RED HAT CONSULTING

Advanced Application Development on Openshift

Prepared for: Mitzicom

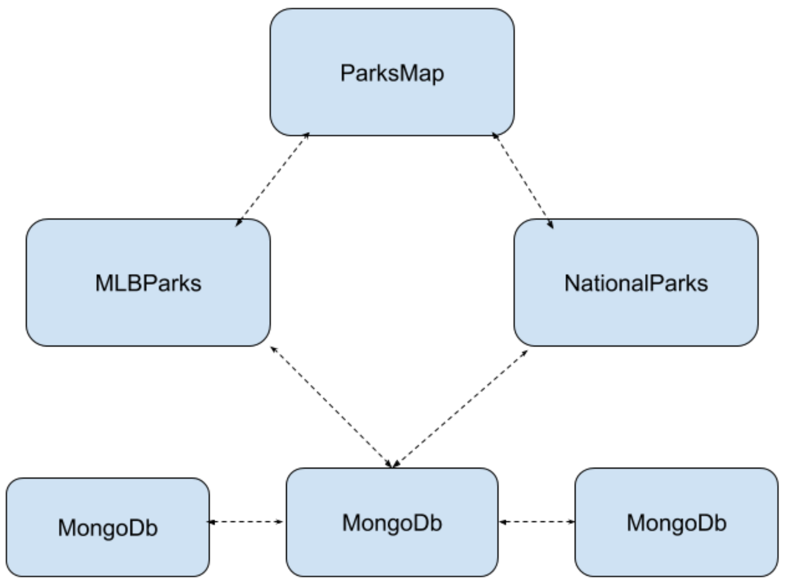
Colum Foskin
cfoskin@redhat.com
November 2018

TABLE OF CONTENT

Introduction

This is a Proof of Concept (POC) project using Red Hat OpenShift Container Platform (OCP) to determine the feasibility of using OpenShift as a target for an existing Java-based microservices workload. This project will demonstrate a fully integrated CI/CD pipeline using Nexus as the artifact repository and container registry and SonarQube for source code analysis. The microservices will be deployed to production in a blue-green strategy orchestrated by Jenkins. The application will consist of three microservices that will use a MongoDb replica set for storage, as shown in the following diagram.

Production Architecture



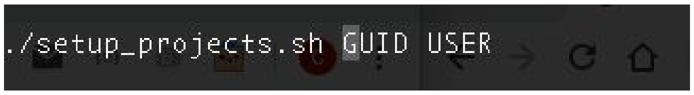
OpenShift Projects Overview

This POC will consist of five Projects deployed on OpenShift. The following section will detail what these will be and what purpose they serve. The projects will consist of most of OpenShifts key components or Objects such as:

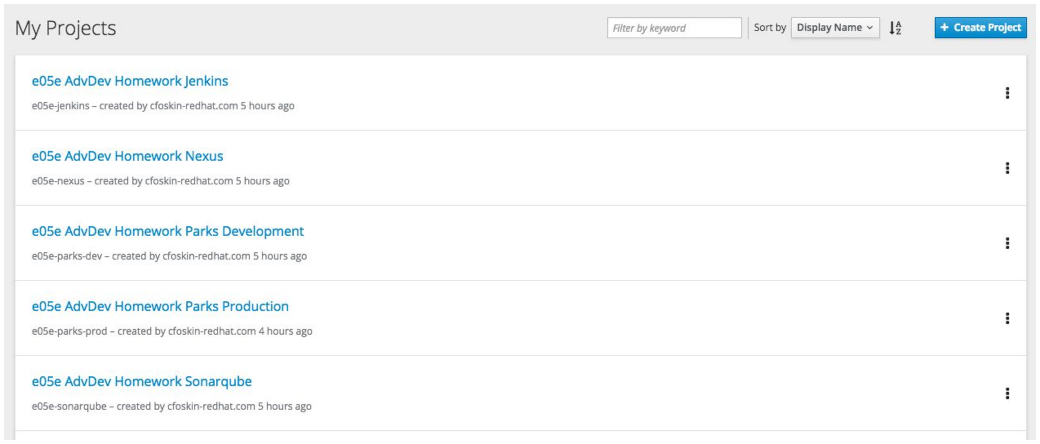
- Pods
- Services
- Routes
- Deployments/DeploymentConfig
- ConfigMaps
- Secrets
- StatefulSets
- ImageStreams
- Pipelines
- Builds
- PersistentVolumeClaims

Project Creation

A script called ***create_projects.sh*** is provided to create the required projects. The following parameters are required when running this script: **GUID , USER.**



```
./setup_projects.sh GUID USER
```



My Projects

e05e AdvDev Homework Jenkins
e05e-jenkins – created by cfoskin-redhat.com 5 hours ago

e05e AdvDev Homework Nexus
e05e-nexus – created by cfoskin-redhat.com 5 hours ago

e05e AdvDev Homework Parks Development
e05e-parks-dev – created by cfoskin-redhat.com 5 hours ago

e05e AdvDev Homework Parks Production
e05e-parks-prod – created by cfoskin-redhat.com 4 hours ago

e05e AdvDev Homework Sonarqube
e05e-sonarqube – created by cfoskin-redhat.com 5 hours ago

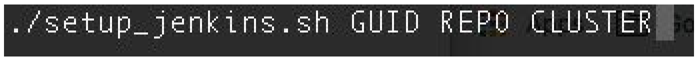
Project Deployments

Continuous Integration & Continuous Deployment (CI/CD) Projects - Infrastructure

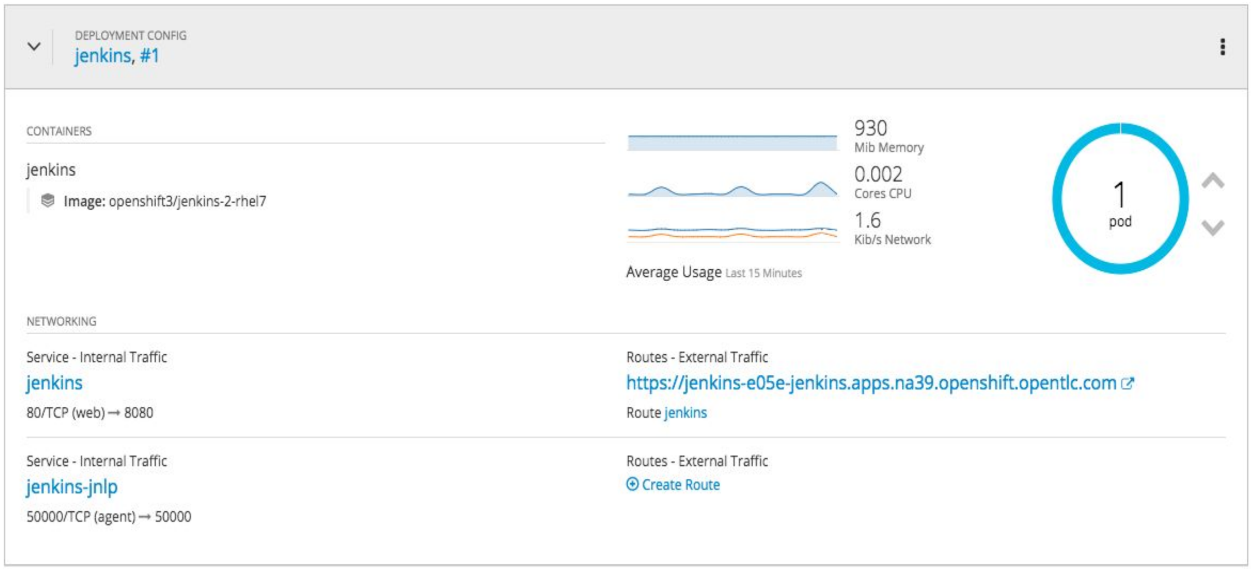
Jenkins

The Jenkins project consists of a Jenkins CI server to manage builds for each microservice. The Jenkins server uses a Persistent Volume for persistent storage.The Jenkins master will orchestrate builds to happen on a slave pod and will not perform the builds itself. The slave pods contain all of the required software to perform the Jenkins CI/CD pipeline stages.

A script called ***setup_jenkins.sh*** is provided to setup the jenkins project. The following parameters are required when running this script: **GUID , REPO & CLUSTER.**



```
./setup_jenkins.sh GUID REPO CLUSTER
```



DEPLOYMENT CONFIG
jenkins, #1

CONTAINERS
jenkins
Image: openshift3/jenkins-2-rhel7

930 Mib Memory
0.002 Cores CPU
1.6 Kib/s Network
Average Usage Last 15 Minutes

1 pod

NETWORKING

Service - Internal Traffic
jenkins
80/TCP (web) → 8080

Routes - External Traffic
https://jenkins-e05e-jenkins.apps.na39.openshift.opentlc.com
Route jenkins

Service - Internal Traffic
jenkins-jnlp
50000/TCP (agent) → 50000

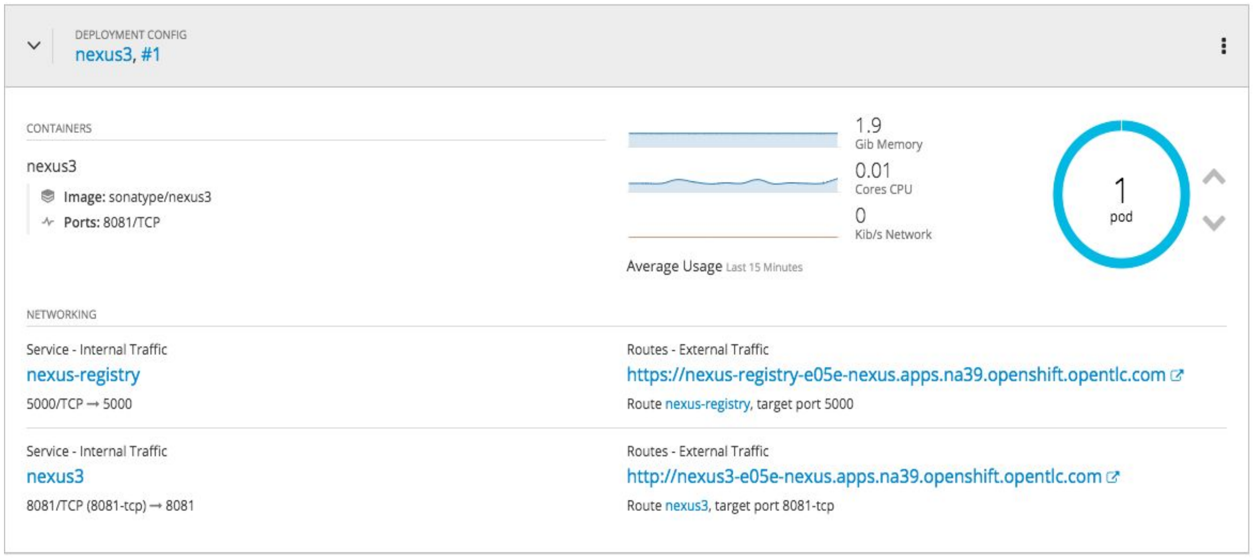
Routes - External Traffic
Create Route

Nexus

The Nexus project will consist of a Nexus 3 server which will be used to cache Red Hat and other build artifacts and it will also be used as a Docker registry. Nexus 3 will be configured to use a Persistent Volume for persistent storage.

A script called ***setup_nexus.sh*** is provided to setup the Nexus project. The following parameters are required when running this script: **GUID.**

```
./setup_nexus.sh GUID
```



Service - Internal Traffic
nexus-registry
5000/TCP → 5000

Routes - External Traffic
https://nexus-registry-e05e-nexus.apps.na39.openshift.opentlc.com
Route nexus-registry, target port 5000

Service - Internal Traffic
nexus3
8081/TCP (8081-tcp) → 8081

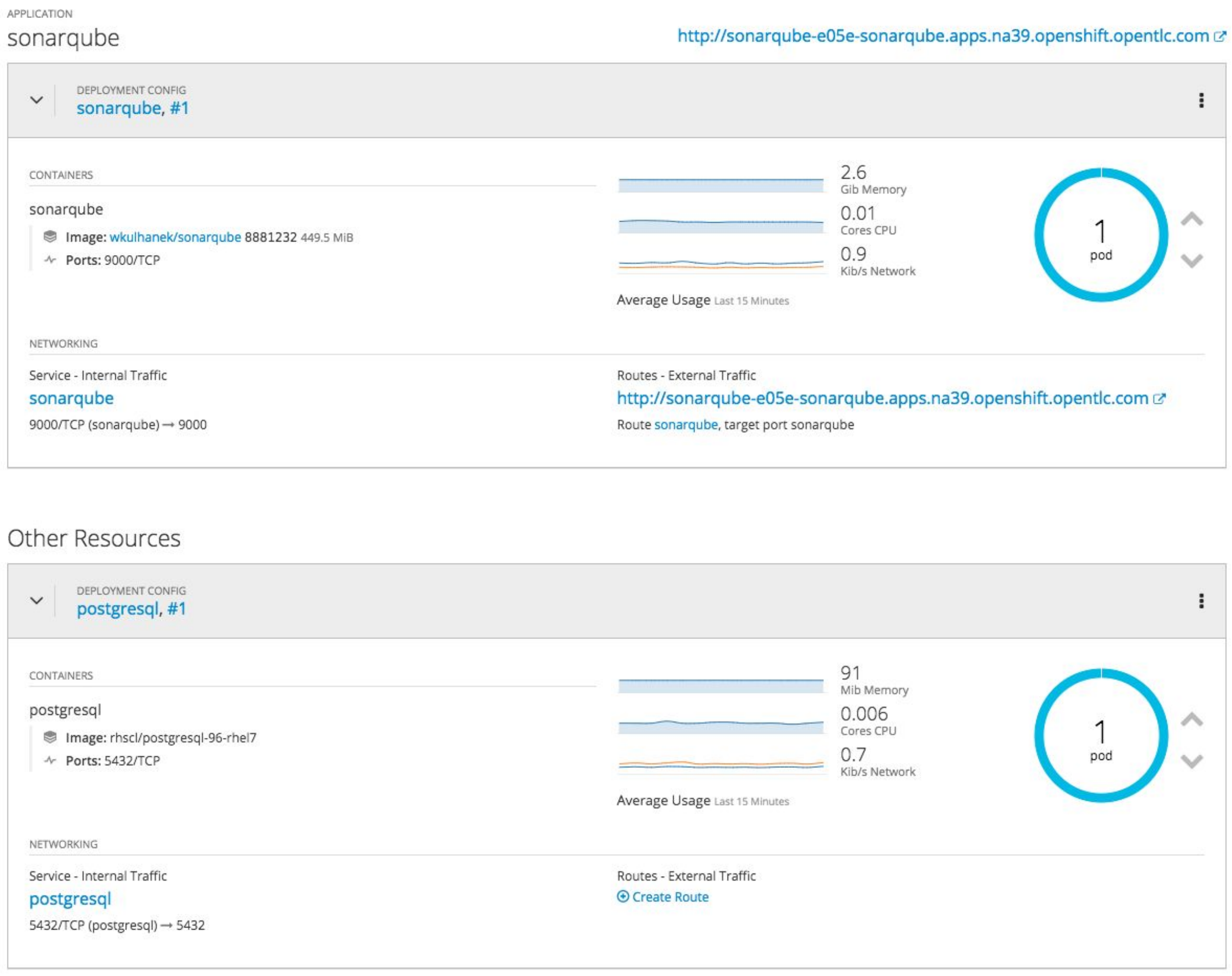
Routes - External Traffic
http://nexus3-e05e-nexus.apps.na39.openshift.opentlc.com
Route nexus3, target port 8081-tcp

SonarQube

SonarQube will be used for analysing the code of each microservice and this will be deployed with a Postgresql instance which will be backed using a Persistent Volume for persistent storage.

A script called ***setup_sonar.sh*** is provided to create the SonarQube project. The following parameters are required when running this script: **GUID.**

```
./setup_sonar.sh GUID
```



APPLICATION
sonarqube
http://sonarqube-e05e-sonarqube.apps.na39.openshift.opentlc.com

DEPLOYMENT CONFIG
sonarqube, #1

CONTAINERS
sonarqube
Image: wkulhanek/sonarqube 8881232 449.5 MiB
Ports: 9000/TCP

2.6 Gib Memory
0.01 Cores CPU
0.9 Kib/s Network
Average Usage Last 15 Minutes

1 pod

NETWORKING
Service - Internal Traffic
sonarqube
9000/TCP (sonarqube) → 9000

Routes - External Traffic
http://sonarqube-e05e-sonarqube.apps.na39.openshift.opentlc.com
Route sonarqube, target port sonarqube

Other Resources

DEPLOYMENT CONFIG
postgresql, #1

CONTAINERS
postgresql
Image: rhscl/postgresql-96-rhel7
Ports: 5432/TCP

91 Mib Memory
0.006 Cores CPU
0.7 Kib/s Network
Average Usage Last 15 Minutes

1 pod

NETWORKING
Service - Internal Traffic
postgresql
5432/TCP (postgresql) → 5432

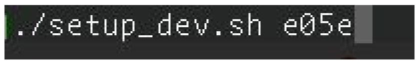
Routes - External Traffic
Create Route

Production & Development Projects

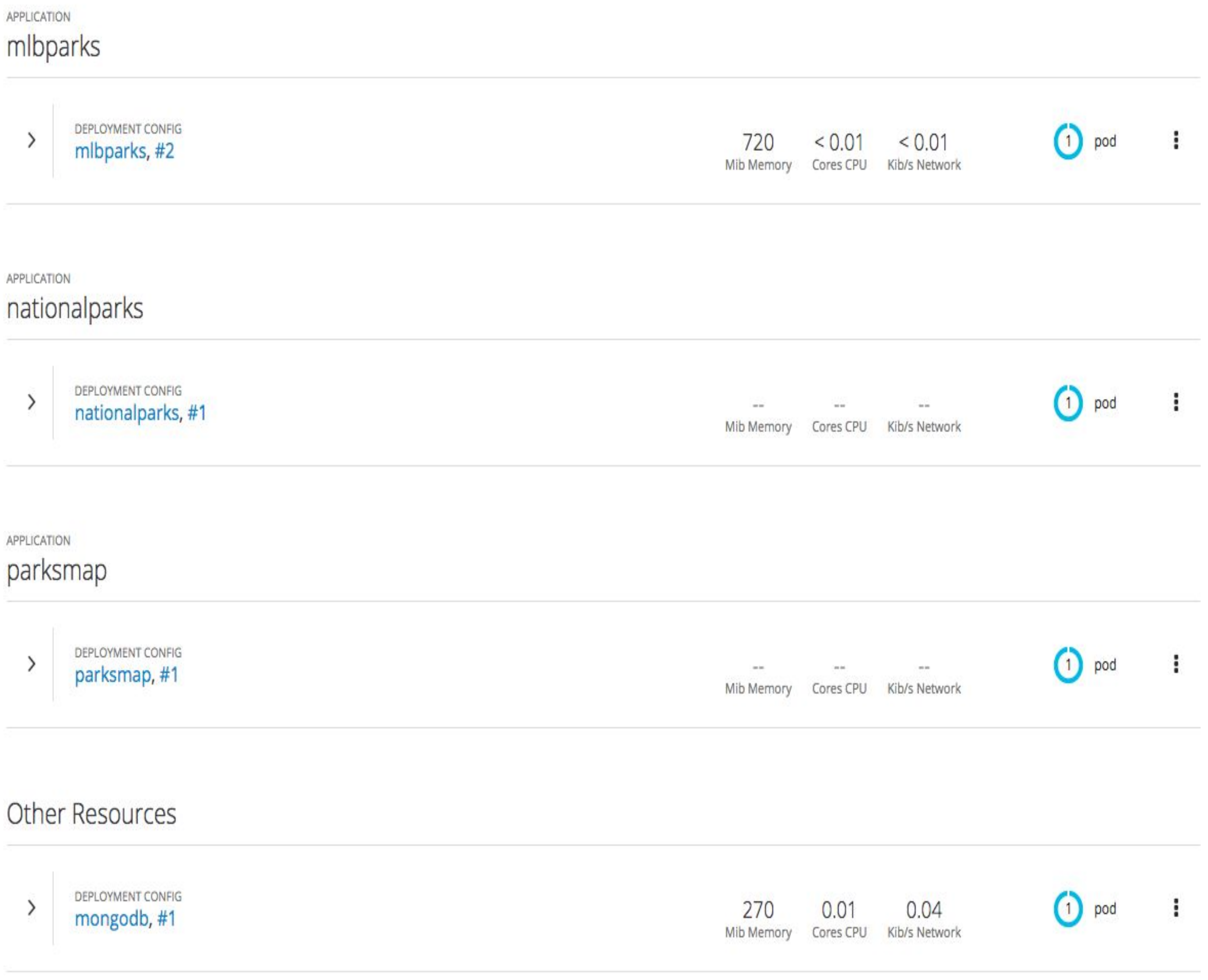
Development Project

The development project will be be used as a test environment for testing the microservices to conserve resources - instead of a separate “Test” project. This would also be the project that the developers would use for new features on the applications. This project will have a single persistent MongoDB pod for the microservices application.

A script called ***setup_dev.sh*** is provided to setup the Development project. The following parameters are required when running this script: **GUID.**



```
./setup_dev.sh e05e
```



APPLICATION
mlbparks

DEPLOYMENT CONFIG mlbparks, #2	720 Mib Memory	< 0.01 Cores CPU	< 0.01 Kib/s Network	1 pod

APPLICATION
nationalparks

DEPLOYMENT CONFIG nationalparks, #1	-- Mib Memory	-- Cores CPU	-- Kib/s Network	1 pod

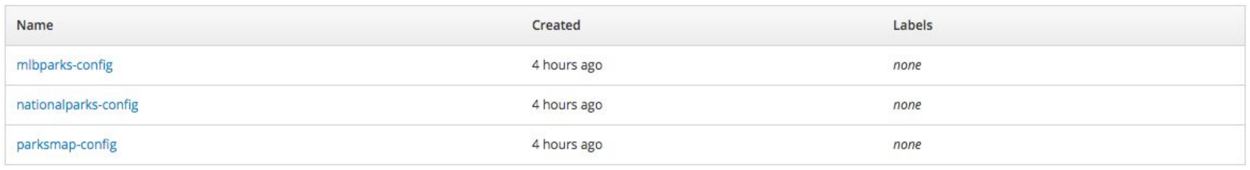
APPLICATION
parksmap

DEPLOYMENT CONFIG parksmap, #1	-- Mib Memory	-- Cores CPU	-- Kib/s Network	1 pod

Other Resources

DEPLOYMENT CONFIG mongodb, #1	270 Mib Memory	0.01 Cores CPU	0.04 Kib/s Network	1 pod

Each microservice will be configured using a ConfigMap.



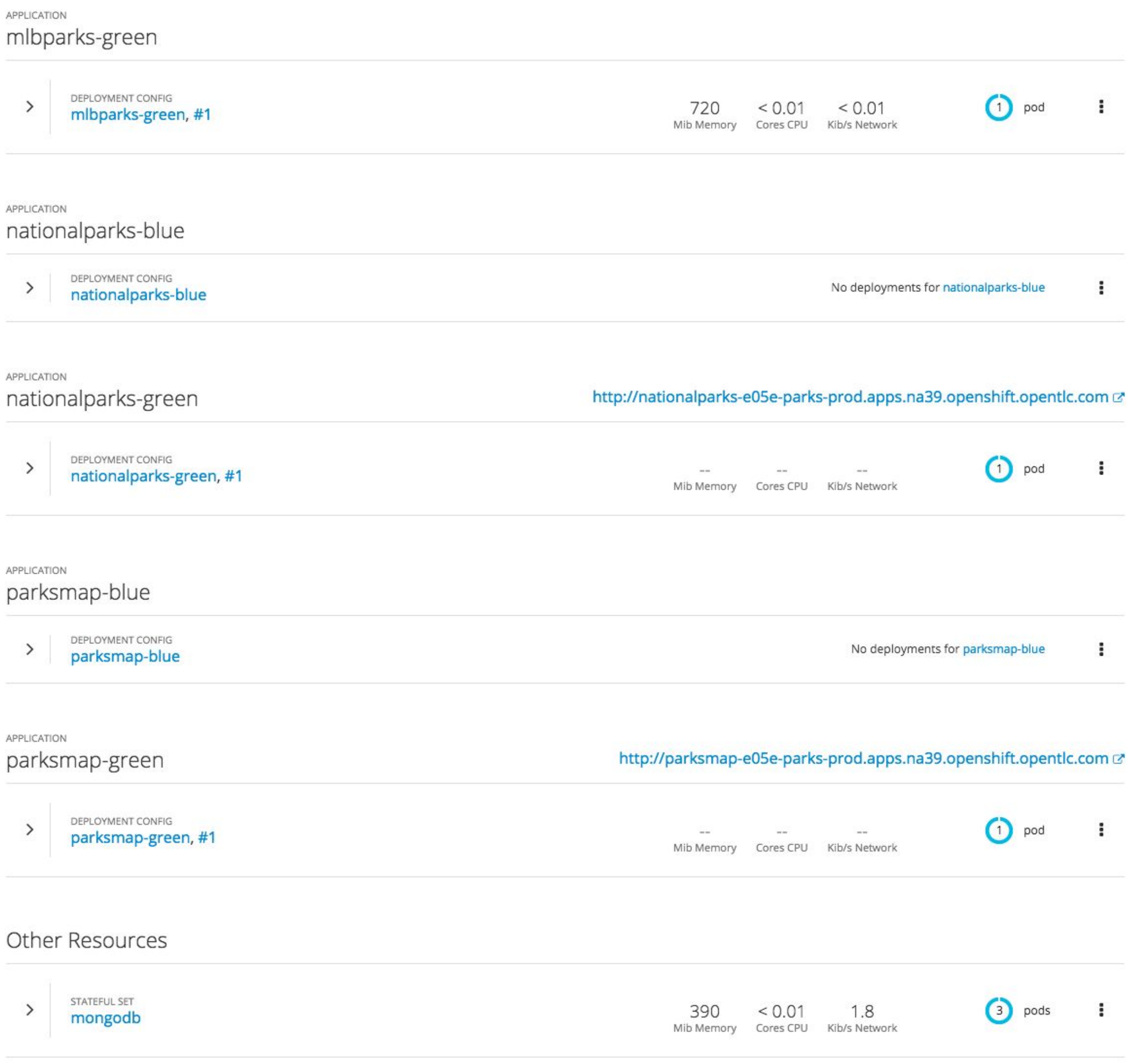
Name	Created	Labels
mlbparks-config	4 hours ago	*none*
nationalparks-config	4 hours ago	*none*
parksmap-config	4 hours ago	*none*

Production Project

This project will consist of both blue & green versions of the microservices which will allow for blue-green deployment strategy. The microservices that run here will have come through the CI/CD pipelines and deemed to be ready for production.

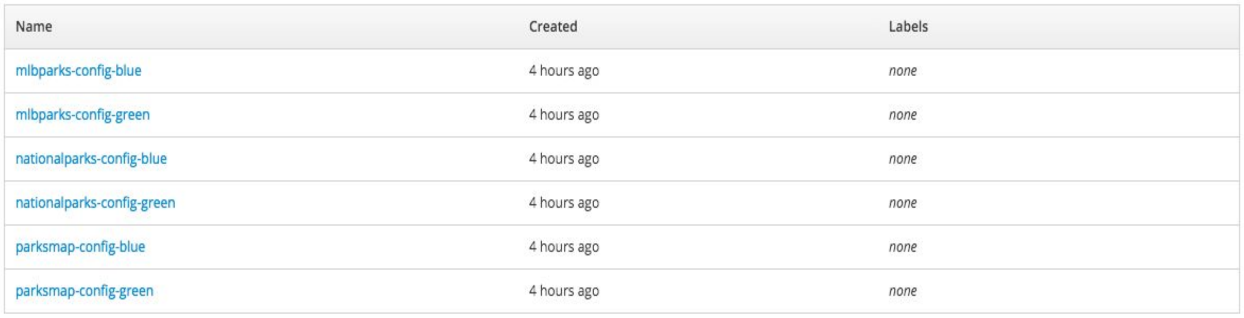
A script called ***setup_prod.sh*** is provided to setup the Production project. The following parameters are required when running this script: **GUID.**

```
./setup_prod.sh GUID
```



```
APPLICATION
mlbparks-green
  DEPLOYMENT CONFIG
  mlbparks-green, #1        720 Mib Memory   < 0.01 Cores CPU   < 0.01 Kib/s Network   1 pod

APPLICATION
nationalparks-blue
  DEPLOYMENT CONFIG
  nationalparks-blue        No deployments for nationalparks-blue

APPLICATION
nationalparks-green         http://nationalparks-e05e-parks-prod.apps.na39.openshift.opentlc.com
  DEPLOYMENT CONFIG
  nationalparks-green, #1   -- Mib Memory   -- Cores CPU   -- Kib/s Network   1 pod

APPLICATION
parksmap-blue
  DEPLOYMENT CONFIG
  parksmap-blue             No deployments for parksmap-blue

APPLICATION
parksmap-green              http://parksmap-e05e-parks-prod.apps.na39.openshift.opentlc.com
  DEPLOYMENT CONFIG
  parksmap-green, #1        -- Mib Memory   -- Cores CPU   -- Kib/s Network   1 pod

Other Resources
  STATEFUL SET
  mongodb                   390 Mib Memory   < 0.01 Cores CPU   1.8 Kib/s Network   3 pods
```

Each of the microservices are configured using ConfigMaps.



