Spring Cloud Brewery on OpenShift

Setup Guide

Kamesh Sampath

Version {version}, 2017-08-09

Spring Cloud Samples: Brewery on OpenShift

Chapter 1. Spring Cloud Brewery on OpenShift

1.1. Overview

This guides walks you through to setup Spring Cloud Samples - Brewery on OpenShift. The steps in this guide could be the first step or a Proof of Technology(PoT) on how to migrate an existing Spring Cloud/Boot application on to Kubernetes or Openshift.

In the process of migration, the original Spring Cloud Samples - Brewery will be modified to make it deployable on to Kubernetes or Openshift.

The application will be migrated based on these iterations,

- ✓ Iteration I - As-is deployment of the Spring Cloud Samples - Brewery with minimal or no code change
- ✓ Iteration II - Will use native OpenShift/Kubernetes features such as service discovery, loadbalancing & externalization of the config
- [] Iteration III - Replace RabbitMQ with JBoss A-MQ
- [] Iteration IV - Service Mesh with Istio

1.2. Pre-Requisite

You have a OpenShift cluster running locally using **minishift** or **CDK**, or have access to **OpenShift Container Platform**

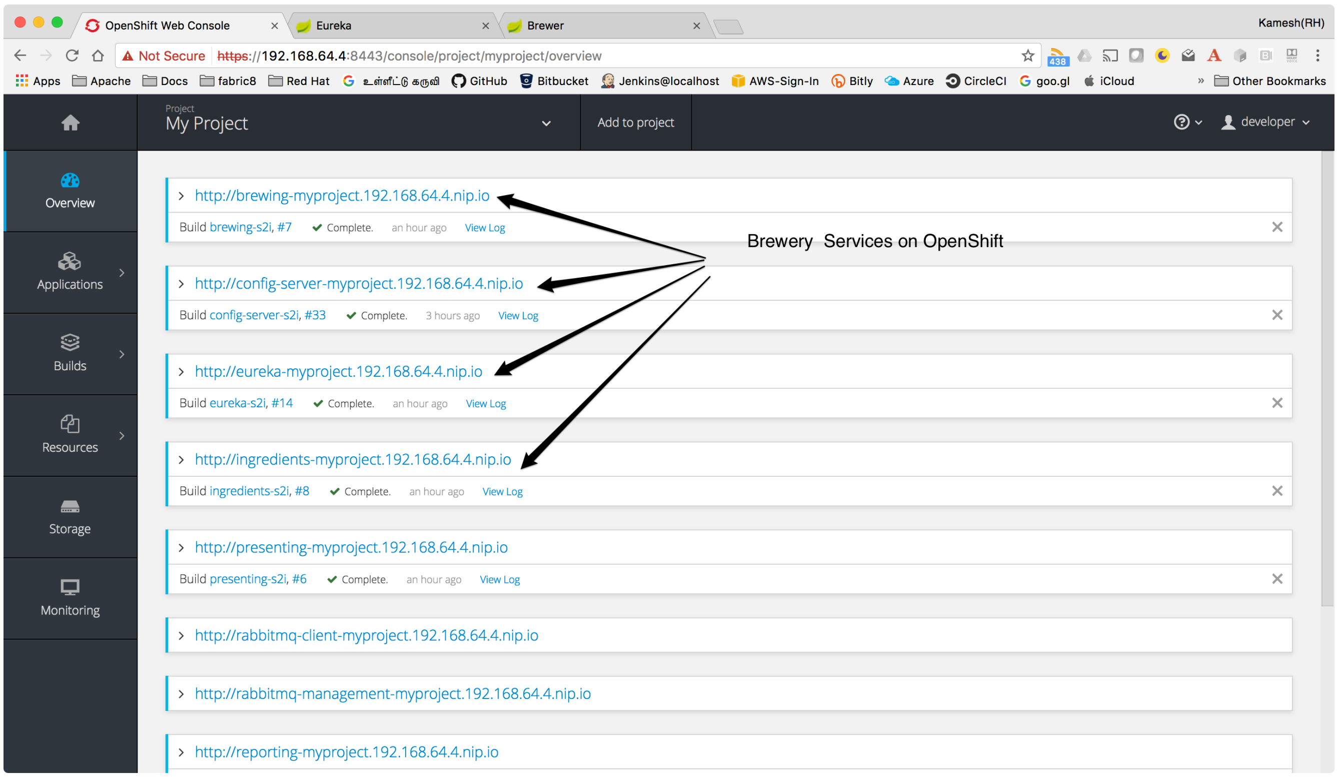
- At least 7Gb of RAM is required to run the Brewery application
- Openshift Origin 1.4.1 is required till we have fixed this issue
- You can boost the deployment if you pull these docker images used by the s2i process `docker pull fabric8/s2i-java:2.0` and `docker pull fabric8/java-jboss-openjdk8-jdk:1.2`

This is needed only in OpenShift and applicable for iteations 2 and above

```
oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default -n $(oc project -q)
```

1.3. Accessing the Applications

You can view the application urls from OpenShift Web Console. A successful deployment will have all the applications running with single pod. The following screenshots shows how the Eureka will look like when all the clients registers with it



http://brewing-myproject.192.168.64.4.nip.io
Build brewing-s2i, #7 ✓ Complete. an hour ago View Log

http://config-server-myproject.192.168.64.4.nip.io
Build config-server-s2i, #33 ✓ Complete. 3 hours ago View Log

http://eureka-myproject.192.168.64.4.nip.io
Build eureka-s2i, #14 ✓ Complete. an hour ago View Log

http://ingredients-myproject.192.168.64.4.nip.io
Build ingredients-s2i, #8 ✓ Complete. an hour ago View Log

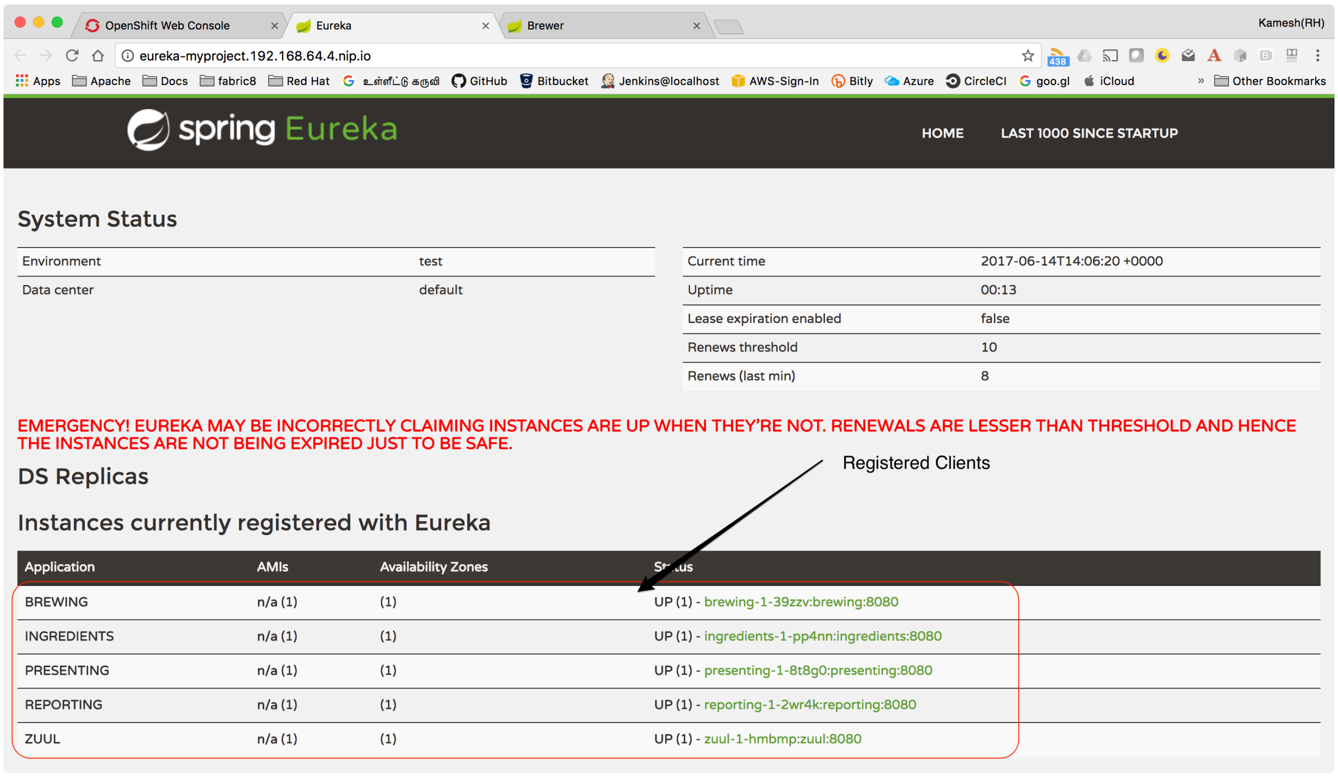
http://presenting-myproject.192.168.64.4.nip.io
Build presenting-s2i, #6 ✓ Complete. an hour ago View Log

http://rabbitmq-client-myproject.192.168.64.4.nip.io

http://rabbitmq-management-myproject.192.168.64.4.nip.io

http://reporting-myproject.192.168.64.4.nip.io

Figure 1. Brewery Services



System Status

Environment	test
Data center	default

Current time	2017-06-14T14:06:20 +0000
Uptime	00:13
Lease expiration enabled	false
Renews threshold	10
Renews (last min)	8

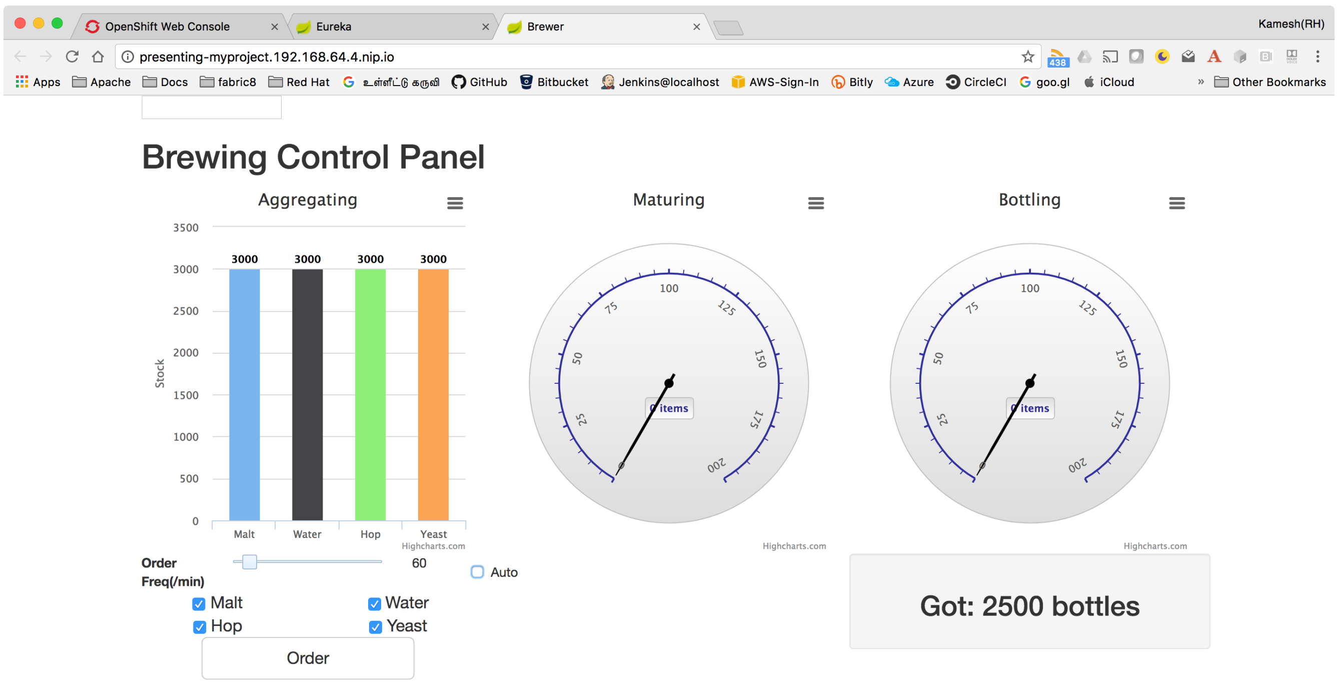
EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BREWING	n/a (1)	(1)	UP (1) - brewing-1-39zzv:brewing:8080
INGREDIENTS	n/a (1)	(1)	UP (1) - ingredients-1-pp4nn:ingredients:8080
PRESENTING	n/a (1)	(1)	UP (1) - presenting-1-8t8g0:presenting:8080
REPORTING	n/a (1)	(1)	UP (1) - reporting-1-2wr4k:reporting:8080
ZUUL	n/a (1)	(1)	UP (1) - zuul-1-hmbmp:zuul:8080

Figure 2. Eureka on OpenShift



Order Freq(/min) 60

Auto [ ]

[✓] Malt [✓] Water

[✓] Hop [✓] Yeast

Got: 2500 bottles

Figure 3. Brewer App

1.4. References

- Maven Properties
- Kubernetes
- Openshift

Chapter 2. Iteration I

The Iteration-1 is supposed to be the as-is deployment of the *Brewery* application on to Kubernetes or Openshift. This will have all the components from the original https://github.com/spring-cloud-samples/brewery with modifications required :

- Use the Fabric8 Maven Plugin to generate the Kubernetes/OpenShift resources needed to deploy the applications on the platform
- Design OpenShift templates top deploy RabbitMQ and the different servers such as Eureka, Config-Server, Zipkin, … (to be discussed)

to have it deployed on to Kubernetes or Openshift

2.1. Setup

2.1.1. Clone

```
git clone -b iteration-1 https://github.com/redhat-developer-demos/brewery.git
```

Through out this document we will call the directoy where the project was cloned as *$PROJECT_HOME*

2.1.2. Deploy Applications

Table 1. Application List

	Application	Folder	Remarks
	RabbitMQ	**$PROJECT_HOME**/extras/rabbitmq	Message Broker - https://www.rabbitmq.com/
	common	**$PROJECT_HOME**/common	Common shared library
	Eureka	**$PROJECT_HOME**/eureka	Service Registry - https://github.com/Netflix/eureka/wiki/Eureka-at-a-glance
	Config Server	**$PROJECT_HOME**/config-server	Centralized Configuration Server - https://cloud.spring.io/spring-cloud-config/spring-cloud-config.html
	Zipkin Server	**$PROJECT_HOME**/zipkin-server	Distributed Tracing system

	Application	Folder	Remarks
	Zuul	**$PROJECT_HOME**/zuul	Java based Proxy
	Ingredients	**$PROJECT_HOME**/ingredients	
	Reporting	**$PROJECT_HOME**/reporting	
	Brewing	**$PROJECT_HOME**/brewing	
	Presenting	**$PROJECT_HOME**/presenting	

2.1.3. Building

Brewery application uses gradle for build, we will leverage on the same to get the application artifacts ready. To build the applicaiton run the following command

```
./gradlew -DWHAT_TO_TEST="SLEUTH_STREAM" clean build ①
./mvnw -N install ②
```

① We will be using Spring Cloud Sleuth for sending trace information to Zipkin

② This will install the brewery parent pom in local maven repository

2.1.4. Deploying to OpenShift

As part of this lift and shift of existing application, to make it work as-is, there is certain order of applicaiton deployment might be required. The following section explains the deployment of the application in the same order as expected (you can expriment with it if you like :))

RabbitMQ

Go to the directory **$PROJECT_HOME/extras/rabbitmq**, and execute the following command

```
./mvnw -Dfabric8.mode=kubernetes clean fabric8:deploy
```

This will take some time to get it running as the deployment needs to download the `rabbitmq` docker image

Config Server

Go to the directory **$PROJECT_HOME/config-server**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Since this is the first Java application to be deployed, it may take some time to download the necessary images from docker hub.

Eureka

Go to the directory **$PROJECT_HOME/eureka**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Zipkin Server

Go to the directory **$PROJECT_HOME/zipkin-server**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Zuul

Go to the directory **$PROJECT_HOME/zuul**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Ingredients

Go to the directory **$PROJECT_HOME/ingredients**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Reporting

Go to the directory **$PROJECT_HOME/reporting**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Brewing

Go to the directory **$PROJECT_HOME/brewing**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Presenting

Go to the directory **$PROJECT_HOME/presenting**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Chapter 3. Iteration II

The Iteration II will drop/undeploy few of the NetFlix OSS components that are superflous inside Kubernetes or Openshift. The following sections shows how to get the Iteration II deployed on to Kubernetes or Openshift. This iteration uses the Spring Cloud Kubernetes - the Spring Cloud based discovery client for Kubernetes

3.1. Setup

3.1.1. Clone

```
git clone -b iteration-2 https://github.com/redhat-developer-demos/brewery.git
```

Through out this document we will call the directoy where the project was cloned as *$PROJECT_HOME*

3.2. Deploy Applications

Table 2. Application List

	Application	Folder	Remarks
	RabbitMQ	**$PROJECT_HOME**/extras/rabbitmq	Message Broker - https://www.rabbitmq.com/
	common	**$PROJECT_HOME**/common	Common shared library
X	eureka	**$PROJECT_HOME**/eureka	Application will use Kubernetes Services
X	config-server	**$PROJECT_HOME**/config-server	Application will use Kubernetes ConfigMaps
	Zipkin Server	**$PROJECT_HOME**/zipkin-server	Distributed Tracing system
	Zuul	**$PROJECT_HOME**/zuul	Java based Proxy

	Application	Folder	Remarks
	Ingredients	**$PROJECT_HOME**/ingredients	
	Reporting	**$PROJECT_HOME**/reporting	
	Brewing	**$PROJECT_HOME**/brewing	
	Presenting	**$PROJECT_HOME**/presenting	

3.2.1. Building

Brewery applicaiton uses gradle for build, we will leverage on the same to get the application artifacts ready. To build the applicaiton run the following command

```
./gradlew -DWHAT_TO_TEST="SLEUTH_STREAM" clean build ①
./mvnw -N install ②
```

① We will be using Spring Cloud Sleuth for sending trace information to Zipkin

② This will install the brewery parent pom in local maven repository

3.2.2. Deploying to OpenShift

As part of this lift and shift of existing application, to make it work as-is, there is certain order of applicaiton deployment might be required. The following section explains the deployment of the application in the same order as expected (you can expriment with it if you like :))

RabbitMQ

Go to the directory **$PROJECT_HOME/extras/rabbitmq**, and execute the following command

```
./mvnw -Dfabric8.mode=kubernetes clean fabric8:deploy
```

This will take some time to get it running as the deployment needs to download the `rabbitmq` docker image

Zipkin Server

Go to the directory **$PROJECT_HOME/zipkin-server**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Zuul

Go to the directory **$PROJECT_HOME/zuul**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Ingredients

Go to the directory **$PROJECT_HOME/ingredients**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Reporting

Go to the directory **$PROJECT_HOME/reporting**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Brewing

Go to the directory **$PROJECT_HOME/brewing**, and execute the following command

```
./mvnw clean fabric8:deploy
```

Presenting

Go to the directory **$PROJECT_HOME/presenting**, and execute the following command

```
./mvnw clean fabric8:deploy
```

3.3. Acceptance Testing

The **$PROJECT_HOME/acceptance-tests** holds the test cases for testing the application. To perform we need to have have some ports forwarded from Kubernetes/OpenShift to localhost(where you build the application)

```
oc port-forward zipkin-1-06wmt 9411:8080 ①
oc port-forward presenting-1-wzhfn 9991:8080 ②
```

① forward port 8080 from Zipkin pod to listen on localhost:9411

② forward port 8080 from Presenting pod to listen on localhost:9991

Please update the pod names based on your local deployment

To run acceptance testing, execute following command from $PROJECT_HOME,

```
./gradlew -DWHAT_TO_TEST="SLEUTH_STREAM" :acceptance-tests:acceptanceTests
```