

Spring Cloud Brewery on OpenShift

Setup Guide

Kamesh Sampath

Version {version}, 2017-08-31

Spring Cloud Samples: Brewery on OpenShift

1. Spring Cloud Brewery on OpenShift	2
1.1. Overview	2
1.2. Pre-Requisite	2
1.3. Accessing the Applications	2
1.4. References	4
2. Iteration I	5
2.1. Setup	5
2.1.1. Clone	5
2.1.2. Deploy Applications	5
2.1.3. Building	6
2.1.4. Deploying to OpenShift	6
RabbitMQ	6
Config Server	7
Eureka	7
Zipkin Server	7
Zuul	7
Ingredients	7
Reporting	7
Brewing	7
Presenting	8
3. Iteration II	9
3.1. Setup	9
3.1.1. Clone	9
3.2. Pre-Requisite	9
3.2.1. General	9
3.2.2. RabbitMQ	9
3.3. Deploy Applications	10
3.3.1. Building	11
3.3.2. Deploying to OpenShift	11
RabbitMQ	11
Zipkin Server	11
Zuul	12
Ingredients	12
Reporting	12
Brewing	12
Presenting	12
3.4. Acceptance Testing	12

3.5. Deprecated Modules	13
-------------------------------	----

Chapter 1. Spring Cloud Brewery on OpenShift

1.1. Overview

This guide 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](#)