Name	Created	Labels
mlbparks-config-blue	4 hours ago	*none*
mlbparks-config-green	4 hours ago	*none*
nationalparks-config-blue	4 hours ago	*none*
nationalparks-config-green	4 hours ago	*none*
parksmap-config-blue	4 hours ago	*none*
parksmap-config-green	4 hours ago	*none*

CI/CD Pipelines

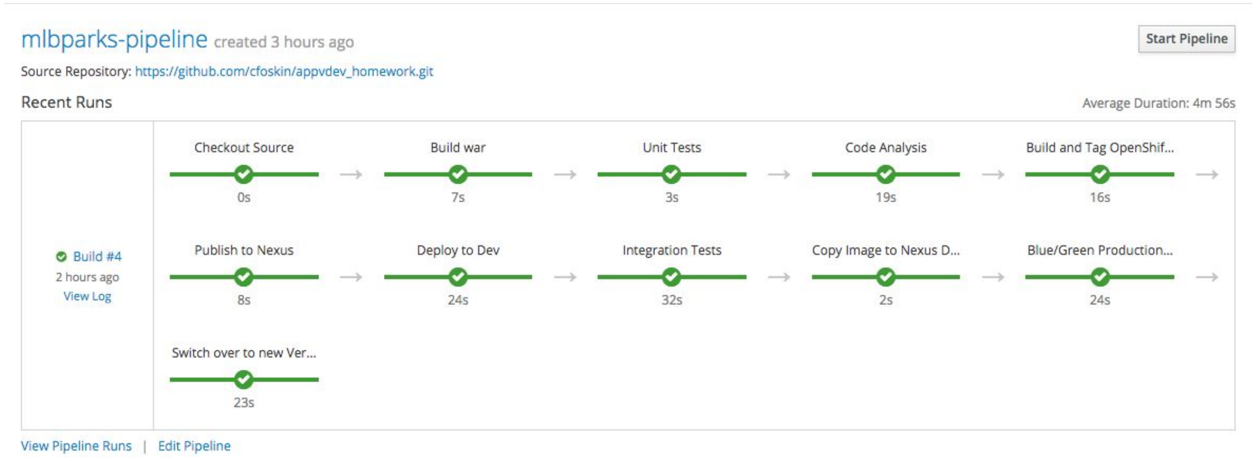
Each microservice application will have its own automated CI/CD Jenkins pipeline.The CI/CD pipeline will consist of the following stages:

- Check out the source code of the project.
- Build the war/jar file using Maven.
- Run unit tests using Maven.
- Perform a code analysis using SonarQube.
- Build the Openshift image & tag it.
- Publish the Maven artifact to the Nexus repository.
- Deploy the application to the Openshift Development project.
- Run a set of integration tests to ensure successful deployment.
- Copy the built image to the Nexus docker registry.
- Deploy the application to the Openshift Production project using the blue-green deployment strategy.
- Complete the switch over to the new version of the application after confirmation from the user.

The Pipelines are defined in a Jenkinsfile that is present in each microservice sub-directory of the application project. They are automatically created in Jenkins by using an Openshift BuildConfig with the JenkinsPipeline build strategy when the ***setup_jenkins.sh*** is ran to configure the Jenkins project.

The three pipelines and each stage that is executed can be seen in the following screenshots.

MLBParks Pipeline



mlbparks-pipeline created 3 hours ago

Source Repository: https://github.com/cfoskin/appvdev_homework.git

Recent Runs

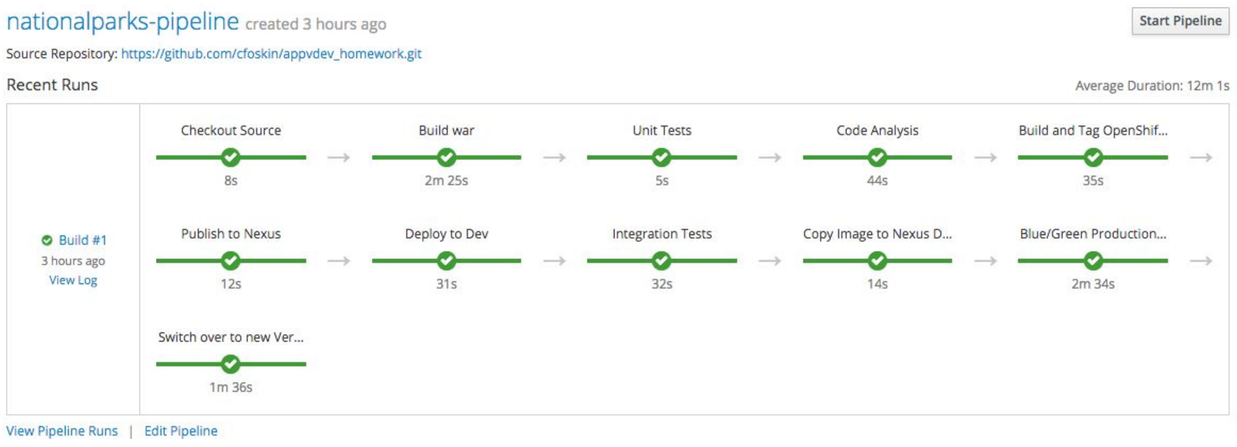
Average Duration: 4m 56s

Build #4
2 hours ago
View Log

Stage	Duration
Checkout Source	0s
Build war	7s
Unit Tests	3s
Code Analysis	19s
Build and Tag OpenShif...	16s
Publish to Nexus	8s
Deploy to Dev	24s
Integration Tests	32s
Copy Image to Nexus D...	2s
Blue/Green Production...	24s
Switch over to new Ver...	23s

