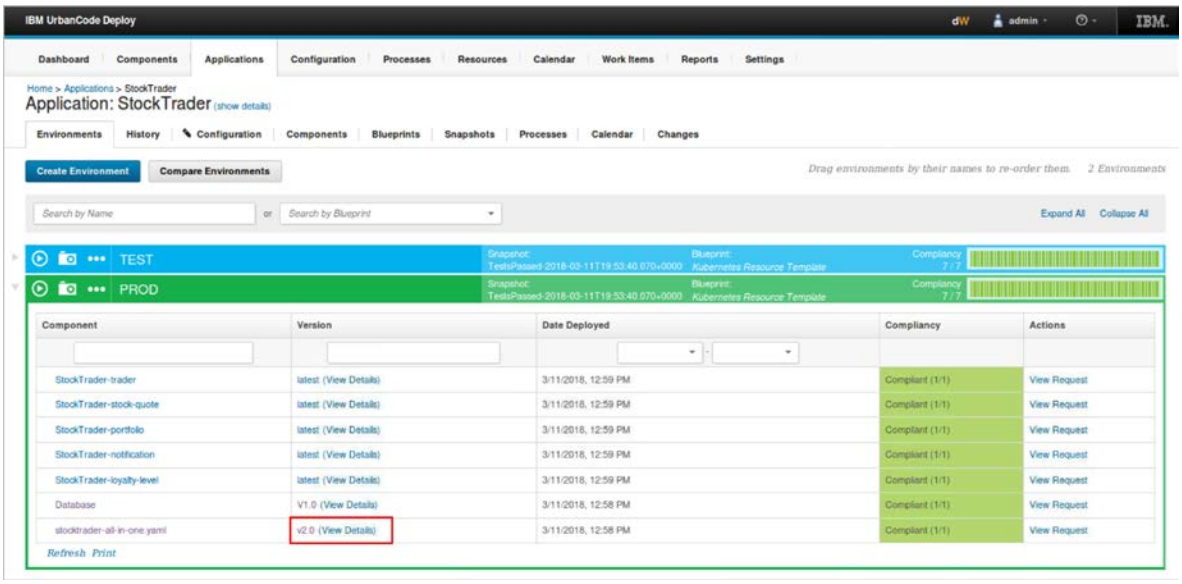


Notice the `@registryHostAndRepo@` token is replaced with a new namespace: production.

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.



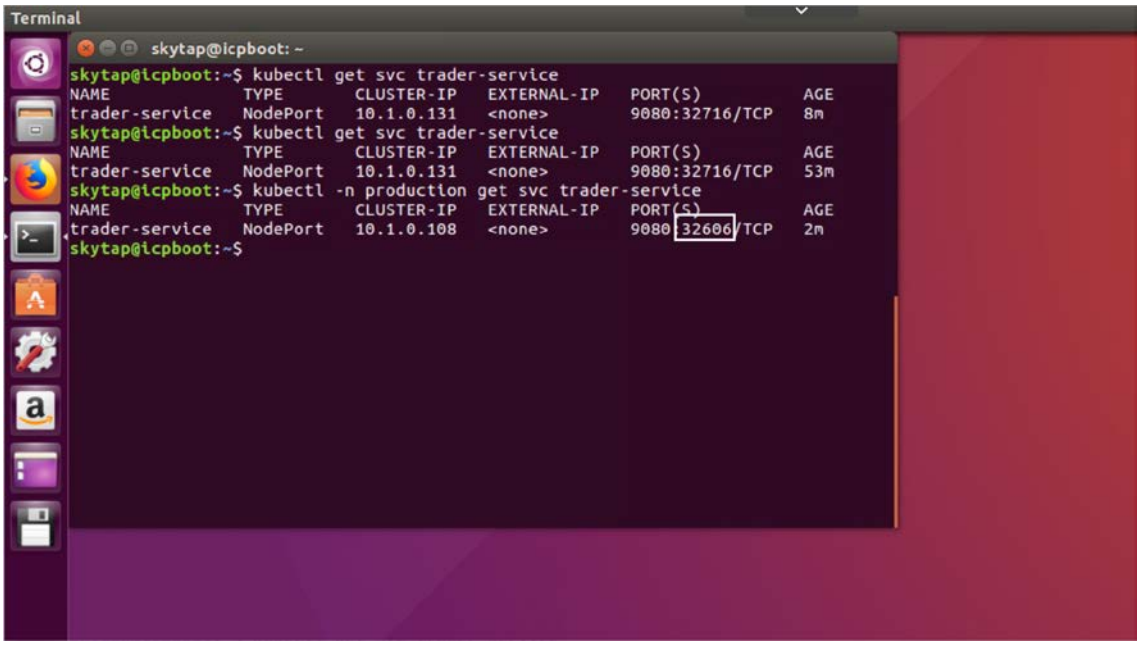
9. When the deployment is finished, click the **StockTrader** breadcrumb link to return to the Environments tab.
10. Expand the PROD environment to see the installed components:



Verify the application deployment to the production environment

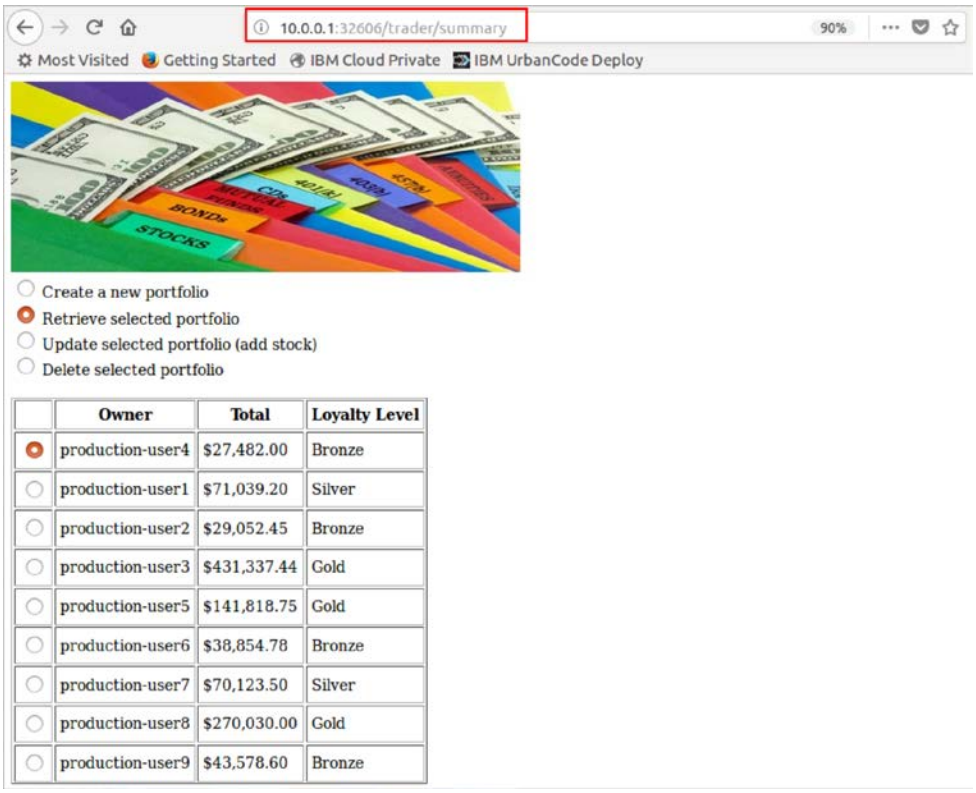
Now that the StockTrader application is deployed to the production environment, you can view the web UI that's in IBM Cloud Private. First, you need to retrieve the port number.

1. 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.
2. Copy the port number.



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.

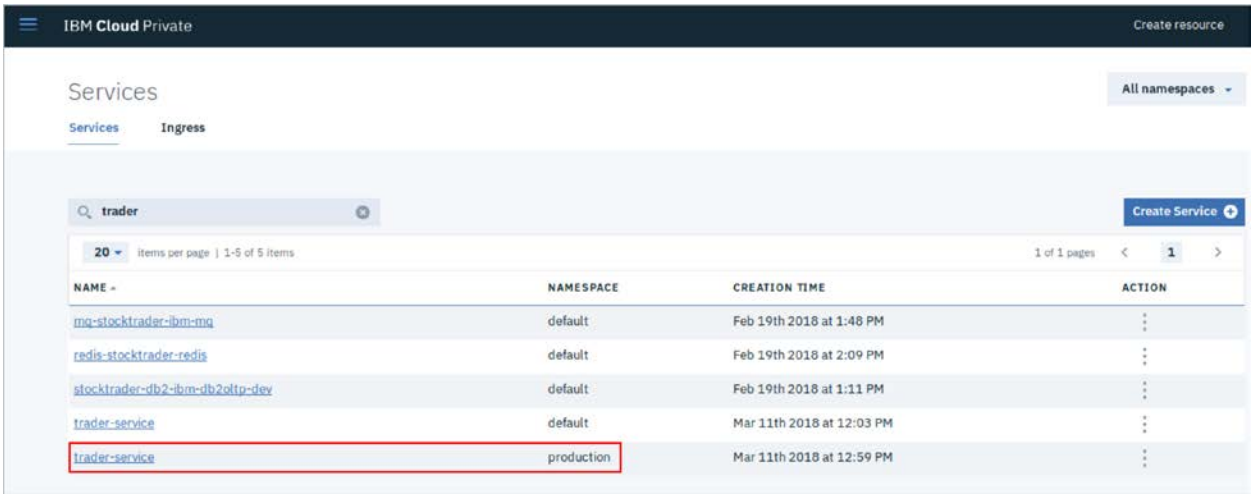


Test the application, as you did earlier, by creating a portfolio, adding stock to an account, and retrieving portfolio information.

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.