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

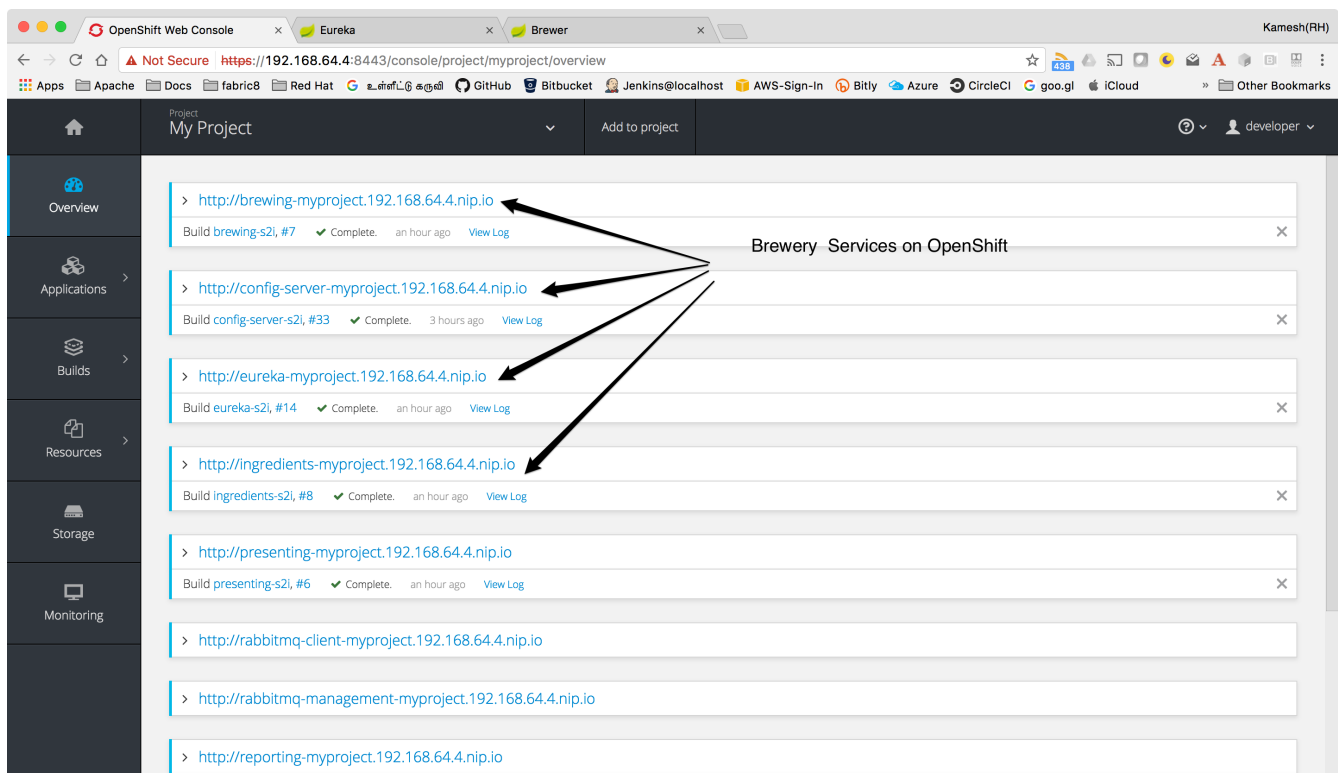


Figure 1. Brewery Services

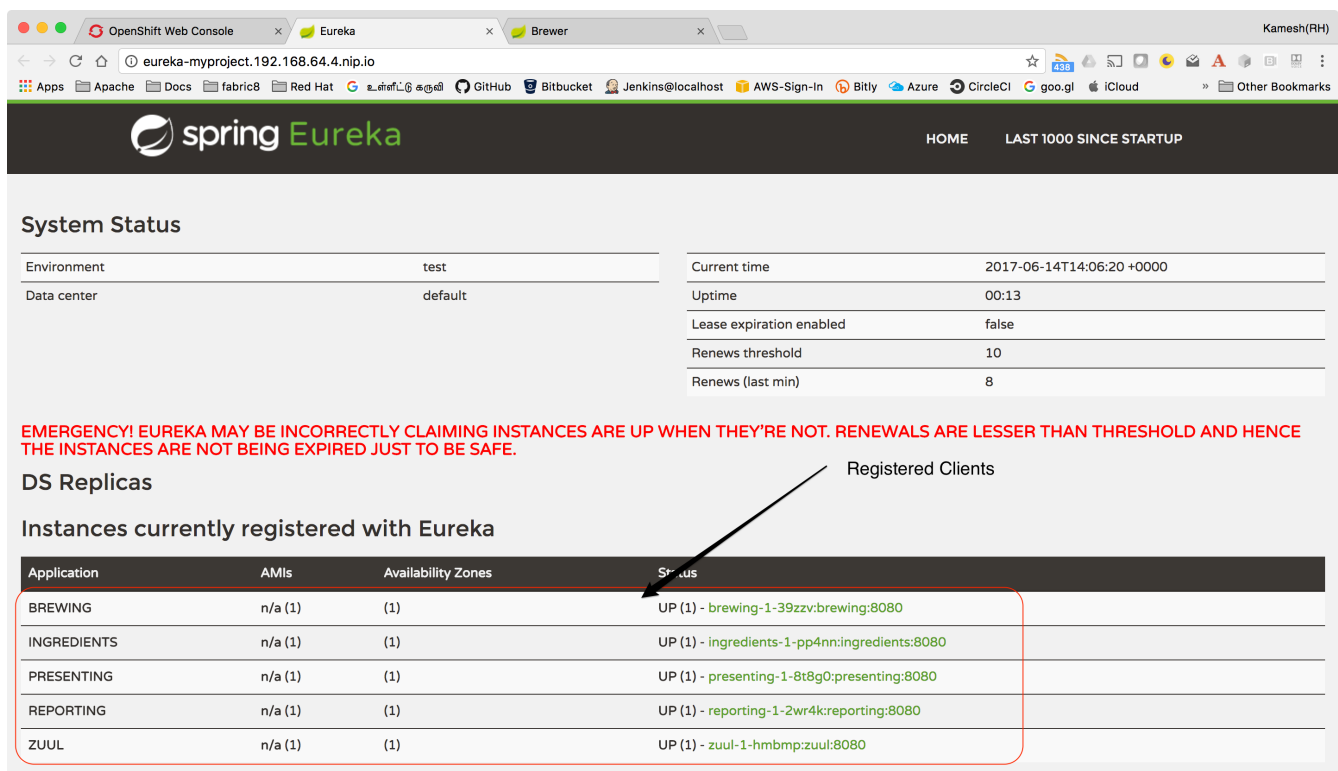


Figure 2. Eureka on OpenShift

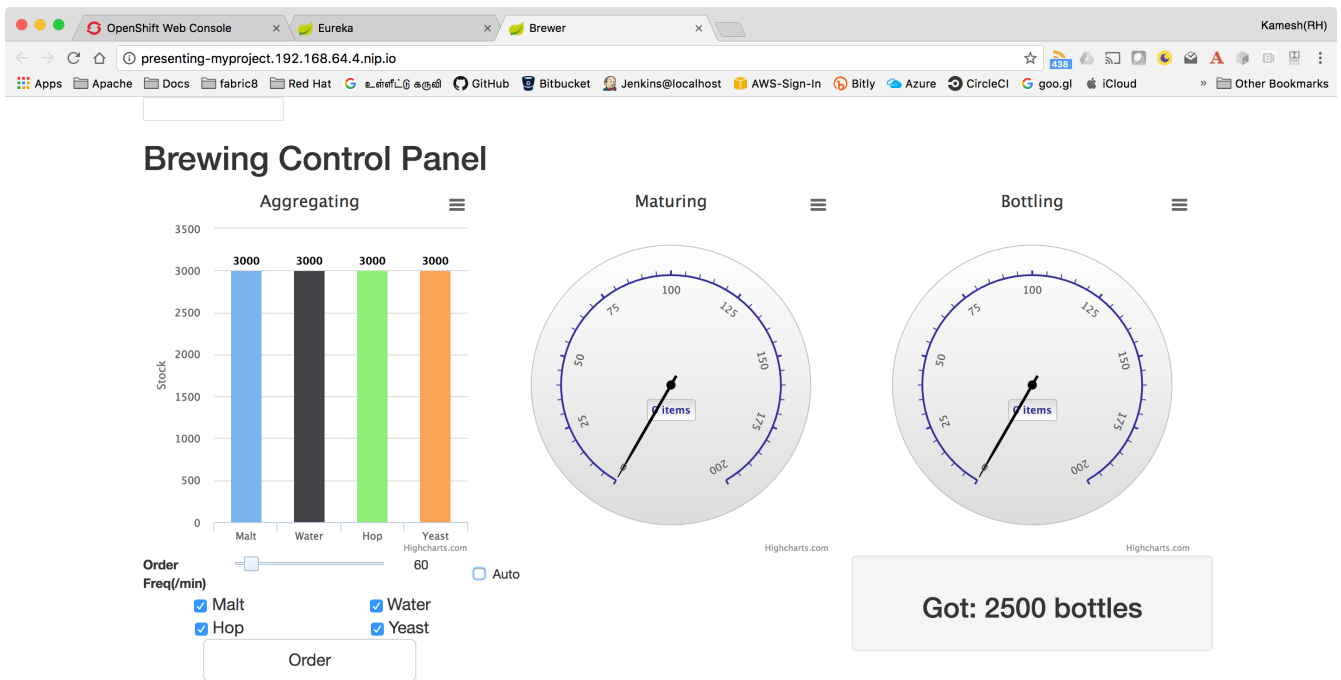


Figure 3. Brewer App

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

	Applicatio n	Folder	Remarks
	RabbitMQ	<code>\$PROJECT_HOME/ext ras/rabbit mq</code>	Message Broker - https://www.rabbitmq.com/
	common	<code>\$PROJECT_HOME/co mmon</code>	Common shared library
	Eureka	<code>\$PROJECT_HOME/eu reka</code>	Service Registry - https://github.com/Netflix/eureka/wiki/Eureka-at-a-glance
	Config Server	<code>\$PROJECT_HOME/co nfig-server</code>	Centralized Configuration Server - https://cloud.spring.io/spring-cloud-config/spring-cloud-config.html
	Zipkin Server	<code>\$PROJECT_HOME/zip kin-server</code>	Distributed Tracing system

	Application	Folder	Remarks
	Zuul	<code>\$PROJECT_HOME/zuul</code>	Java based Proxy
	Ingredients	<code>\$PROJECT_HOME/ingredients</code>	
	Reporting	<code>\$PROJECT_HOME/reporting</code>	
	Brewing	<code>\$PROJECT_HOME/brewing</code>	
	Presenting	<code>\$PROJECT_HOME/presenting</code>	

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 [deprecate](#) few of the [Netflix OSS](#) components that are superfluous 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 directory where the project was cloned as `$PROJECT_HOME`

3.2. Pre-Requisite

3.2.1. General

The `spring-cloud-kubernetes` library used in the project requires the `default` service account to have view permissions, to enable that we execute the following command,

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

3.2.2. RabbitMQ

The [RabbitMQ](#) container image when run is run using `root` user, which is restricted in OpenShift by default, to enable the container to be run with `root` we need to give special permissions.

We create a separate service account called **brewery** and add the required SCC to it as shown below.

```
oc adm policy add-scc-to-user privileged -z brewery ①
```

```
oc adm policy add-scc-to-user anyuid -z brewery ①
```

① **brewery** service is created when RabbitMQ is deployed

More info on privileged access, refer [here](#)

3.3. Deploy Applications

Table 2. Application List

	Application	Folder	Remarks
	RabbitMQ	\$PROJECT_HOME/extends/rabbitmq	Message Broker - https://www.rabbitmq.com/
	common	\$PROJECT_HOME/common	Common shared library