View Pipeline Runs | Edit Pipeline

NationalParks Pipeline



nationalparks-pipeline created 3 hours ago

Source Repository: https://github.com/cfoskin/appvdev_homework.git

Recent Runs

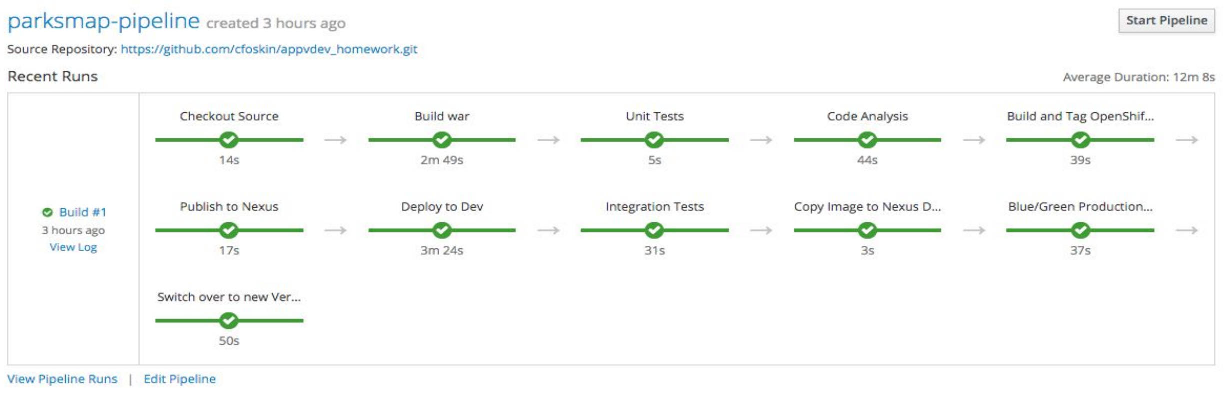
Average Duration: 12m 1s

Build #1
3 hours ago
View Log

Stage	Duration
Checkout Source	8s
Build war	2m 25s
Unit Tests	5s
Code Analysis	44s
Build and Tag OpenShif...	35s
Publish to Nexus	12s
Deploy to Dev	31s
Integration Tests	32s
Copy Image to Nexus D...	14s
Blue/Green Production...	2m 34s
Switch over to new Ver...	1m 36s

View Pipeline Runs | Edit Pipeline

ParksMap Pipeline



parksmap-pipeline created 3 hours ago

Source Repository: https://github.com/cfoskin/appvdev_homework.git

Recent Runs

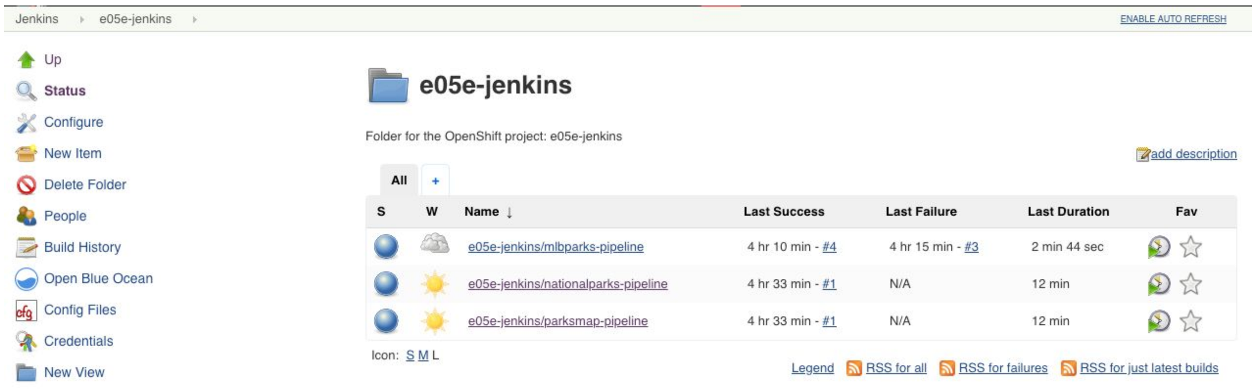
Average Duration: 12m 8s

Build #1
3 hours ago
View Log

Stage	Duration
Checkout Source	14s
Build war	2m 49s
Unit Tests	5s
Code Analysis	44s
Build and Tag OpenShif...	39s
Publish to Nexus	17s
Deploy to Dev	3m 24s
Integration Tests	31s
Copy Image to Nexus D...	3s
Blue/Green Production...	37s
Switch over to new Ver...	50s

View Pipeline Runs | Edit Pipeline

The pipelines can also be viewed or managed via the Jenkins console.



Folder for the OpenShift project: e05e-jenkins

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
●	☁	e05e-jenkins/mlbparks-pipeline	4 hr 10 min - #4	4 hr 15 min - #3	2 min 44 sec	☆
●	☀	e05e-jenkins/nationalparks-pipeline	4 hr 33 min - #1	N/A	12 min	☆
●	☀	e05e-jenkins/parksmap-pipeline	4 hr 33 min - #1	N/A	12 min	☆

Project Source Code

The source code for this project can be found in a GitHub repository [1]. This includes all of the scripts and Openshift templates needed to setup and configure each project. It also includes the code for each microservice where each one has its own Jenkinsfile with the pipeline definition.

[1] https://github.com/cfoskin/appvdev_homework