**MitziCom**

**Shawn K. Patterson**



**Executive Summary**

This Engagement Journal go over the step-by-step instructions for a successful implementation of Red Hat OpenShift 3.11 along with a CI/CD pipeline.

**Requirements**

Requirements included creating a POC to demonstrate the advantages of using OpenShift to manage AppDev deployments with an CI/CD pipeline that included Gogs, Jenkins and Sonarqube. A simple NodeJS deployment was tested before the CI/CD pipeline rollout.

**NodeJS Deployment**

**Commands**

Create nodesjs application

oc new-projet nodejs-mongo-persistent

oc new-app nodejs-mongo-persistent

oc expose nodejs-mongo-persistent

[root@bastion ~]# oc get route

NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

nodejs-mongo-persistent nodejs-mongo-persistent-nodejs-mongo-persistent.apps.c54c.example.opentlc.com nodejs-mongo-persistent <all> None

[root@bastion ~]# oc get pods

NAME READY STATUS RESTARTS AGE

mongodb-1-mgs97 1/1 Running 5 7d

nodejs-mongo-persistent-1-build 0/1 Completed 0 7d

nodejs-mongo-persistent-1-dqrtk 1/1 Running 4 7d

**CI/CD Pipeline**

**Nexus Deployment**

* Created Project cicd-nexus
* Deployed sonatype/nexus3:latest from Dockerhub
* Exposed Nexus service
* Paused Nexus application for patching
* Adjusted resource requirements and attached Persistent Volume (PV) to /nexus-data
* Added readiness probes
* Ran setup\_nexus.sh script to auto-create repositories for application buildout.

**Commands:**

oc new-project cicd-nexus --display-name "Shared Nexus"

oc new-app sonatype/nexus3:latest

oc expose svc nexus3

oc rollout pause dc nexus3

oc patch dc nexus3 --patch='{ "spec": { "strategy": { "type": "Recreate" }}}'

oc set resources dc nexus3 --limits=memory=2Gi,cpu=2 --requests=memory=1Gi,cpu=500m

# Create persistent volume for mount

echo "apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nexus-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 4Gi" | oc create -f -

oc set volume dc/nexus3 --add --overwrite --name=nexus3-volume-1 --mount-path=/nexus-data/ --type persistentVolumeClaim --claim-name=nexus-pvc

oc set probe dc/nexus3 --liveness --failure-threshold 3 --initial-delay-seconds 60 -- echo ok

oc set probe dc/nexus3 --readiness --failure-threshold 3 --initial-delay-seconds 60 --get-url=http://:8081/

oc rollout resume dc nexus3

# Commands run on bastion host

curl -o setup\_nexus3.sh -s https://raw.githubusercontent.com/redhat-gpte-devopsautomation/ocp\_advanced\_development\_resources/master/nexus/setup\_nexus3.sh

chmod +x setup\_nexus3.sh

./setup\_nexus3.sh admin admin123 http://$(oc get route nexus3 --template='{{ .spec.host }}')

rm setup\_nexus3.sh

oc expose dc nexus3 --port=5000 --name=nexus-registry

oc create route edge nexus-registry --service=nexus-registry --port=5000

oc annotate route nexus3 console.alpha.openshift.io/overview-app-route=true

oc annotate route nexus-registry console.alpha.openshift.io/overview-app-route=false

**Sonarqube Deployment**

* Created a project called cicd-sonarqube
* