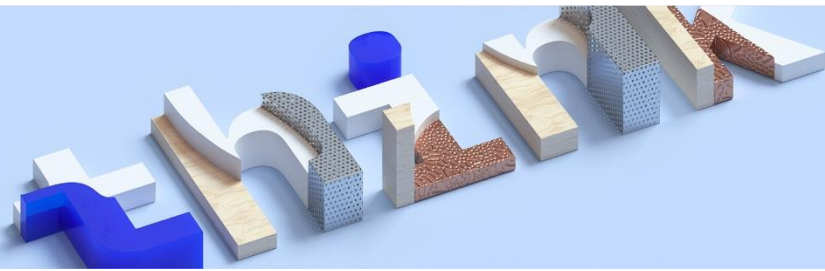


**think 2018**



## **Lab Center – Hands-on Lab**

### **Sessions 1361 and 9136**

### **Microservice Acceleration with Microclimate**

Kevin Postreich, IBM

Yi Tang, IBM

# Table of Contents

<b>MICROCLIMATE INTRODUCTION</b>	<b>3</b>
DEPLOYING TO CONTAINERS?	3
ITERATIVE DEVELOPMENT	3
<b>LAB OVERVIEW 4</b>	
LAB SCENARIO	4
LAB STEPS	5
PREREQUISITES	5
THE LOGIN CREDENTIALS FOR THE LAB VM ARE AS FOLLOWS:	5
<b>PART 0: LAB PREPARATION PREREQUISITES</b>	<b>6</b>
0.1 LAUNCHING THE LAB VM	6
0.2 CONFIGURE GITHUB ACCOUNT	6
0.3 VIEW THE VM ENVIRONMENT	15
<b>PART 1: CREATE, TEST AND DEPLOY STOCK QUOTE MICROSERVICE APPLICATION</b>	<b>18</b>
1.1 CREATING STOCK QUOTE MICROSERVICE APPLICATION	18
1.2 ADDING THE BUSINESS LOGIC TO YOUR MICROSERVICE APPLICATION	22
1.2.1 ADD THE BUSINESS LOGIC TO THE APPLICATION	23
1.2.2 REVIEW A FEW KEY ARTIFACTS GENERATED BY MICROCLIMATE	23
1.3 EXPLORE SOME OF MICROCLIMATE CAPABILITIES	25
1.3.1 VIEW THE STOCKQUOTE SOURCE CODE USING THE INTEGRATED THEIA EDITOR IN MICROCLIMATE	25
1.3.2 VIEW THE APPLICATION SERVER LOG FROM MICROCLIMATE	29
1.3.3 OPEN THE APP TO TEST THE STOCKQUOTE SERVICE ENDPOINT FROM MICROCLIMATE	29
1.3.3 RUN A LOAD TEST FOR STOCKQUOTE SERVICE	31
<b>PART 2: DEPLOY THE MICROSERVICE APPLICATION TO IBM CLOUD PRIVATE USING THE INTEGRATED JENKINS BUILD PIPELINE</b>	<b>34</b>
2.1 CREATE REPOSITORY IN GITHUB	34
2.2 ADDING THE SOURCE CODE OF YOUR MICROSERVICE PROJECT TO GITHUB	37
2.3 MICROCLIMATE PIPELINE CONFIGURATION AND DEPLOYMENT	39
2.4 TEST THE MICROSERVICE APPLICATION IN IBM CLOUD PRIVATE	47

---

## Microclimate Introduction

The past few years have seen an explosion of technologies - microservices, Docker, polyglot, ... the list goes on. However, what about your development environment? Most developers still use a single-language Integrated Development Environment (IDE), debugger, and local tools that they were using years ago. It's high time to upgrade!

**Microclimate** is a Docker-ized, end-to-end development environment. Generate microservices in Java, Node, and Swift, edit them, and see your changes on the fly. When you're happy with the code, check it in and the pipeline will automatically build and deploy to Kubernetes.

And ,we're not just another friendly IDE! We have included application monitoring tools and an HTTP load-driver to start, and plan to include other tools over time. Each of these is pre-installed and pre-configured for your service so you can just switch tabs and go. No hunting for other tools or trying to set them up when you run into problems.

### Deploying to Containers?

If you are deploying to containers, why not start your development there too? Microclimate allows you to create your microservices with automated build in Docker from day one. No more time spent trying to recreate problems that only happen on one machine, or differences when switching to containers in production.

The environment is built using Docker too - so you can run it locally, or host it on your cloud platform and use it over the web. Likewise, no more issues with local settings or installers for multiple tools.

### Iterative Development

We are developers too, so being able to iterate rapidly is very important to us. That is why we have setup every project to react to changes immediately, regardless of language, Docker, or which IDE you use.

## Lab overview

This lab exercise shows how to develop a microservice application using the **Microclimate** Framework, IBM Cloud private (ICP) and GitHub.

You will use the Microclimate Portal UI to create a new microservice project, test your microservice, view the source code using the integrated Theia editor, and run a load test using the integrated performance test tools.

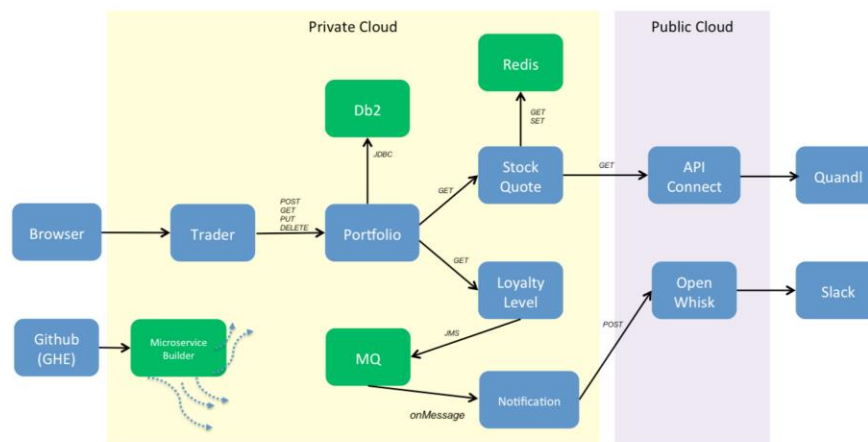
Then, once you are satisfied the microservice is working and performing as expected, you will use the integrated CICD pipeline to deploy your microservice to IBM Cloud Private (Kubernetes Cluster).

## Lab scenario

IBM Cloud Private has been installed in your company data center. Your development team has created a **StockTrader** web application that will be deployed to the Kubernetes environment.

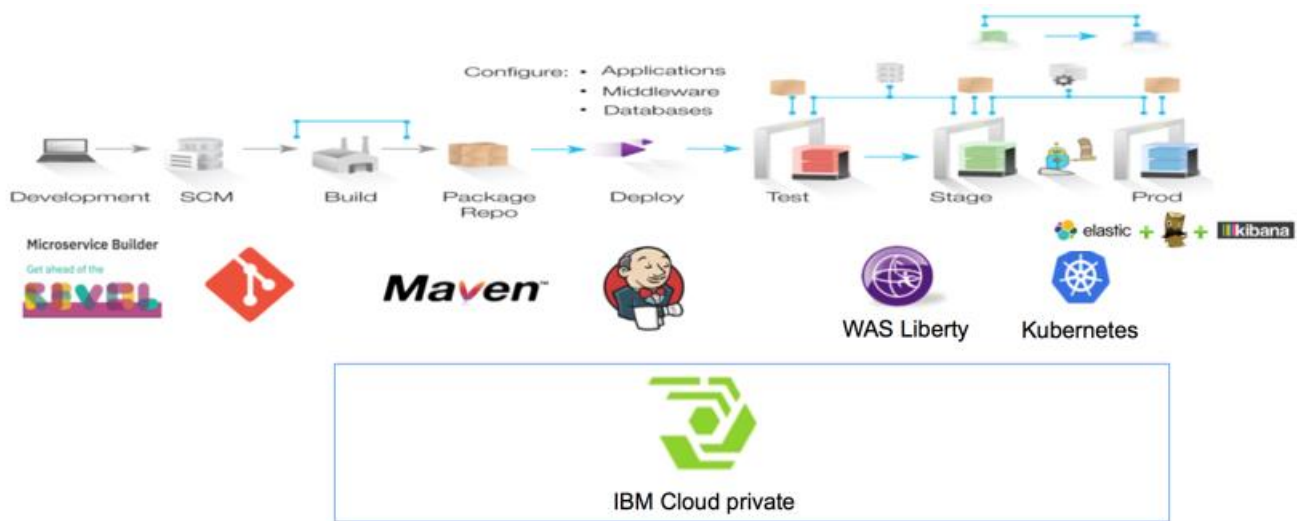
**StockTrader** is a microservice-based application utilizing various features of IBM WebSphere Liberty runtime (JAX-RS, Servlets, JPA, etc.) which stores the portfolio information in a DB2 database. Additionally, the customer's loyalty level of (Bronze, Silver, Gold, etc) is based on his/her portfolio balance. In a more advanced scenario, outside the scope of this lab, changes in the customer's loyalty level is sent to the customer via slack, using Open Whisk as shown in the application architecture diagram below.

### StockTrader Application Architecture



As you can see from the diagram, the **StockTrader** application consists five microservice applications, including **Trader**, **Portfolio**, **Stock Quote**, **Loyalty Level** and **Notification**. Your team has already created a **StockTrader** application Helm chart and deployed these microservices to IBM Cloud Private.

However, as the lead developer, you are looking into IBM tools to increase productivity of the delivery of the microservice applications through a DevOps CICD pipeline to your IBM Cloud Private, Kubernetes environment. You will create, test, and deploy the **stock quote microservice** to the Cloud platform using Microclimates inner loop testing capabilities, load test tool, and CICD pipeline.



## Lab steps

In this lab, you are going to create, test and deploy the **Stock Quote** microservice application to the Kubernetes cluster in ICP environment using IBM Microclimate. Here are the activities involved in this process:

- Configure your GitHub account
- Create your first microservice project with Microclimate
- Work with Microclimate portal to view code, run load test, view monitoring data and stats
- Check your project code into GitHub repository
- Configure the Integrated Microclimate CI/CD pipeline
- Use the CI/CD pipeline to build and deploy your microservice to Kubernetes cluster in ICP
- Validate the application works as expected in the IBM Cloud environment

## Prerequisites

The lab exercises require that you have a GitHub account. If you do not have your GitHub account, you can go to GitHub web site (<http://github.com>) to create one free of charge.

This lab also requires you have internet access.

**The login credentials for the lab VM are as follows:**

User ID: **ibmdemo**

Password: **passw0rd**

---

## Part 0: Lab Preparation Prerequisites

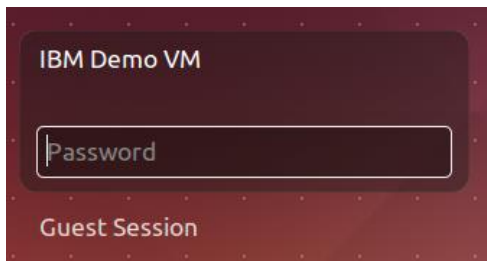
### 0.1 Launching the lab VM

- \_\_1. Start the lab VM.
- \_\_2. Login to the lab VM with the login credentials:

User ID: **ibmdemo**


Password: **passw0rd**

**Note:** the password is the same Ubuntu Terminal **sudo** password you are going to use in the Lab.



### 0.2 Configure GitHub account

In order to securely integrate Microclimate with GitHub, you will need to do a few configurations in your GitHub account. If you do not have a GitHub account, you can create one as shown below.

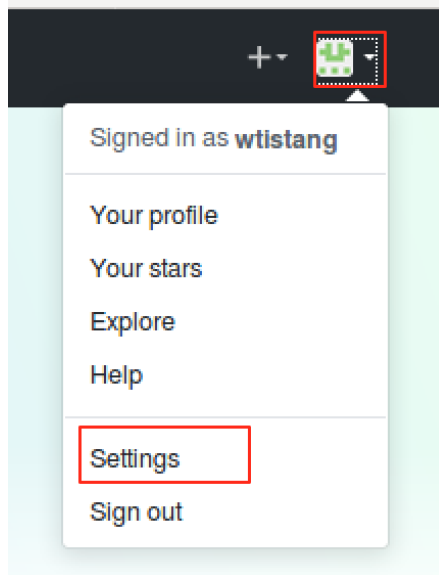
- \_\_1. From the **Firefox** web browser window, go to GitHub URL as described below
  - \_\_a. Open a new browser tab
  - \_\_b. Click **GitHub** book mark  to launch the GitHub.
- \_\_2. If you already have a GitHub account, you can **sign in**. Otherwise, you can **sign up a free new account**.

\_\_\_3. After signed in, you need to make several configurations changes to your GitHub account.

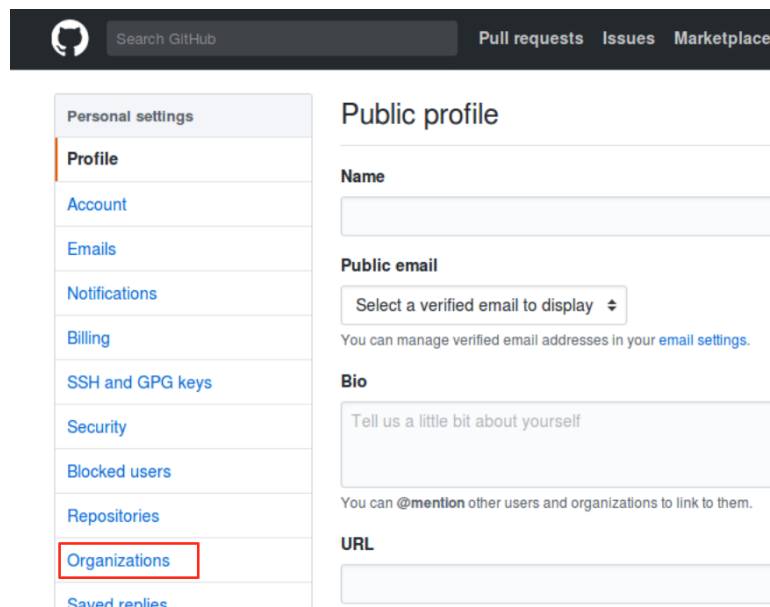
\_\_\_a. Create an organization.

**Note:** Microclimate requires a GitHub organization. If you already have an existing organization, you can use it and skip this step. Typically, the default organization for your account is the same name as your GitHub Username. You can use this Organization, if you like. Or create a new Organization in your account as follows:

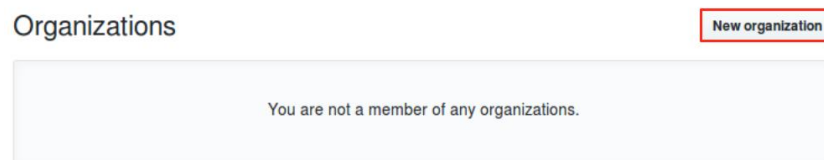
- i. To create a new Org, Click your **account icon** on the top right corner, and select **Settings** from the down menu.



- ii. In the Settings page, select **Organizations**.




- iii. Click **New organization**.





- iv. Enter your Github **Organization name**, **Billing email** (valid email format required), choose **Free plan** and click **Create organization**.



# Sign up your team

 Step 1:  
Set up the organization

 Step 2:  
Invite members

 Step 3:  
Organize your team

## Create an organization account

Organization name

wtistangdev ✓

The organization account will live at <https://github.com/wtistangdev>

Billing email

wtistangdev@gmail.com

Receipts will be sent here

## Choose your plan

<input checked="" type="radio"/>	<b>Free</b> Unlimited users and public repositories	\$0
<input type="radio"/>	<b>Team</b> Starts at \$25 / month which includes your first 5 users. Unlimited public repositories Unlimited private repositories	\$9 per user / month
<input type="radio"/>	<b>Business</b> Includes everything in the Team plan, plus: SAML based single sign-on (SSO) 99.95% Guaranteed uptime SLA 24/5 email support with < 8 hour response time <a href="#">Need help getting started or on-premises hosting? Contact us.</a>	\$21 per user / month

☐ This account is owned by a business (see the [Corporate Terms of Service](#) for details)

By clicking on "Create organization" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).




Create organization

Organization account is used to plan, build, release software — all while discussing ideas

The credit card and billing information is linked to the organization account.

v. Click **Submit**.

## Organization details

 Step 1: Set up the organization	 Step 2: Invite members	 Step 3: Organization details
--	---	---

We're collecting data to help us understand how people use GitHub organizations. We'd appreciate it if you'd share a little about your plans for this organization with us. This step is optional and the data will only be used for research purposes. You can [skip this step](#) to opt out.

### What do you plan to use your organization for?

- ☐ Professional work, for-profit
- ☐ Professional work, non-profit (not including educational programs)
- ☐ Educational purposes (includes academic research)
- ☐ An open source project
- ☐ Hobby projects
- ☐ A hackathon
- ☐ I'm not sure yet
- ☐ Other (please describe)

### How long do you plan to use this organization?

- ☐ Just a few days
- ☐ A few weeks to months
- ☐ A year or more
- ☐ Indefinitely
- ☐ I'm not sure yet
- ☐ Other (please describe)

### About how many people do you expect to work with in this organization?

- ☐ I plan to work alone
- ☐ 5 or fewer
- ☐ 6 to 20
- ☐ 21 to 50
- ☐ More than 50
- ☐ I'm not sure yet
- ☐ Other (please describe)

[skip this step](#)

Your organization will be created.

## Organizations


 **wtistangdev** owner

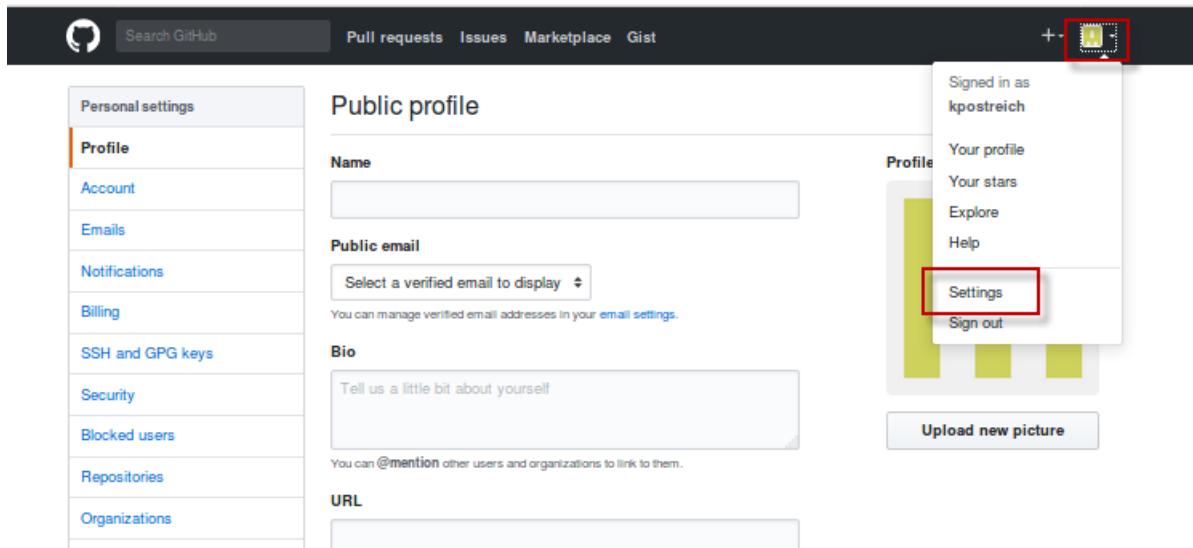
\_\_b. Record your **GitHub Org** in the Cheatsheet

**GitHub Organization Name:** \_\_\_\_\_

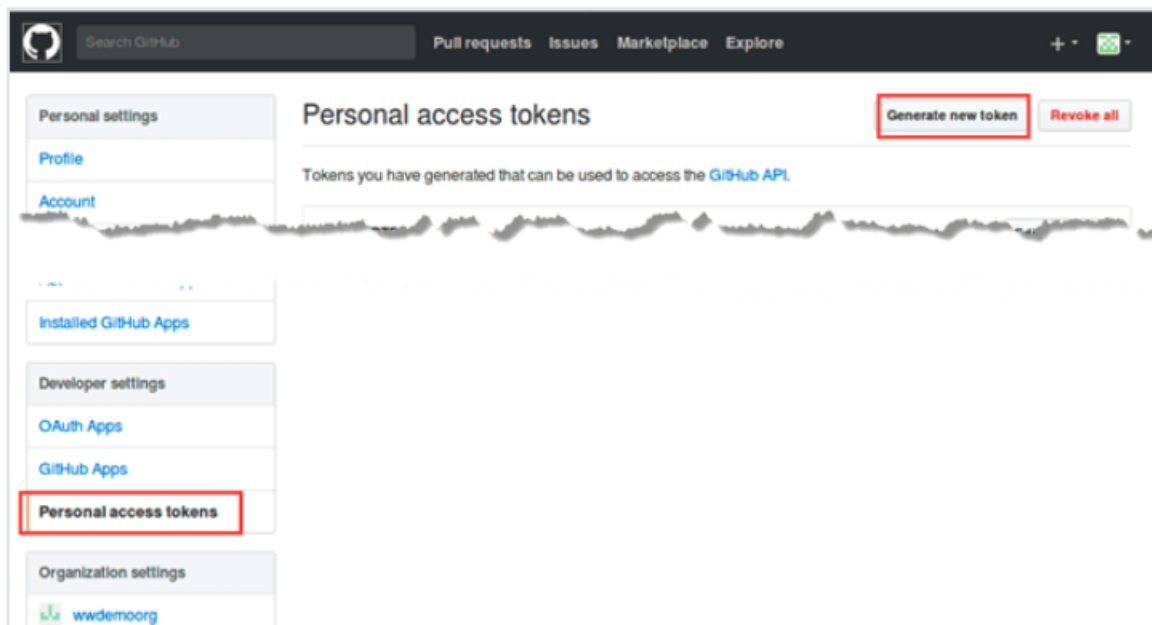
\_\_4. Creating personal access token

GitHub allows for automated access using a personal access token. You will create this personal access token and record the values in the cheat sheet for a later step when you create a Jenkins job that will access the code.

- \_\_a. Click the User icon  in the upper right corner of the GitHub page. Then click on **Settings** from the menu



- \_\_b. From the GitHub **Settings** page, go to **Developer settings > Personal access tokens**. Then click **Generate new token**



If you are asked to log in, enter your GitHub password.

- \_\_c. Add a **token description** to the field; it can be any name you like.
- \_\_d. Select the **scopes** shown below, by clicking the check boxes next to them.

- repo:status
- public\_repo
- admin:repo\_hook
- admin:org\_hook

\_\_e. Click the **Generate token** button

### New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Token description**

Access token for lab.

What's this token for?

**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> admin:org	Full control of orgs and teams
<input type="checkbox"/> write:org	Read and write org and team membership
<input type="checkbox"/> read:org	Read org and team membership
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> user:pgp_key	Write user pgp keys
<input type="checkbox"/> read:pgp_key	Read user pgp keys

**Generate token** [Cancel](#)

The token is now generated.

## Personal access tokens

[Generate new token](#)[Revoke all](#)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 18397c18fe558614f76b79cbb295336758ea39c4

[Edit](#)[Delete](#)

ⓘ Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

- \_\_f. Copy and paste the **personal access token** to the **Cheatsheet** and save it, you need to use it later.

**GitHub Personal Access OATH Token:** \_\_\_\_\_

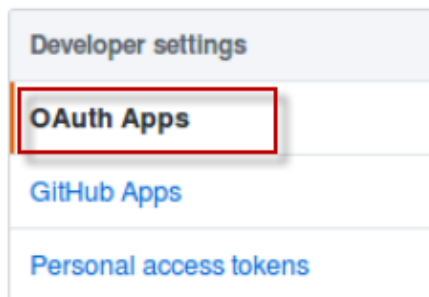


### Please Read!

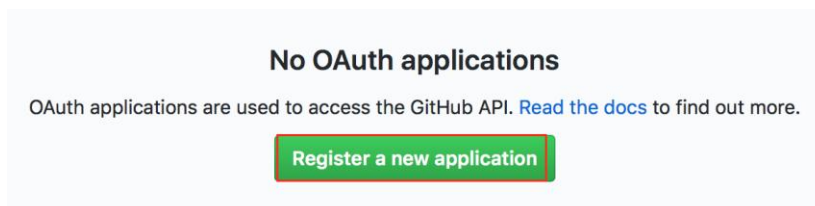
In GitHub, you can only access this Personal access token once right after you created it. If you cannot remember the Personal access Token, you have to regenerate it.

- \_\_5. Next, add the **GitHub OAuth** integration so that you can log in to Jenkins using GitHub for authentication.

- \_\_a. In GitHub page, go to **Settings > Developer settings > OAuth Apps**



- \_\_b. Click **Register a new application**.

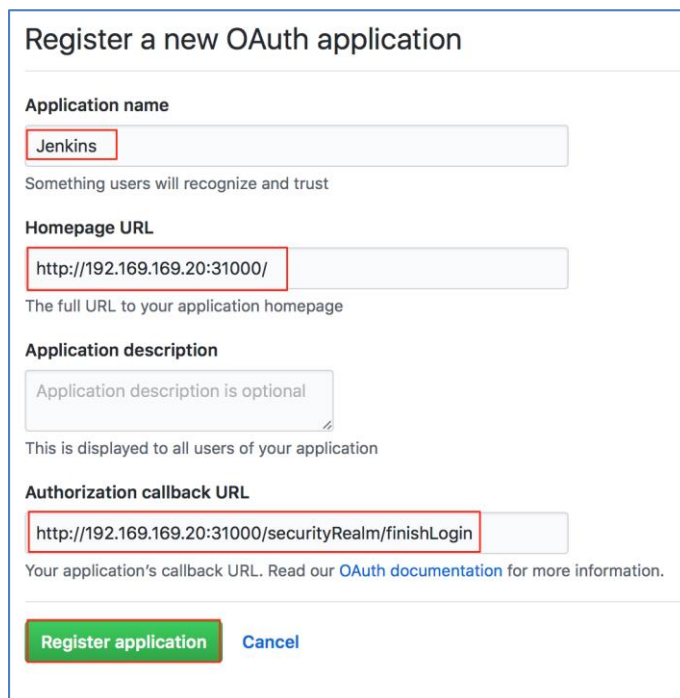


\_\_c. Enter your **application name** as: Jenkins

\_\_d. Enter the **homepage url** as:  
`http://192.169.169.20:31000/`

\_\_e. Enter **authorization callback url** as:  
`http://192.169.169.20:31000/securityRealm/finishLogin`

\_\_f. Click the **Register application** button



**Register a new OAuth application**

**Application name**  
Jenkins  
Something users will recognize and trust

**Homepage URL**  
http://192.169.169.20:31000/  
The full URL to your application homepage

**Application description**  
Application description is optional  
This is displayed to all users of your application

**Authorization callback URL**  
http://192.169.169.20:31000/securityRealm/finishLogin  
Your application's callback URL. Read our [OAuth documentation](#) for more information.

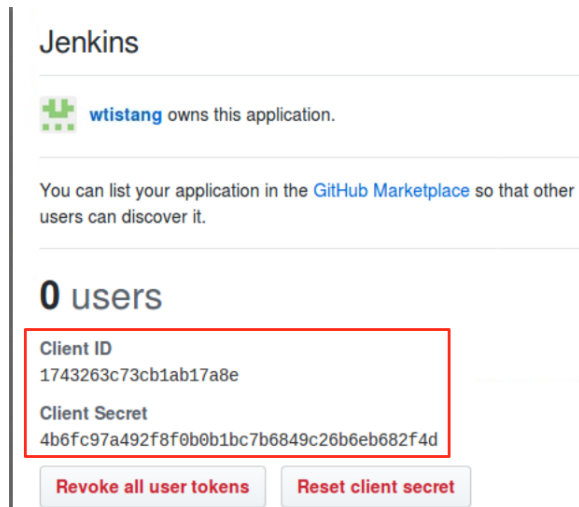
**Register application** [Cancel](#)

\_\_g. After registration, your OAuth application has the following two hex strings:

**TIP:** Scroll to the top of the page to view the **Client ID** and **Client Secret** information.

- **Client ID**
- **Client Secret**

Record these values in the **Cheatsheet** as shown below.




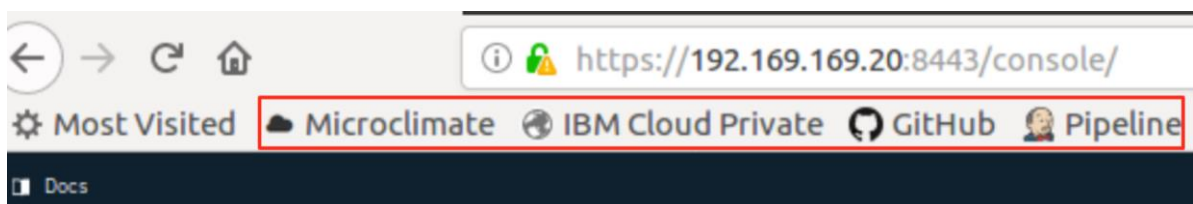
GitHub App (Client) ID: \_\_\_\_\_

GitHub Client App Secret: \_\_\_\_\_

=== You are done when GitHub configuration ===

### 0.3 View the VM environment

- \_\_\_1. Open a **Firefox** web browser  and you can see four web book marks in the Bookmarks Toolbar.
- IBM Microclimate
  - IBM Cloud private – ICP
  - GitHub – GitHub application
  - Pipeline – IBM MSB pipeline application



These are the tools used to build, deploy and test your microservice application during this lab.



### Please Read!

Do not click the pipeline bookmark at this time!

- \_\_2. Open a new Browser tab, and click the **IBM Cloud private** bookmark to launch the IBM Cloud private community edition.
- \_\_3. If you see the message “Your connection is not secure”, do the following:
  - \_\_a. Click **Advanced**
  - \_\_b. Click **Add Exception...**
  - \_\_c. Click **Confirm Security Exception**
- \_\_4. Accept the default User ID/Password as: **admin / admin** and click **Login**.

IBM Cloud Private

Fast. Flexible. Intelligent.  
Open. Enterprise-grade.

---

Log in to your account.

Username

admin

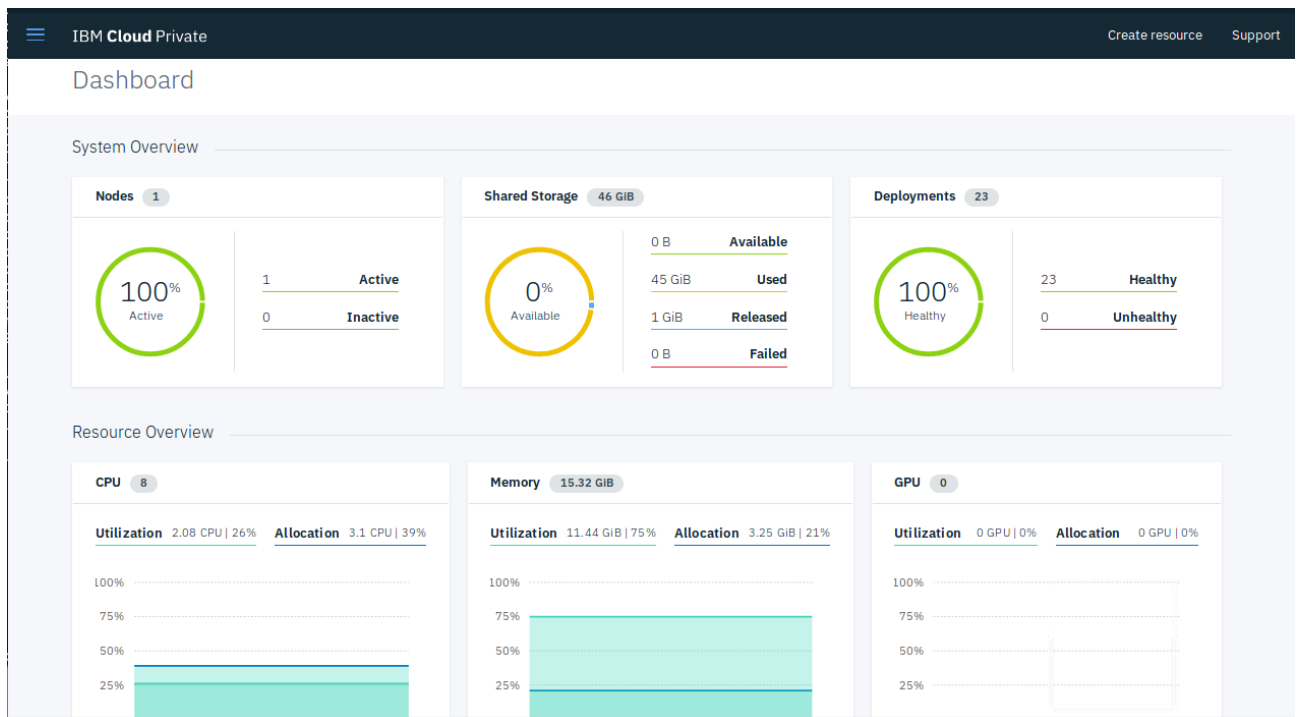
Password

\*\*\*\*\*

Log in

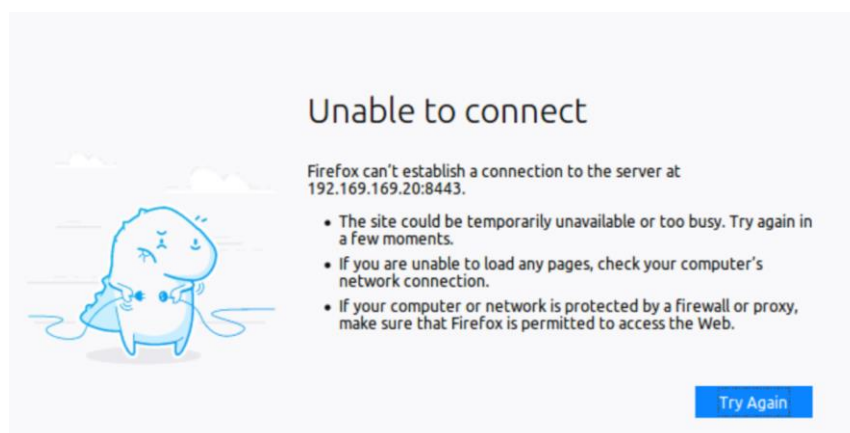
The ICP dashboard is shown as followings. In this lab, you will use the Microclimate integrated pipeline to deploy the stockquote microservice to this IBM Cloud Private environment.





### Please Read!

Note: If you see the following error message, it means the ICP is still in the process to start. Just wait for a few minutes and refresh the page and you will get the login page.



---

## Part 1: Create, test and deploy Stock Quote microservice application



**PART 1** is required to be completed before moving to the next parts of the lab.

This part focuses on using the Microclimate Portal to create, test, and work with a microservice based application. In this development phase, your microservice will be built and deployed to a Docker container running in your local environment. Modifications to the application code are immediately injected into the running container for a rapid inner loop development experience.

### 1.1 Creating Stock Quote microservice application

In this section, you are going to create a microservice application called “**stockquote**” using the Microclimate portal. Then, add the business logic to it by leveraging the application source code that is provided for you. Microclimate creates and pre-configures the necessary artifacts for building, testing, and deploying your service to Kubernetes via integrated CICD pipeline.

#### 1.1.1 Start the Microclimate portal

At the time of this writing, Microclimate is in Beta. Some of the steps to work with Microclimate Beta are different from the anticipated experience when the tools become generally available. Your feedback is important, and very much appreciated.

##### \_\_1. Start Microclimate

- \_\_a. Open a new Terminal window, and navigate to the microclimate-master directory, using the following command:

```
cd microclimate-master
```

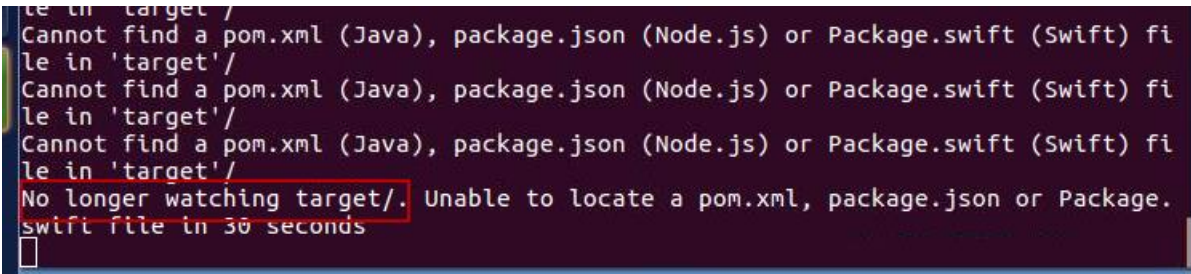


- \_\_b. Run the following command to start Microclimate

```
./run.sh
```

The script will take a few moments to complete as it sets up the microclimate environment for you.

**Note:** In the Beta, when you see the message, **No longer watching target**, Microclimate is up and running.



```
le in 'target'/
Cannot find a pom.xml (Java), package.json (Node.js) or Package.swift (Swift) fi
le in 'target'/
Cannot find a pom.xml (Java), package.json (Node.js) or Package.swift (Swift) fi
le in 'target'/
Cannot find a pom.xml (Java), package.json (Node.js) or Package.swift (Swift) fi
le in 'target'/
No longer watching target/. Unable to locate a pom.xml, package.json or Package.
swift file in 30 seconds
```

\_\_c. From the Terminal window, enter **Ctrl-C** to end the run.sh script.

\_\_2. Launch Microclimate from the Firefox browser

\_\_a. Open a new tab in the **Firefox** web browser

\_\_b. Click the **Microclimate** bookmark  in the Bookmark Toolbar

The Microclimate Getting Started page is displayed. Next, you will create a new Java based Microservice project.

### 1.1.2 Working with your first microservice project

\_\_1. Create your first microservice

\_\_a. From the Microclimate portal, click the **New Project** button

\_\_b. From the **New Project** page, select **.Java** language

\_\_c. Name your project **stockquote**

**Note:** For the purposes of this lab, ensure you name the service “**stockquote**”. Scripts we configured for the lab depend on this name.

\_\_d. Click the **Next** button to continue

Microclimate Projects

## New project

Select your language

.java ✓

.js

.swift

Name your project

stockquote

Cancel Next

\_\_e. From the **Select your Framework** page, select **MicroProfile / Java EE**

\_\_f. Click the **Create** button

Notice that the context root for your microservice is set to “**stockquote**”. This is the context root for accessing the service at runtime.

## Project stockquote

Select your framework

MicroProfile / Java EE ✓

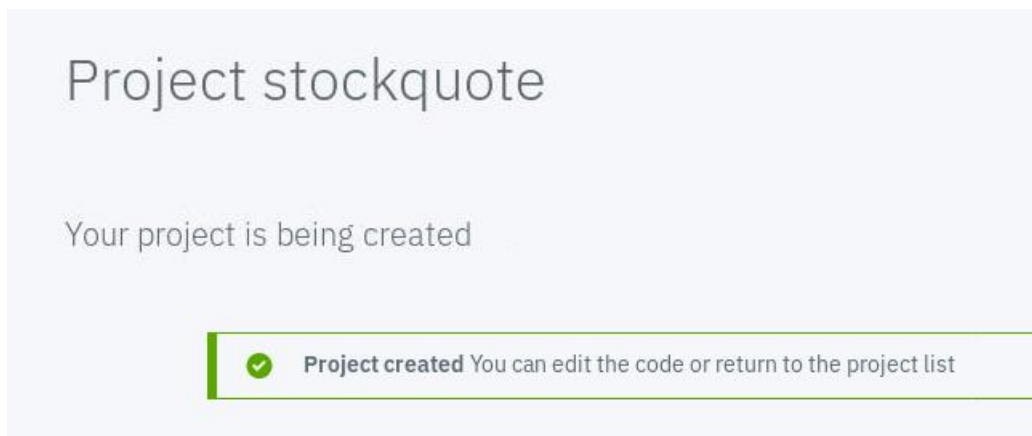
Spring

Context root

stockquote

Cancel Previous Create

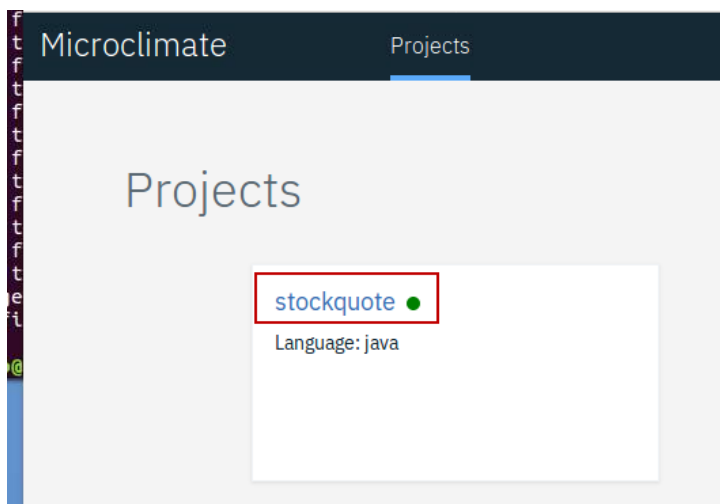
\_\_g. Your **stockquote** project is created, click the **Close** button



At this time, Microclimate is generating the project artifacts, performing a Maven build, creating a Docker image for your application, and deploying the microservice to a Docker container in the local virtual machine.

After a few moment, when the application is running, your **stockquote** project status indicator will turn from red to **green**.

**Note:** You can continue the lab while Microclimate is building and deploying your application.



Your **stockquote** project is now created and it is stored at:

```
/home/ibmdemo/microclimate-master/microclimate-workspace/stockquote
```

## \_\_2. Change the file permissions for your stockquote project

In this lab, you created a project that contains artifacts for a Java EE / Microprofile microservice. However, we have provided the application business logic that you will add to the project, via a shell script. The shell script needs to have write permission to this project directory.

### \_\_a. Open a Terminal window

- \_\_b. Navigate to the microclimate workspace directory, using the following command:

```
cd /home/ibmdemo/microclimate-master/microclimate-workspace
```

- \_\_c. Change the file permissions for stockquote using the following command:

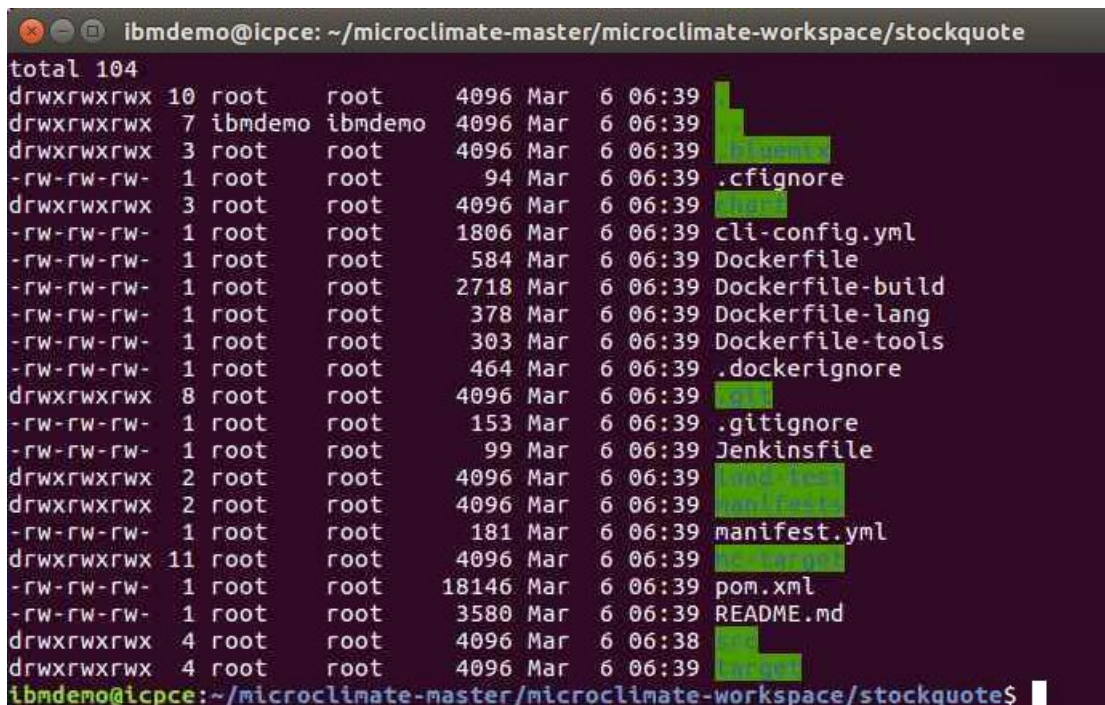
```
sudo chmod a+w -R stockquote/
```

when prompted for a password, type `passw0rd`

- \_\_d. Change to the **stockquote** directory and list the contents of the folder, using the following commands:

```
cd stockquote
```

```
ls -al
```



```
ibmdemo@icpce: ~/microclimate-master/microclimate-workspace/stockquote
total 104
drwxrwxrwx 10 root    root    4096 Mar  6 06:39 .
drwxrwxrwx  7 ibmdemo ibmdemo 4096 Mar  6 06:39 ..
drwxrwxrwx  3 root    root    4096 Mar  6 06:39 ibmdemo
-rw-rw-rw-  1 root    root     94 Mar  6 06:39 .cfignore
drwxrwxrwx  3 root    root    4096 Mar  6 06:39 .git
-rw-rw-rw-  1 root    root   1806 Mar  6 06:39 cli-config.yml
-rw-rw-rw-  1 root    root    584 Mar  6 06:39 Dockerfile
-rw-rw-rw-  1 root    root   2718 Mar  6 06:39 Dockerfile-build
-rw-rw-rw-  1 root    root    378 Mar  6 06:39 Dockerfile-lang
-rw-rw-rw-  1 root    root    303 Mar  6 06:39 Dockerfile-tools
-rw-rw-rw-  1 root    root    464 Mar  6 06:39 .dockerignore
drwxrwxrwx  8 root    root    4096 Mar  6 06:39 .idea
-rw-rw-rw-  1 root    root    153 Mar  6 06:39 .gitignore
-rw-rw-rw-  1 root    root     99 Mar  6 06:39 Jenkinsfile
drwxrwxrwx  2 root    root    4096 Mar  6 06:39 .mvn
drwxrwxrwx  2 root    root    4096 Mar  6 06:39 .mvn
-rw-rw-rw-  1 root    root    181 Mar  6 06:39 manifest.yml
drwxrwxrwx 11 root    root    4096 Mar  6 06:39 .mavenrc
-rw-rw-rw-  1 root    root   18146 Mar  6 06:39 pom.xml
-rw-rw-rw-  1 root    root   3580 Mar  6 06:39 README.md
drwxrwxrwx  4 root    root    4096 Mar  6 06:38 .src
drwxrwxrwx  4 root    root    4096 Mar  6 06:39 .src
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$
```

## 1.2 Adding the business logic to your microservice application

The **stockquote** microservice you created does not have the business logic for the stock quote service. In this lab, we have provided that code for you.. In order to get the business logic code into the microservice, you will run a shell script to add the business logic. In reality, you would write your own code, using your favorite IDE development tools via integration with Microclimate.

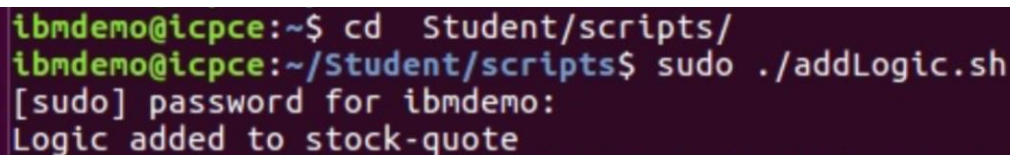
**Note:** In the Beta, an update to the pom.xml file does not trigger a full rebuild. Therefore, the shell script will also restart Microclimate, which will trigger a full rebuild of the project. Rest assured, when Microclimate is generally available, restating Microclimate will no longer be required.

### 1.2.1 Add the business logic to the application

- \_\_\_1. In a terminal window, change to the directory where the shell script is located. Then run the shell script to copy the business logic into the microservice project:

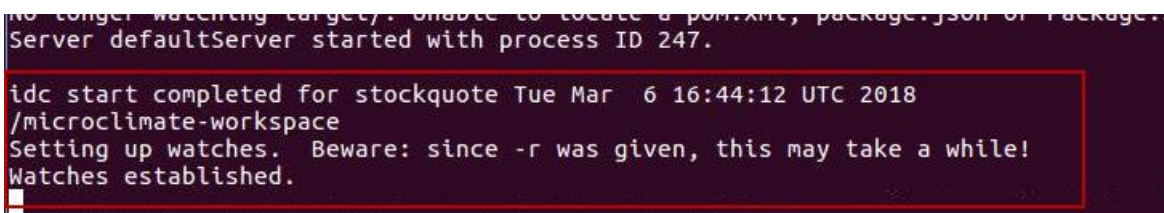
```
cd /home/ibmdemo/Student/scripts/  
sudo ./addLogic.sh
```

If prompted for a password, type: `passw0rd`



```
ibmdemo@icpce:~$ cd Student/scripts/  
ibmdemo@icpce:~/Student/scripts$ sudo ./addLogic.sh  
[sudo] password for ibmdemo:  
Logic added to stock-quote
```

**Note:** it will take a few moments for Microclimate to restart. You will know that Microclimate is ready when you see the following messages in the Terminal window.



```
no longer watching target: unable to locate a pom.xml, package.json or Package-  
Server defaultServer started with process ID 247.  
idc start completed for stockquote Tue Mar 6 16:44:12 UTC 2018  
/microclimate-workspace  
Setting up watches. Beware: since -r was given, this may take a while!  
Watches established.  
█  
█
```

- \_\_\_2. When Microclimate is ready, enter **CTRL-C** in the Terminal window, to terminate the script

Microclimate is now ready. Your new stockquote application has been rebuilt and is running in the local Docker environment.

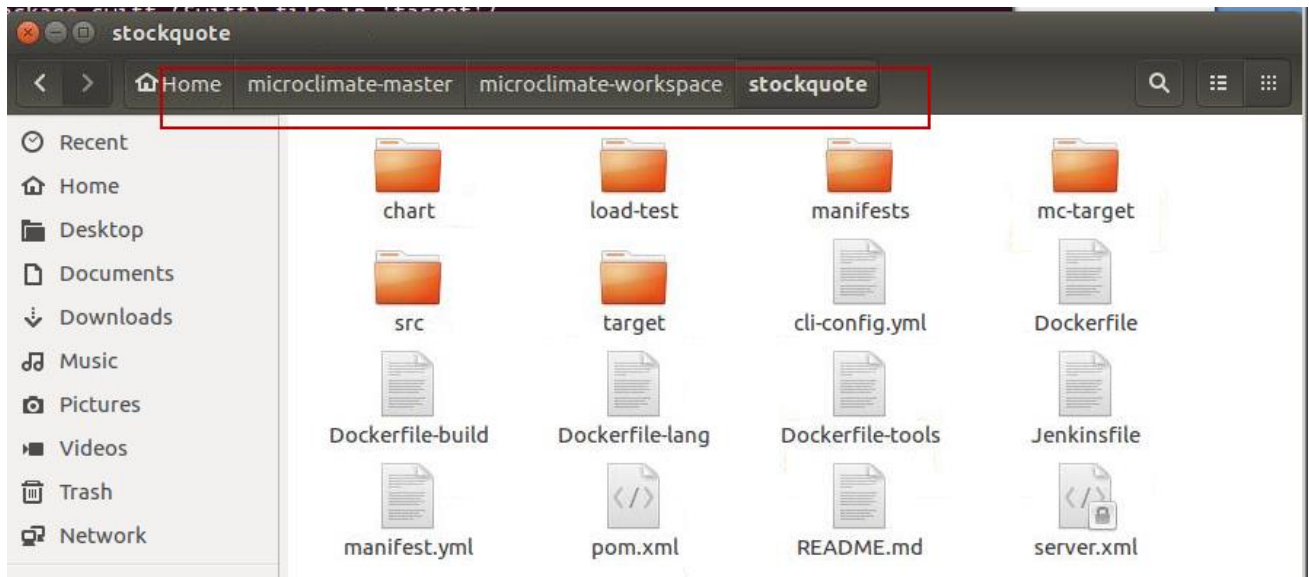
### 1.2.2 Review a few key artifacts generated by Microclimate



- \_\_\_1. Open the File Viewer from the Ubuntu desktop
- \_\_\_2. Navigate to **Home > Microclimate-master > microclimate-workspace > stockquote**



This is the top-level directory for your microservice. You see a combination of artifacts generated by Microclimate, and updated artifacts we copied in specifically for our stockquote application.



\_\_3. View the **stockquote** project to see the project structure.



Feel free to view and explore the project artifacts. When you are ready, continue with the lab.

Below, we highlight key artifacts created by Microclimate, or updated by the addLogic script.

The project includes the standard Maven structure for application build and tests.

The microservice project files generated by Microclimate include maven **pom.xml** file, **Dockerfile** for Docker container, **Jenkinsfile** for Jenkins pipeline job, and Kubernetes Helm Chart deployment files which is located in the **chart** folder.

Microclimate also generated a Jmeter script used for performing a basic load test of your microservice in your development environment.

Recall that you ran the shell script to add the business logic and update artifacts to properly build and test the stockquote microservice. For your reference, the files that we created or updated are listed below:

- **server.xml** – The Liberty server configuration for stockquote service



- **TestPlan.jmx**, located under the **load-test** folder – Updated to invoke the stockquote service, 20 iterations with 1 thread
- **src folder** – Business logic in stockquote.java
- **pom.xml** – Updated to build the stockquote service
- **Dockerfile** – Used by microclimate to build the Docker image for the microservice
- **Jenkinsfile** – Used with the Microclimate integrated CICD pipeline for deployment to IBM Cloud Private
- **Chart folder** – Updated Kubernetes deployment artifacts to deploy and configure stockquote to IBM Cloud Private. For example, pulling deployment variables from pre-configured Kubernetes secrets in ICP. Yaml configuration files for configuring Kubernetes.

## 1.3 Explore some of Microclimate capabilities

As you learned in the introduction portion of this lab, Microclimate provides some powerful features that help developers be more productive when developing microservice applications deployed to Docker containers. Microclimate delivers a developer friendly portal that provides intuitive capabilities for developing, testing, and deploying microservices in containerized environments.

In this section of the lab, you will get familiar with only a few of the capabilities, including:

- Using the integrated Theia editor
- Viewing the Application Server log from a running Docker container
- Run the application (Liberty runtime in Docker container)
- Running a load test against the application
- Viewing basic performance monitoring data and statistics for your microservice endpoint

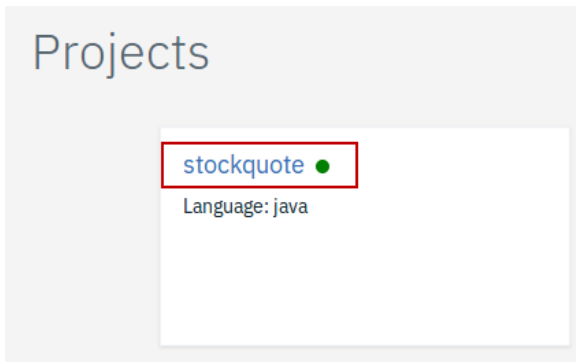
### 1.3.1 View the stockquote source code using the integrated Theia editor in Microclimate

\_\_1. Launch Microclimate from the Firefox browser

\_\_a. Open a new browser tab in Firefox

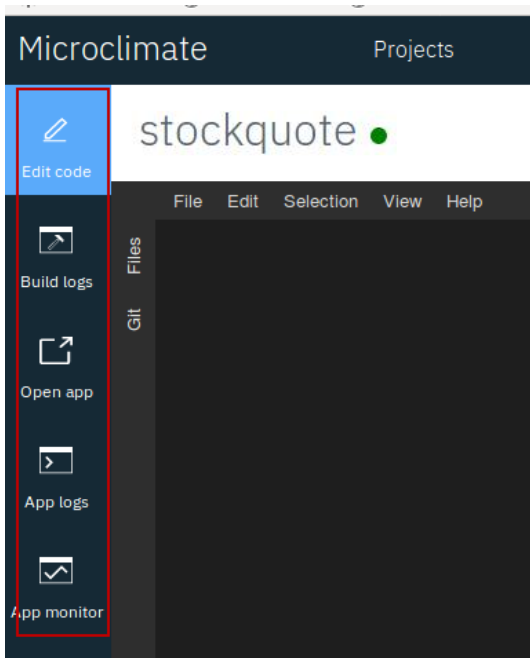
\_\_b. Click the **Microclimate** bookmark  in the Firefox Bookmark Toolbar

Your **stockquote** project is listed in the **projects** view. The project should be in the running state, indicated by the GREEN dot.

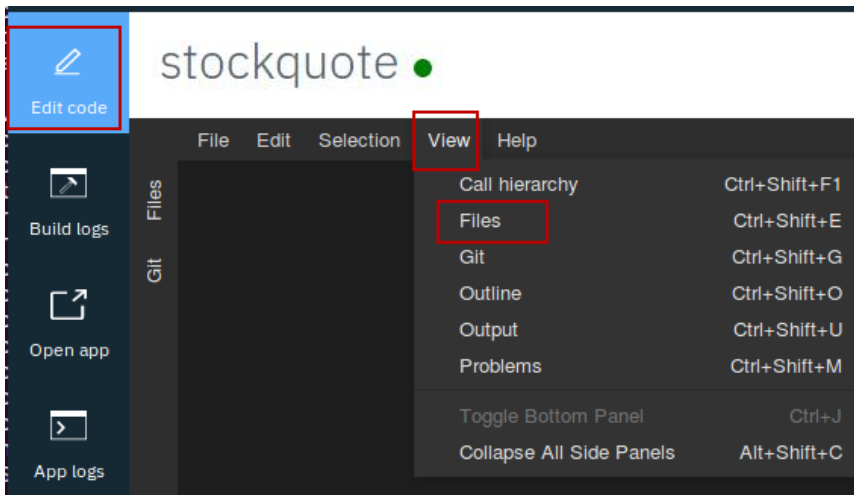


- \_\_2. Click on the **stockquote** project to open it in the Microclimate portal

Notice that the portal provides easy to access actions for editing the code, viewing build logs, running the application, viewing application server logs, and monitoring the application.

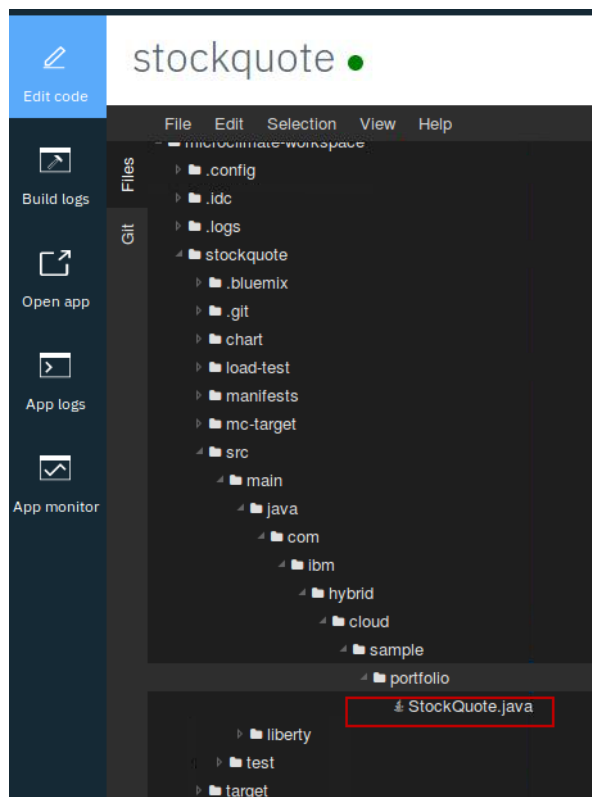


- \_\_3. Let's view the stockquote java source using the integrated Theia editor
- \_\_a. Click on the **Edit Code** action from the left navigation bar in the portal
  - \_\_b. From the main pane, select **view > Files** from the menu



- \_\_\_c. The Microclimate projects are listed in the Files explorer.
- \_\_\_d. Drill down into the **stockquote** project, locating the directory where the stockquote.java file is located.

```
Stockquote > src > main > java > com > ibm > hybrid > cloud >
sample > portfolio
```

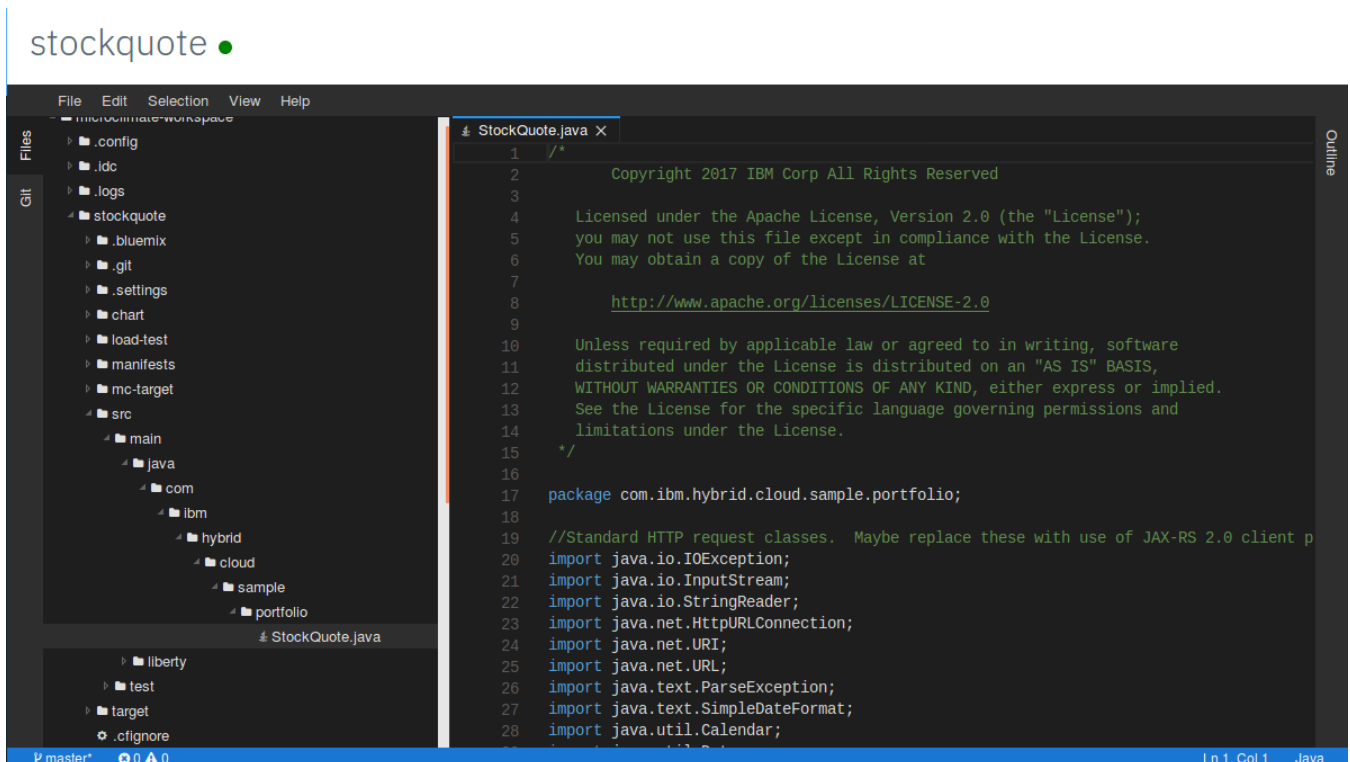


- \_\_\_e. Double Click on **StockQuote.java** to open it in the integrated Theia editor.

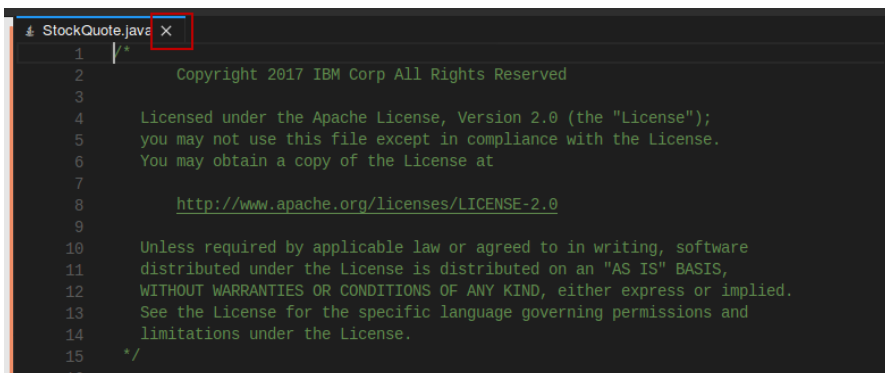
The source code is displayed in the editor pane.

While you do not need to modify the source code during the lab, it is important to note that changes made to the source code triggers an incremental build and injects the updated code into the running Docker container.

This is a nice feature in cases when you only need to make minor updates to your code. Additional features in Microclimate will soon become available that will allow developers to disable this function for cases where significant code changes are occurring.



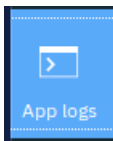
\_\_f. Click the **X** next to **StockQuote.java** to close the file from edit mode.



### 1.3.2 View the Application Server log from Microclimate

Using the **View App log** action in Microclimate, you can quickly and easily see if the Liberty runtime is up, or if there are errors and application exceptions on the server.

- \_\_1. View the Application log for the stockquote service.
  - \_\_a. From the Microclimate action pane, click the **App logs** action



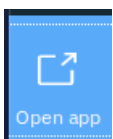
- \_\_b. From this simple log view, you can determine that the Liberty server is running and the stockquote application is deployed.

```
I SESN0176I: A new session context will be created for application key default_host/jwt
I SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID generation.
I SESN0176I: A new session context will be created for application key default_host/health
I SESN0176I: A new session context will be created for application key default_host/metrics
I SESN0176I: A new session context will be created for application key default_host/ibm/api
I SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID generation.
I SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID generation.
I SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID generation.
I DYNAL056I: Dynamic Cache (object cache) initialized successfully.
I WELD-000900: 2.4.3 (Final)
I SRVE0242I: [com.ibm.ws.microprofile.metrics] [/metrics] [MetricsRESTProxyServlet]: Initialization successful.
I SRVE0169I: Loading Web Module: javametrics.web.
I SRVE0250I: Web Module javametrics.web has been bound to default host.
A CWWKT0016I: Web application available (default host): http://ba738933350f:9080/javametrics-dash/
A CWWKZ0001I: Application javametrics-dash started in 1.346 seconds.
I SRVE0169I: Loading Web Module: stockquote-1.0-SNAPSHOT.
I SRVE0250I: Web Module stockquote-1.0-SNAPSHOT has been bound to default host.
A CWWKT0016I: Web application available (default host): http://ba738933350f:9080/stockquote/
A CWWKZ0001I: Application stockquote started in 1.371 seconds.
I SESN0176I: A new session context will be created for application key default_host/stockquote
I SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID generation.
I SESN0176I: A new session context will be created for application key default_host/javametrics-dash
I SESN0172I: The session manager is using the Java default SecureRandom implementation for session ID generation.
A CWWKF0012I: The server installed the following features: [microProfile-1.2, jsp-2.3, managedBeans-1.0, mpFaultT
I CWWKF0008I: Feature update completed in 10.630 seconds.
A CWWKF0011I: The server defaultServer is ready to run a smarter planet.
I CWWKH0046I: Adding a websocket ServerEndpoint with the following URI: /javametrics-socket
I SRVE9103I: A configuration file for a web server plugin was automatically generated for this server at /root/ap
```

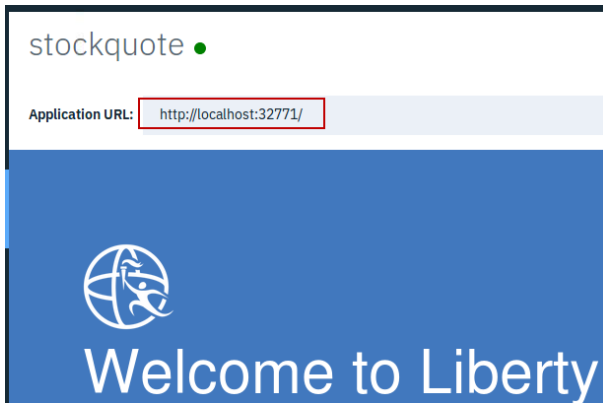
### 1.3.3 Open the app to test the stockquote service endpoint from Microclimate

Microclimate provides a convenient way to validate your service endpoint is available, and that your service responds with an appropriate response.

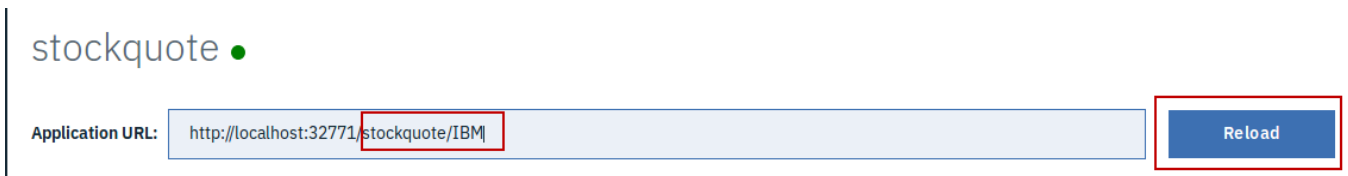
- \_\_1. Test the stockquote service endpoint
  - \_\_a. Click on the **Open App** action from the Microclimate action pane



- \_\_b. In the main view, the Application URL references the localhost and port where the stockquote service is running, in Docker. The Liberty welcome page is shown, indicating that the Application URL is available.



- \_\_c. Update the Application URL to point to your stockquote service by appending the following text:
- I. Append `stockquote/IBM` to the end of the Application URL
  - II. Then click the **Reload** button



- \_\_d. The normal response from the service is json string showing the

```
{"symbol": "IBM", "date": "2018-03-06", "price": 156.12}
```

The stockquote REST service endpoint requires a stock symbol as an attribute to the service. The attribute can be any valid NASDAQ stock symbol. In the example above, I provided IBM, which the service will return the stock price for IBM stock.

**Note:** the service uses a freely available API from Quandl to lookup stock price.



The Quandl API allows up to 50 API calls per day from anonymous users. Our stock quote service will detect if we are rate limited by Quandl. And, if so, we will simply return a DUMMY stock quote value. When rate limited by Quandl, it returns 429 HTTP Response code and can be seen in the Application log, indicating we have exceeded the maximum number of FREE API calls to the service.

```
R java.io.IOException: Server returned HTTP response code: 429 for URL: https://www.quandl.com/
```

### 1.3.3 Run a load test for stockquote service

Microclimate has built-in the capability for developers to quickly and easily run a small load test against their microservice. The Jmeter tooling and script are pre-deployed and pre-configured. Of course, as a developer, you can modify the provided TestPlan.jmx script or provide your own to perform your own custom performance tests.

In this section, you will run the load test that we modified for the project. The modified test plan simply invokes the stockquote/{Symbol} endpoint. We have also reduced the number of iterations and threads for simplicity, attempting to keep from being rate limited by Quandl. The test plan will run 20 iterations with 1 user. We wanted to keep within the 50 free Quandl API calls per day.

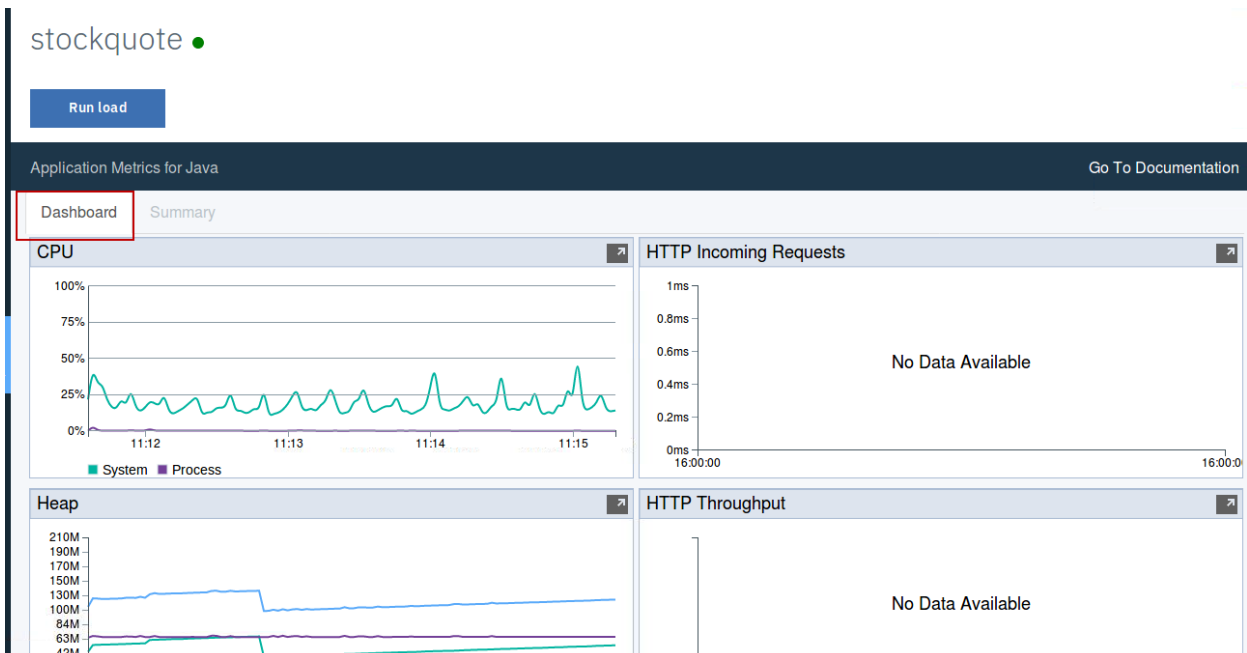
\_\_1. Launch the App monitor action in Microclimate

\_\_a. Click on the **App Monitor** action from the Microclimate action pane



The Application Metrics for Java Dashboard view will be displayed. It may take a couple of moments for the CPU and Heap charts to display the data.

**Note:** The HTTP Incoming Requests and HTTP Throughput will NOT contain data until we run the load test.



\_\_2. Click the **Run load** button to start the small load test

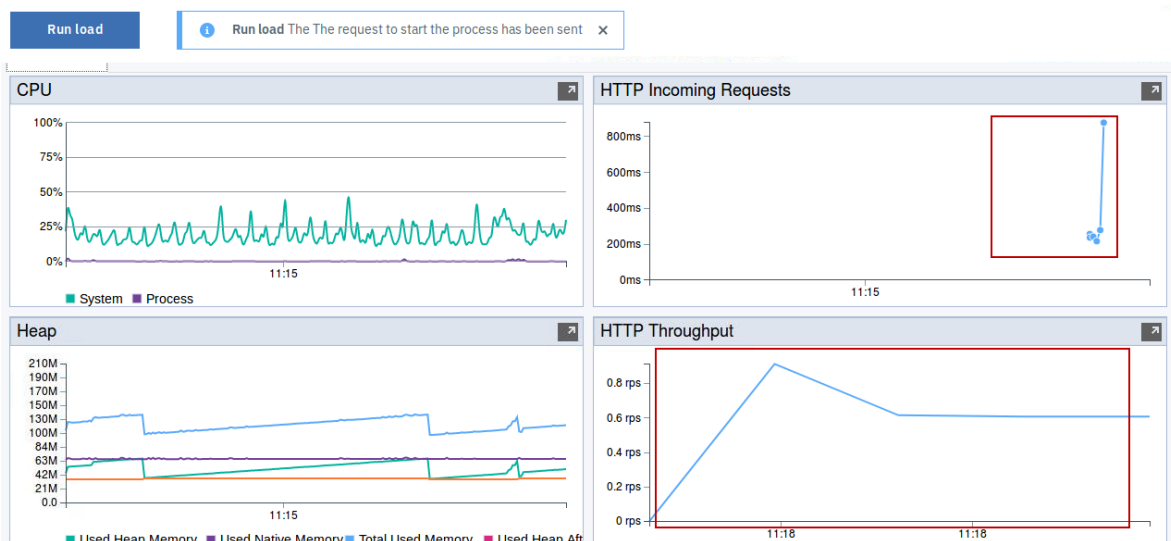
stockquote ●

Run load

\_\_3. Within a moment, the HTTP requests and Throughput charts will show the LIVE data.

Remember, we have set the Test Plan to only run 20 executions.

While the data is limited, you can clearly see that the data is live. Developers should find this very useful to do early performance testing of services to ensure they meet the SLA and conform to performance budgets set by the business. At a minimum, developers will get a very early indicator if performance is really out of line with expectations, and adjust. Microclimate is providing the means for development teams to **shift left**, very important non-functional testing for early indicators.



\_\_4. From the Performance Dashboard, click on the **Summary** tab.

stockquote ●

Run load

Run load The The request to start the process has been sent

Application Metrics for Java

Dashboard

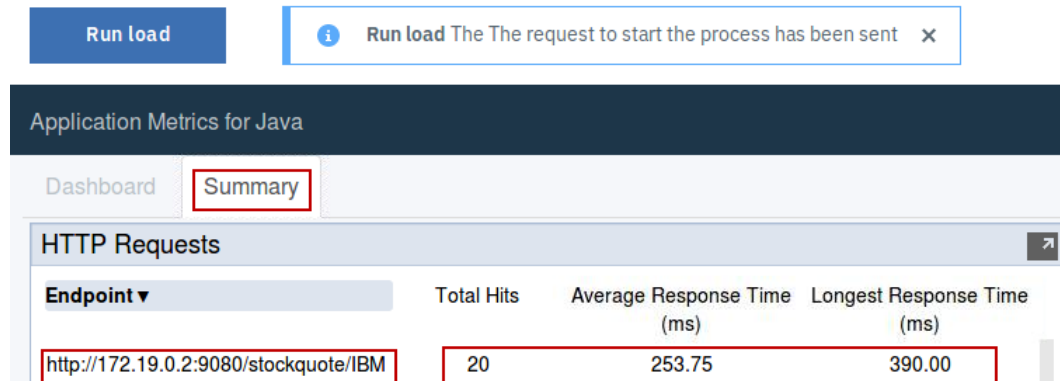
Summary



The **Summary** tab is useful to see the summary data for the endpoints tested during the load test. In our case, we tested the **stockquote/{symbol}** endpoint.

From the summary page, you can review the Total number of hits to the endpoint, the average response time, and the longest running transaction.

stockquote ●



===== Congratulations! You have completed Part 1 of the lab =====

In Part 2 of the lab, you will use the integrated CICD pipeline to deploy the stockquote microservice to IBM Cloud Private (Kubernetes Cluster).

## Part 2: Deploy the microservice application to IBM Cloud private using the integrated Jenkins build pipeline

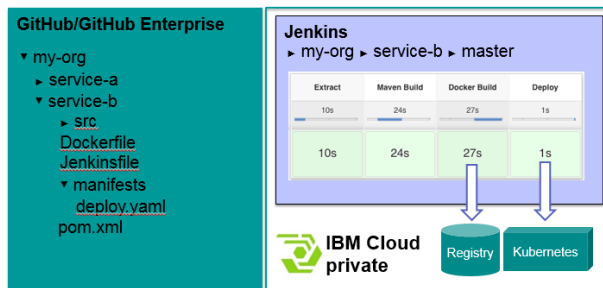


**Part 2** of the lab requires completion of Part 1 of the lab, including Part 0, setting up your personal GitHub Organization.

This part focuses on Microclimate integration with Jenkins pipeline to build and deploy your microservice application to IBM Cloud private.

### Microservice Builder pipeline

Automated build and deployment of microservices for rapid delivery



Microclimate uses Jenkins to build and deploy your code.

Our baseline assumption is that one microservice lives in one Git repository, and is built according to the Jenkinsfile in the project's root directory.

The name of the repository should be the same as the name of the Docker image that it produces.

Here is a standard Jenkinsfile:

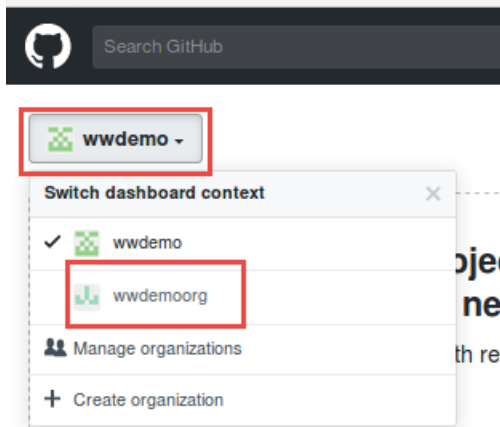
```
@Library('MicroserviceBuilder') _
microserviceBuilderPipeline {
    image = 'microservice-test'
}
```

## 2.1 Create repository in GitHub

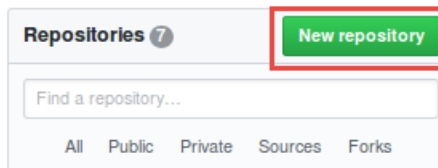
For this part of the lab, you need to create a Github repository. You will create a new repository under the Github organization you created/selected in Part 0 of the lab. You will create a GitHub repository named **stockquote**, and then push the microservice code to the remote repository.

**Note:** Microclimate needs a GitHub repository under your organization to store your microservice application code. In general, the name of this repository should be as same as your local microservice project name.

- \_\_1. In the browser, open a new browser tab, Click on the **GitHub** bookmark
- \_\_2. Sign in to **GitHub.com** using your own Github credentials
- \_\_3. Click on the **organization** selection button, and select your organization



- \_\_4. In the **Repositories** tab, click **New repository** to create a new repository.



- \_\_5. Enter the name of the new repository as: **stockquote**, make it a public repository by setting the **Public** option, and click **Create repository**.

Create a new repository

A repository contains all the files for your project, including the revision history.

---

Owner: **kpostreich** / Repository name: **stockquote** ✓

Great repository names are short and memorable. Need inspiration? How about **silver-potato**.

Description (optional)

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None** ⓘ


**Create repository**

Your GitHub repository is now created. Later in the lab you will populate this repository with code, and the Microclimate CICD pipeline in IBM Cloud Private will fetch the code and build it into a service.

- \_\_\_6. You will use the URL and some of the git commands for the repository later. Open the cheat sheet on the desktop, and copy these to the cheat sheet.

### Example GitHub HTTPS URL:

`https://github.com/<Your_GitHub_Org>/stockquote.git`

**TIP:** You can use the Copy to Clipboard  tool to copy the content from the browser, and then paste the clipboard content into the cheat sheet

**Quick setup — if you've done this kind of thing before**

or **HTTPS** **SSH** `https://github.com/kpostreich/stockquote.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# stockquote" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kpostreich/stockquote.git
git push -u origin master
```

GitHub HTTPS URL: \_\_\_\_\_

Git commands: \_\_\_\_\_

- \_\_\_7. You will use these commands later in this section to add the code from Microclimate to the GitHub repository. The default commands only add the README.md file, but you want to add all the files that are generated by Microclimate. Therefore, you have to make a modification. In the git commands, change the line from

**git add README.md**

to

**git add .**

- \_\_8. Save the cheat sheet
- \_\_9. You have now created a repository in GitHub, and recorded the values for the Access Token, Client ID and Client Secret in the cheat sheet (in the intro section of the lab), together with Git commands needed to get the code from your workstation into the repository. The next step will be to get the code into the repository to get ready for a pipeline to automatically build and deploy the application to ICP.

## 2.2 Adding the source code of your microservice project to GitHub

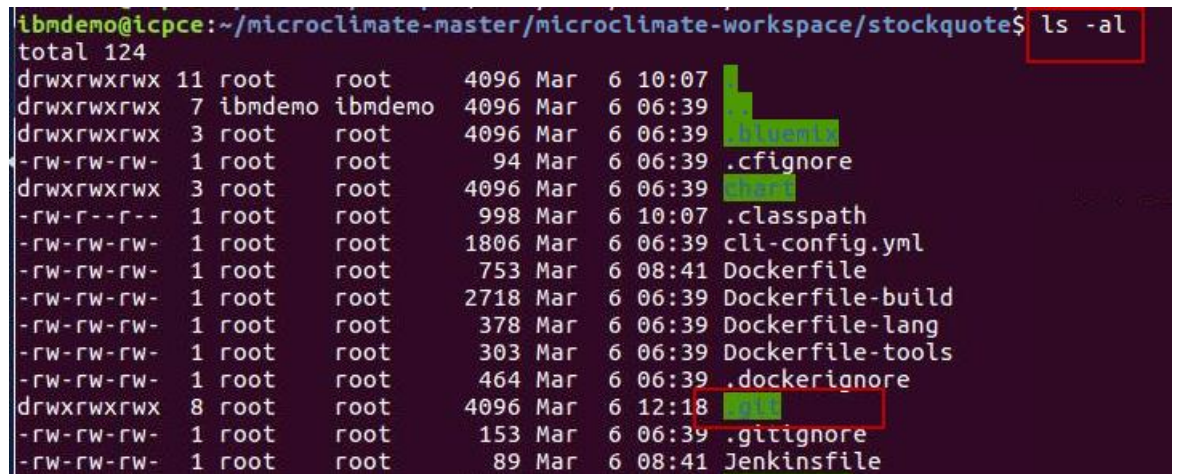
- \_\_1. Now you are ready to add your microservice project source code to GitHub repository you created, using the GitHub commands you recorded in the **Cheatsheet**.

- \_\_a. Go back to a terminal window, and navigate to your **stockquote** microservice project directory.

```
cd /home/ibmdemo/microclimate-master/microclimate-workspace/stockquote
```

- \_\_b. Microclimate has already pre-configured and initialized a local git repository in your stockquote project. The git repo is in directory **.git**, under the stockquote project.

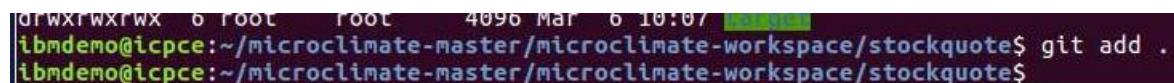
- I. From the command window, enter the command: **ls -al** to see the .git hidden directory.



```
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$ ls -al
total 124
drwxrwxrwx 11 root    root    4096 Mar  6 10:07 .
drwxrwxrwx  7 ibmdemo ibmdemo 4096 Mar  6 06:39 ..
drwxrwxrwx  3 root    root    4096 Mar  6 06:39 .bluewinx
-rw-rw-rw-  1 root    root     94 Mar  6 06:39 .cfignore
drwxrwxrwx  3 root    root    4096 Mar  6 06:39 .class
-rw-r--r--  1 root    root    998 Mar  6 10:07 .classpath
-rw-rw-rw-  1 root    root   1806 Mar  6 06:39 cli-config.yml
-rw-rw-rw-  1 root    root    753 Mar  6 08:41 Dockerfile
-rw-rw-rw-  1 root    root   2718 Mar  6 06:39 Dockerfile-build
-rw-rw-rw-  1 root    root    378 Mar  6 06:39 Dockerfile-lang
-rw-rw-rw-  1 root    root    303 Mar  6 06:39 Dockerfile-tools
-rw-rw-rw-  1 root    root    464 Mar  6 06:39 .dockerignore
drwxrwxrwx  8 root    root    4096 Mar  6 12:18 .git
-rw-rw-rw-  1 root    root    153 Mar  6 06:39 .gitignore
-rw-rw-rw-  1 root    root     89 Mar  6 08:41 Jenkinsfile
```

- \_\_c. Add your project source code to the local git directory with command:

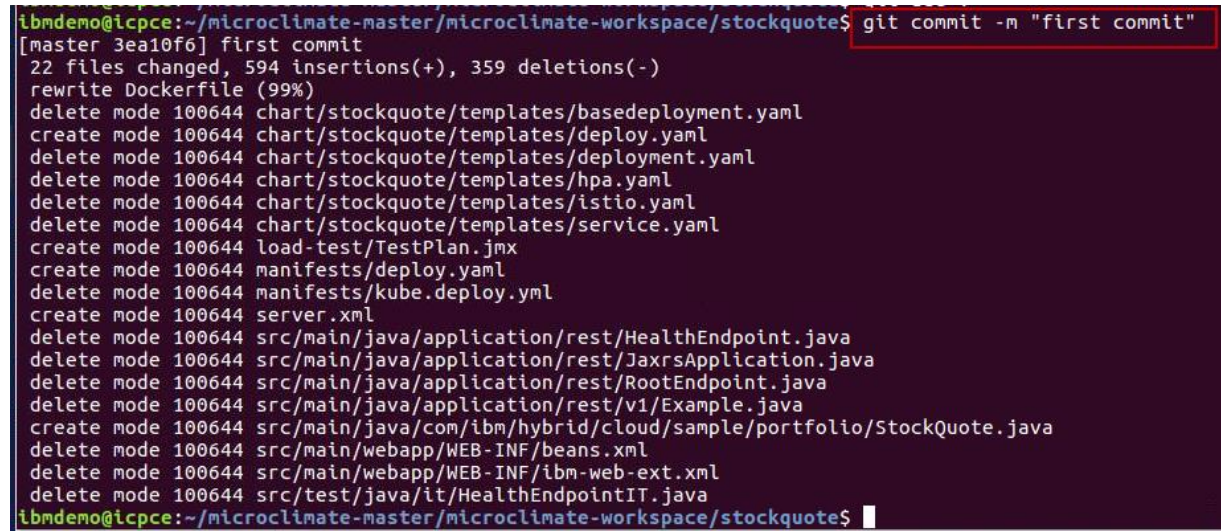
`git add .` **Note:** Include the period '.' In the command



```
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$ git add .
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$
```

- \_\_d. Issue git commit command to commit your code to your local git repo.

```
git commit -m "first commit"
```



```
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$ git commit -m "first commit"
[master 3ea10f6] first commit
22 files changed, 594 insertions(+), 359 deletions(-)
rewrite Dockerfile (99%)
delete mode 100644 chart/stockquote/templates/basedeployment.yaml
create mode 100644 chart/stockquote/templates/deploy.yaml
delete mode 100644 chart/stockquote/templates/deployment.yaml
delete mode 100644 chart/stockquote/templates/hpa.yaml
delete mode 100644 chart/stockquote/templates/istio.yaml
delete mode 100644 chart/stockquote/templates/service.yaml
create mode 100644 load-test/TestPlan.jmx
create mode 100644 manifests/deploy.yaml
delete mode 100644 manifests/kube.deploy.yaml
create mode 100644 server.xml
delete mode 100644 src/main/java/application/rest/HealthEndpoint.java
delete mode 100644 src/main/java/application/rest/JaxrsApplication.java
delete mode 100644 src/main/java/application/rest/RootEndpoint.java
delete mode 100644 src/main/java/application/rest/v1/Example.java
create mode 100644 src/main/java/com/ibm/hybrid/cloud/sample/portfolio/StockQuote.java
delete mode 100644 src/main/webapp/WEB-INF/beans.xml
delete mode 100644 src/main/webapp/WEB-INF/ibm-web-ext.xml
delete mode 100644 src/test/java/it/HealthEndpointIT.java
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$
```

- \_\_e. Add the local source code to your GitHub account repository.

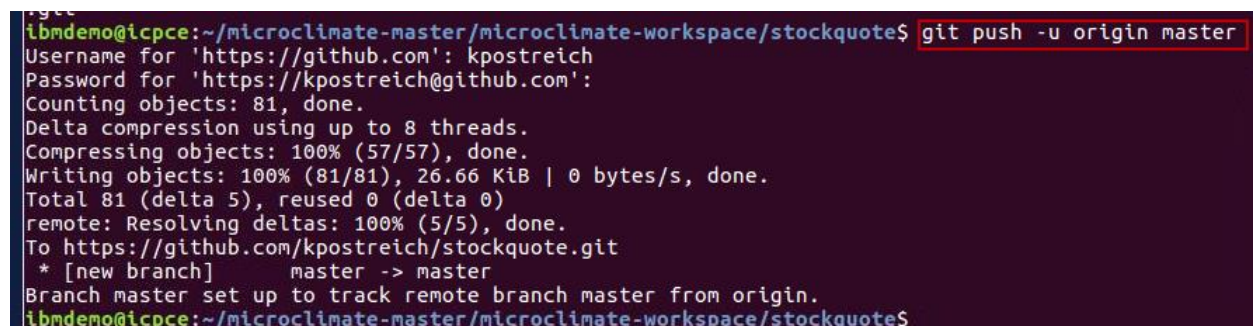
```
git remote add origin https://github.com/<your Github org>/stockquote.git
```

**Note:** Refer to the **Cheatsheet** to gather your GitHub Org, if needed.

- \_\_f. Push the code to your public GitHub project.