	common-zipkin-stream	\$PROJECT_HOME/common-zipkin-stream	Common shared library for the projects that uses the Sleuth Zipkin Stream for tracing
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
	Ingredients	\$PROJECT_HOME/ingredients	
	Reporting	\$PROJECT_HOME/reporting	
	Brewing	\$PROJECT_HOME/brewing	
	Presenting	\$PROJECT_HOME/presenting	

3.3.1. Building

The Iteration II of the brewery application has migrated all the projects to [Apache Maven](#) based build, to build the application run the following command

```
./mvnw -N install ①  
./mvnw clean install ②
```

- ① This will install the brewery parent pom in local maven repository
- ② This will build the applications, if the minishift or OpenShift cluster is running, this will trigger `s2i` builds of the respective application as well

3.3.2. 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 :))



Ensure that all [Pre-Requisite](#) are done before starting deployment.

RabbitMQ

Local Deployment

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

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

External Cloud Deployment

Sometimes you might have access to docker socket typical case when deploying to external cloud, in those cases you can run the following set of commands,

```
./mvnw clean fabric8:resource  
oc apply -f target/classes/META-INF/fabric8/openshift.yml
```

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 fabric8:deploy
```

Zuul

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

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

Ingredients

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

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

Reporting

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

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

Brewing

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

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

Presenting

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

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

3.4. Acceptance Testing

The **\$PROJECT_HOME/acceptance-tests** holds the test cases for testing the application. To perform we need to 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,

```
./mvnw clean test
```

3.5. Deprecated Modules

As part of Iteration-II the following modules have been deprecated,

- Eureka
- Config Server
- common-zipkin
- common-zipkin-old
- zookeeper
- docker