```
git push -u origin master
```

When prompted for Username and Password, enter your GitHub login credentials to push the contents to your GitHub project.

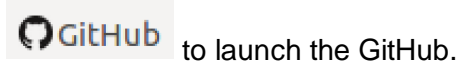


```
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$ git push -u origin master
Username for 'https://github.com': kpostreich
Password for 'https://kpostreich@github.com':
Counting objects: 81, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (57/57), done.
Writing objects: 100% (81/81), 26.66 KiB | 0 bytes/s, done.
Total 81 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/kpostreich/stockquote.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
ibmdemo@icpce:~/microclimate-master/microclimate-workspace/stockquote$
```

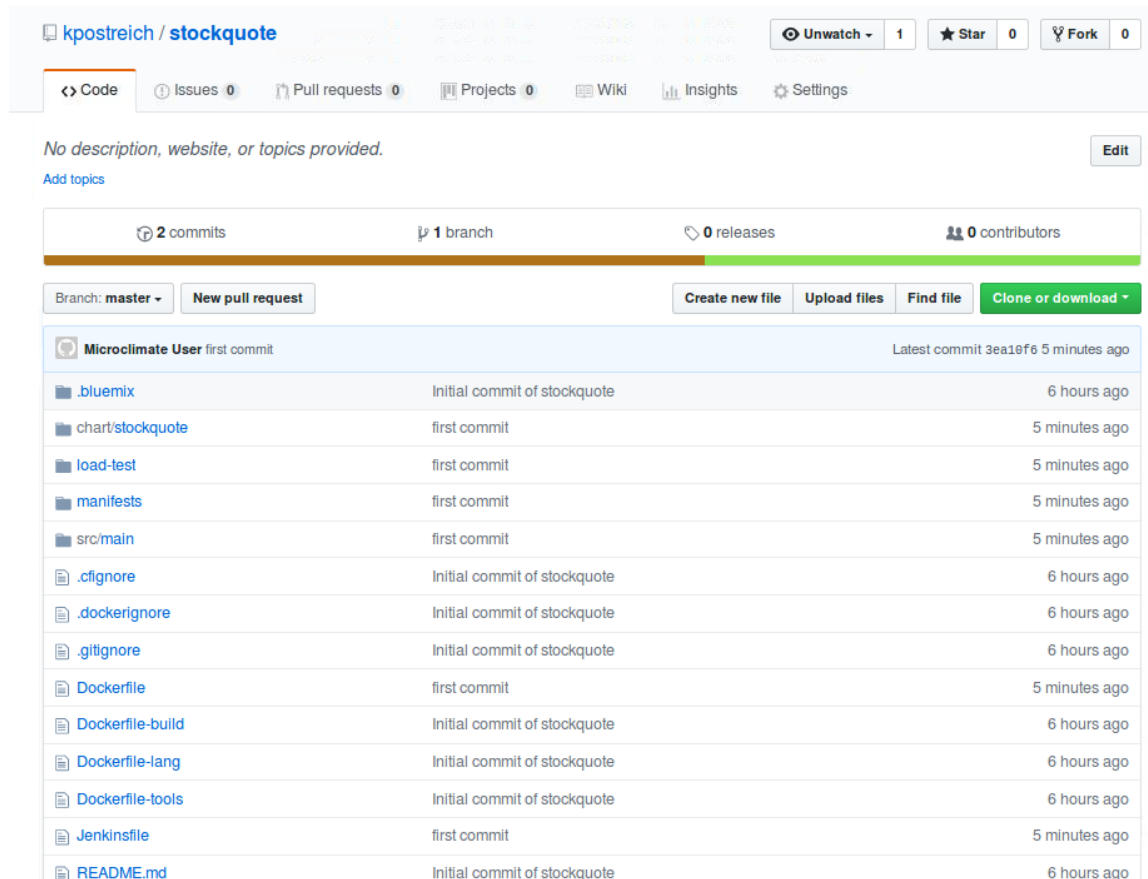
Your microservice project code will be uploaded to your GitHub repository.



- \_\_\_g. To verify your code upload, go back to your Github repository in the **Firefox** web browser window; if you closed the Github browser tab, click GitHub bookmark




- \_\_\_h. In the GitHub page, navigate to the **stockquote** project under your **organization** and you can see all your project source code are uploaded.



You have successfully uploaded your project code to Github.

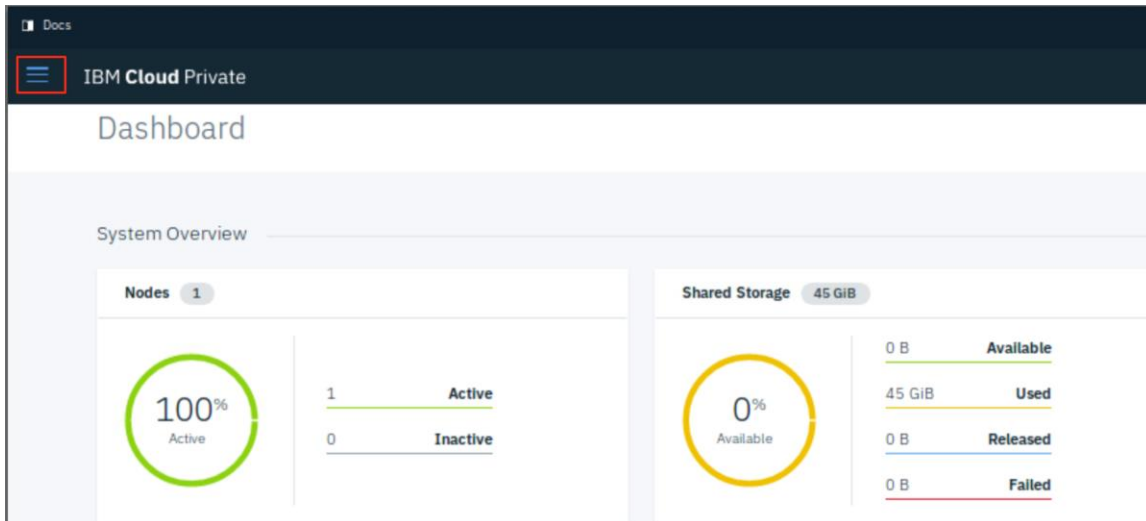
## 2.3 Microclimate pipeline configuration and deployment

A key component of Microclimate is the DevOps pipeline application running in the IBM Cloud Private environment. In this section, you are going to deploy the Microclimate pipeline into IBM Cloud Private, using the IBM supplied Helm chart.

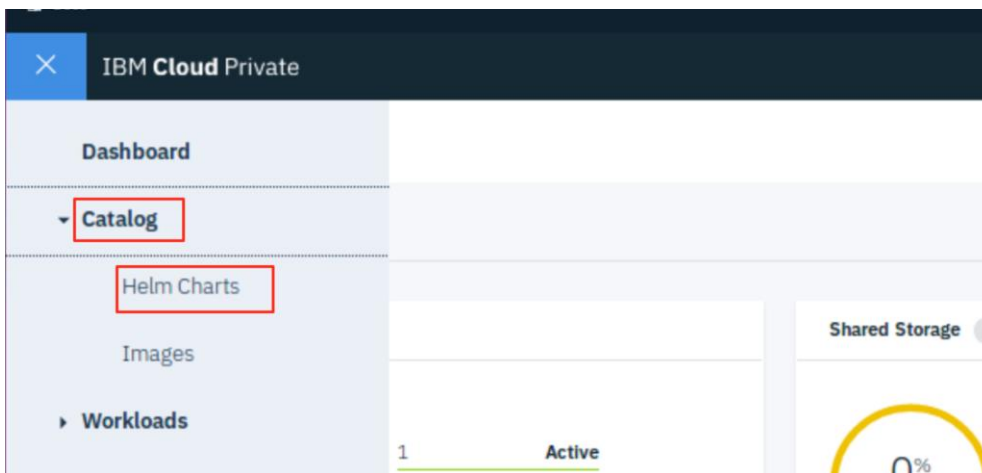
- \_\_\_1. Launch the **Firefox** web browser from the lab VM Desktop. 
- \_\_\_2. Click on the Browser tab that has your IBM Cloud Private Dashboard running. Alternatively, click on the **IBM Cloud private** bookmark in the Bookmarks Toolbar to launch the ICP dashboard.
- \_\_\_3. If prompted to login, accept the default User ID and Password (**admin / admin**) and click **Login**

Once you logged in, you see the ICP dashboard as shown below.

- \_\_\_4. Click the menu icon  on the top left corner.



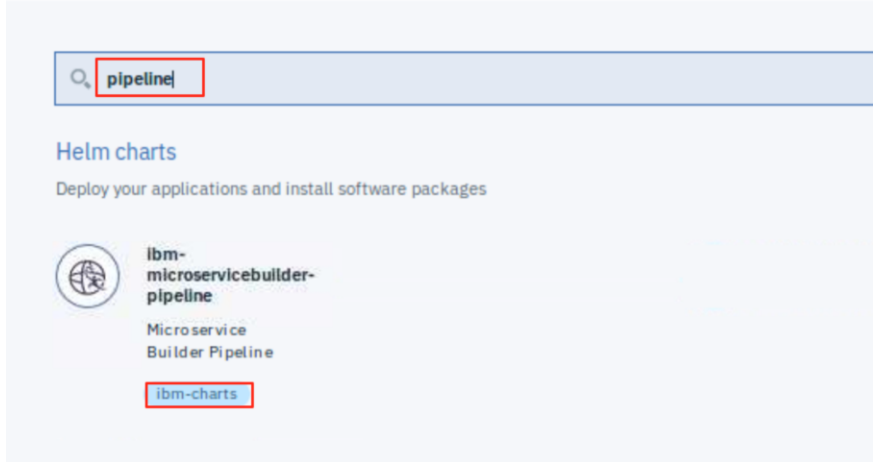
- \_\_\_5. Click **Catalog > Helm Charts**



- \_\_\_6. In the **Helm Charts** page, enter **pipeline** in filter field and click the **ibm-microservicebuilder-pipeline** chart to install it.



## Catalog



\_\_7. Click **Configure** in the bottom of the installation page.

You will configure the pipeline to use your Github user, Org, app secret, OAuth token, and Client ID for the Microclimate pipeline to access your Github project.

Pipeline.TargetNamespace	The Kubernetes namespace to use for this pipeline	
Pipeline.Template.RepositoryUrl	The location of the Git repository from which the microserviceBuilderPipeline.groovy library is obtained.	https://github.com/WASdev/microservicebuilder.lib.git
Pipeline.Template.Version	Version of the groovy library	1.0.0
Pipeline.Test	Setting this to true enables testing in the pipeline.	true
Pipeline.LibertyLicenseJar.BaseUrl	Optionally defines the location of a license upgrade JAR file	

[Configure](#)

\_\_8. In the **Configuration** section, enter the following information:

- **Release name:** stockquote-release
- **Target namespace:** select default from the drop-down menu
- Check the **license agreement** box

**Configuration**

Microservice Builder Pipeline Edit these parameters for configuration

<b>Release name</b> ⓘ	<b>Target namespace</b> ⓘ
stockquote-release	default

☒ I have read and agreed to the [license agreements](#)

\_\_\_9. In the **GitHub** section, enter the following information, using the data that you entered into the cheat sheet.

- **Orgs:** <YOUR GITHUB ORG> from your Cheatsheet
- **OAuth.Token:** <YOUR GITHUB OAUTH TOKEN> from your cheat sheet
- **App.Id:** <YOUR GITHUB APP ID> from your cheat sheet
- **Admins:** <YOUR GITHUB USER NAME>
- **RepoPattern:** `stockquote.*`
- **OAuth.User:** <YOUR GITHUB USER NAME>
- **App.Secret:** <YOUR GITHUB APP SECRET>

**Tip:** Double check the values you paste into the fields before you continue. It is easy to paste the same value into multiple fields by mistake.

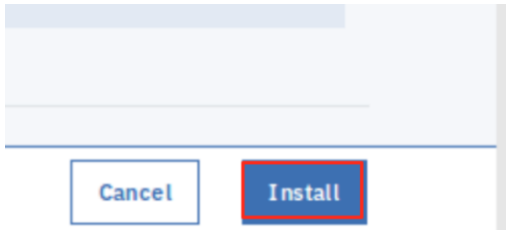
GitHub	
Url	Name
https://github.com	GitHub
Orgs	RepoPattern
kpa...	stockquote.*
OAuth.Token	OAuth.User
ae5cf...	kpa...
App.Id	App.Secret
5ac...	d1ee1cd...
Admins	
kpa...	

\_\_\_10. In the **Pipeline** section, set the **Test step of the pipeline** to **false**.

For the lab, we will skip the Maven integration tests in order to speed up the CICD pipeline process. In reality, you would likely run your test suites as part of the build pipeline.

Pipeline	
Build	Deploy
true	true
Test	Debug
false	false
DeployBranch	Registry.Url
master	mycluster.icp:8500/default

- \_\_11. Click the **Install** button to install the pipeline application to IBM Cloud Private



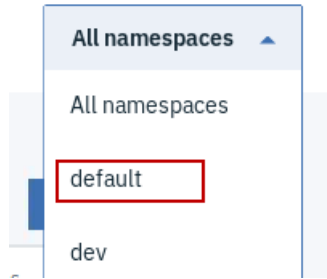
- \_\_12. Your Microclimate pipeline application is now deployed.
- \_\_a. Go to ICP dashboard menu and select **Workloads > Helm Releases**. You will see the pipeline release listed and in the DEPLOYED state.

### Helm releases

Search items					
20	items per page	1-8 of 8 items	1 of 1 pages		
NAME ^	NAMESPACE ^	STATUS ^	UPDATED ^	CHART NAME ^	CHART VERSION ^
<a href="#">loyaltylevel-chart</a>	default	DEPLOYED	Jan 23rd 2018	loyaltylevelchart	0.1.0
<a href="#">my-release</a>	default	DEPLOYED	Jan 4th 2018	redis	1.0.1
<a href="#">notification-chart</a>	default	DEPLOYED	Jan 22nd 2018	notificationchart	0.1.0
<a href="#">portfolio-chart</a>	default	DEPLOYED	Jan 23rd 2018	portfoliochart	0.1.0
<a href="#">stockquote-release</a>	default	DEPLOYED	Mar 6th 2018	ibm-microservicebuilder-pipeline	2.1.0
<a href="#">trader-chart</a>	default	DEPLOYED	Jan 22nd 2018	traderchart	0.1.0
<a href="#">yt-1</a>	default	DEPLOYED	Jan 22nd 2018	ibm-db2oltp-dev	1.1.1
<a href="#">yt-3</a>	default	DEPLOYED	Jan 22nd 2018	ibm-mqadvanced-server-dev	1.1.0

- \_\_13. Ensure that the pipeline workload is fully available in IBM Cloud Private
- \_\_a. Go to ICP dashboard menu and select **Workloads > Deployments**.
- \_\_b. From the **All Namespaces** filter, select the “**default**” namespace from the list

**Note:** You will now only see the workload deployments that are in the default namespace. The pipeline application is deployed to this namespace.



\_\_c. The **stockquote** pipeline should show the number of **AVAILABLE** deployments as: **1**

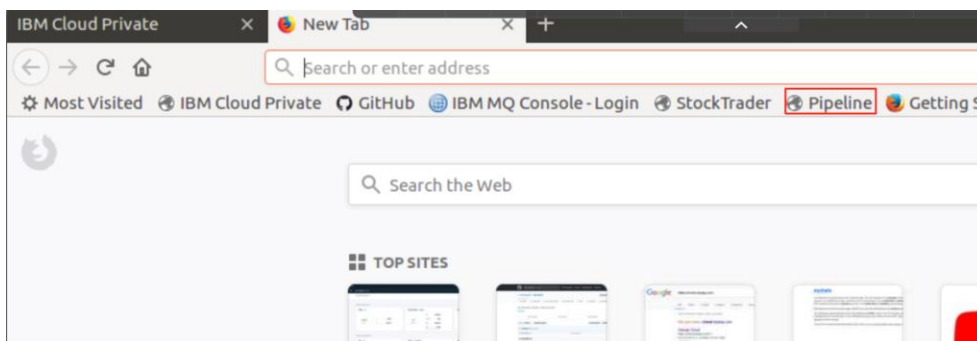
Deployments default

Search items Create Deployment +

20 Items per page | 1-8 of 8 items 1 of 1 pages < 1 >

NAME	NAMESPACE	DESIRED	CURRENT	READY	AVAILABLE	CREATION TIME	ACTION
<a href="#">loyalty-level</a>	default	1	1	1	1	Jan 23rd 2018 at 12:20 PM	⋮
<a href="#">my-release-redis</a>	default	1	1	1	1	Jan 4th 2018 at 9:38 AM	⋮
<a href="#">notification</a>	default	1	1	1	1	Jan 22nd 2018 at 12:27 PM	⋮
<a href="#">portfolio</a>	default	1	1	1	1	Jan 23rd 2018 at 12:19 PM	⋮
<a href="#">stockquote</a>	default	1	1	1	1	Mar 6th 2018 at 2:34 PM	⋮
<a href="#">stockquote-release-ibm-microservicebuilder-pipeline</a>	default	1	1	1	1	Mar 6th 2018 at 2:28 PM	⋮
<a href="#">trader</a>	default	1	1	1	1	Jan 22nd 2018 at 12:27 PM	⋮
<a href="#">yt-1-ibm-db2oltp-dev</a>	default	1	1	1	1	Jan 22nd 2018 at 11:12 AM	⋮

\_\_14. Open a new browser tab and click the **Pipeline** bookmark to launch it.



\_\_15. In the Jenkins page, you may see the pipeline put the build job to the Build Queue, or the build job may have already been picked up by the Build Executor.

- \_\_a. If the job is there, wait until it is done and it says **No builds in the queue**, and the Build Executor Status is empty
- \_\_b. Then click on the link with your **Github organization name** on it.

**Note:** This will take you to the Jenkins job that scans your GitHub repository for updates.

The screenshot shows the Jenkins dashboard. On the left is a sidebar with navigation links. The main area has two sections: 'Build Queue' and 'Build Executor Status'. The 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows three slave nodes: 'jenkins-slave-lb9zg-8vb38', 'jenkins-slave-lb9zg-hb6mv', and 'jenkins-slave-lb9zg-j3l94'. The third node is expanded, showing a build for 'kpostreich = stockquote' on the 'master' branch. The build is marked as successful with a green checkmark.

- \_\_16. Click on **stockquote** and then on **master** to see the details for the build job

The screenshot shows the Jenkins job details page for 'stockquote'. It has two tabs: 'Repositories (1)' and 'Pull Requests (0)'. The 'Repositories (1)' tab is active, showing a table with one repository: 'stockquote'. The 'Branches (1)' tab is also visible, showing a table with one branch: 'master'. Both the repository name and the branch name are highlighted with red boxes.

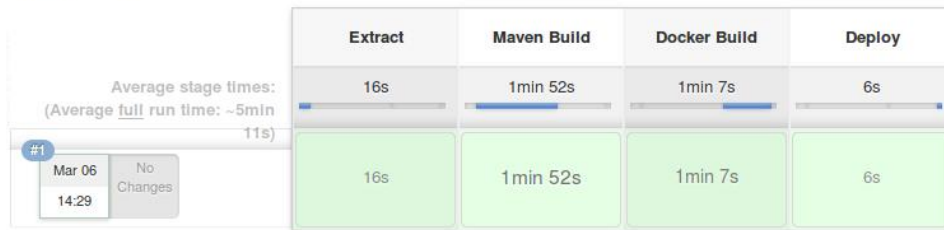
- \_\_\_17. You will see the different stages of the build job: Extract, Maven Build, Docker Build, and Deploy. When the job is complete, it may look like this.

## Branch master

Full project name: kpostreich/stockquote/master

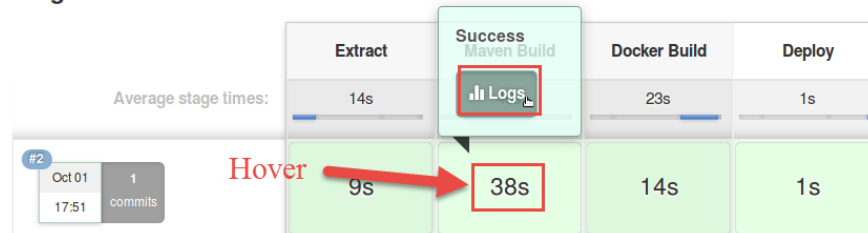


### Stage View



- \_\_\_18. You can access the logs for each stage by hovering over each item, then clicking **Logs**.

### Stage View



- \_\_\_19. You can scroll up and down and investigate the logs if needed

```
Stage Logs (Maven Build)
Shell Script -- mvn -B clean package -- (self time 1min 50s)
Q7T70VAPRZYEU7CWKQYDNPZ64V5XSRVIF3W6M76IFPFLKLH4JQ/target/liberty/wlp/usr/servers/defaultServer.
[INFO] CWWM2001I: server.output.dir is /home/jenkins/workspace/postreich_stockquote_master-P
Q7T70VAPRZYEU7CWKQYDNPZ64V5XSRVIF3W6M76IFPFLKLH4JQ/target/wlp-package/defaultServer.
[INFO] CWWM2001I: Invoke command is [/home/jenkins/workspace/postreich_stockquote_master-PQ7
T70VAPRZYEU7CWKQYDNPZ64V5XSRVIF3W6M76IFPFLKLH4JQ/target/liberty/wlp/bin/server, package, def
aultServer, --archive=/home/jenkins/workspace/postreich_stockquote_master-PQ7T70VAPRZYEU7CWKQ
YDNPZ64V5XSRVIF3W6M76IFPFLKLH4JQ/target/stockquote-1.0-SNAPSHOT.zip, --include=usr].
[INFO] Packaging server defaultServer.
[INFO] Server defaultServer package complete in /home/jenkins/workspace/postreich_stockquote_
master-PQ7T70VAPRZYEU7CWKQYDNPZ64V5XSRVIF3W6M76IFPFLKLH4JQ/target/stockquote-1.0-SNAPSHOT.zi
p.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:32 min
[INFO] Finished at: 2018-03-06T22:33:40Z
[INFO] Final Memory: 33M/434M
[INFO] -----
```

When the job has completed successfully, i.e., all the stages are green, the service will be deployed to IBM Cloud Private Deployment as **stockquote**. The service is created as **stockquote-service**.

	Extract	Maven Build	Docker Build	Deploy
Success	21s	2min 28s	1min 27s	11s
Failure				
Warning				
Information				

**If your deployment fails with the following error:**

**Error: release stockquote failed: services "stockquote-service" already exists**

Here is the corrective actions to take:



- From the ICP UI, delete any existing instances of the **stockquote** from **Workloads > Deployments**
- From the ICP UI, delete any existing instances of the **stockquote-service** from **Network Access > Services**
- From Jenkins, execute the “**Build Now**” from the **master** branch to initiate a fresh build.

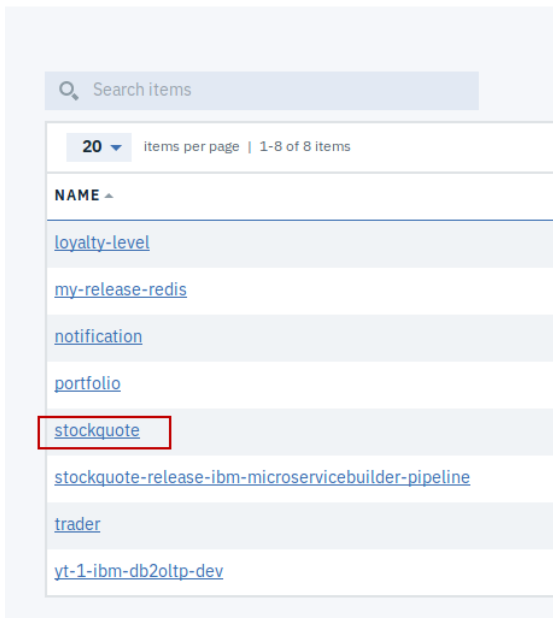
## 2.4 Test the microservice application in IBM Cloud Private

After your microservice application is deployed to Kubernetes cluster in the IBM Cloud private environment, via the Microclimate pipeline, you can run the application from a web browser.

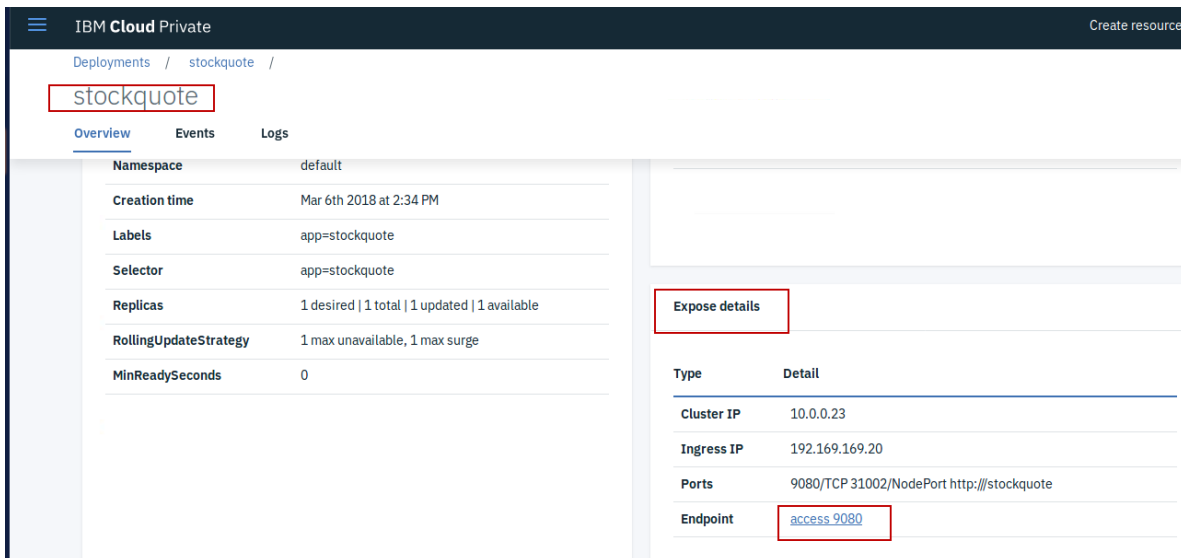
First, you need to determine the host or IP address and port number that the stockquote service is running on. Since this installation of IBM Cloud Private is a single Node installation, we already know the host IP address is **192.169.169.20**

- \_\_\_1. From the **Firefox** web browser window, open the **IBM Cloud Private** tab, or click on the Firefox bookmark to access the IBM Cloud private dashboard.
- \_\_\_2. Go to **Workloads > Deployments** page.
- \_\_\_3. Click on the **stockquote** deployment

## Deployments



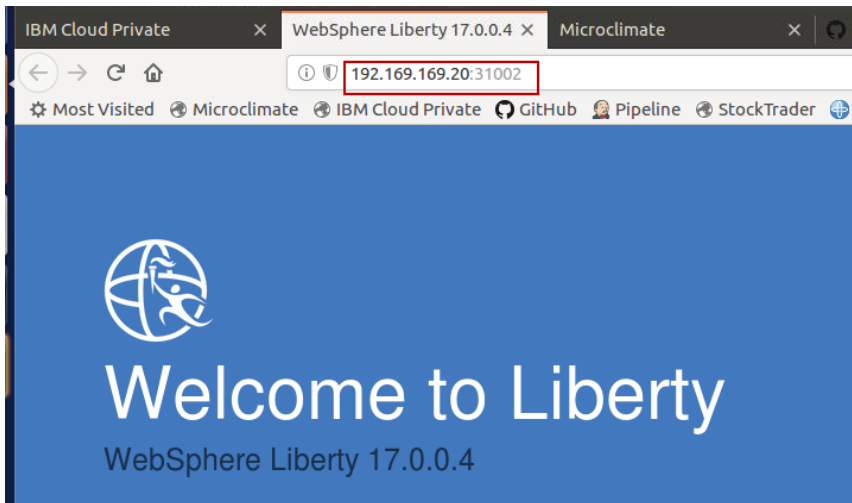
- \_\_\_4. From the **stockquote** Overview page, scroll down to view the **Expose Details** section. Then click on the Endpoint link: **access 9080**, located in the Expose Details section.



- \_\_\_5. This endpoint will take you to the **Welcome to Liberty** page, indicating that the server endpoint is accessible.

Notice the port number? This is an exposed port that Kubernetes assigned for our application. Your port number may be different than shown below.

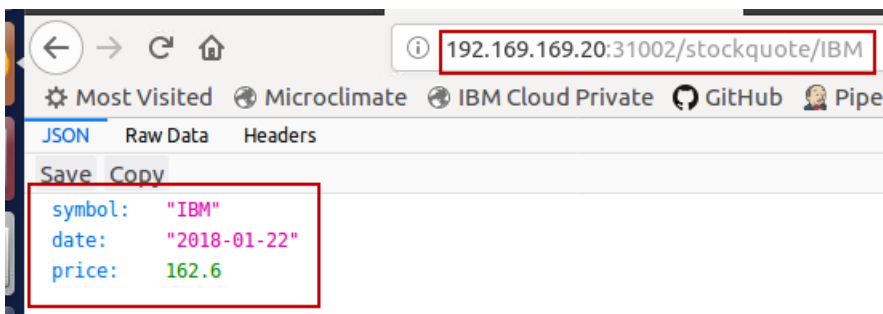




- \_\_\_6. To access the **stockquote** service endpoint, append **/stockquote/IBM** to the URL on the browser address bar.

The expected response is a json response that includes the symbol, date, and price of the stock you requested a quote for in the URL.

**Note:** the stockquote application may pull a quote from a local cache, or generate a DUMMY stock value, if the external Quandl API is not available, or we become rate limited by the service.



The stockquote application invokes a freely available stock API from Quandl. However, the free tier API has limits on the number of calls per day from anonymous users.

Therefore, it is possible that the stockquote service will return a DUMMY stock price, if Quandl returns an HTTP response code 429, indicating the maximum number of API calls per day is reached.

Congratulations! You have successfully created, deployed and tested your first microservice application in Kubernetes cluster deployed to IBM Cloud Private

===== You have successfully completed the lab =====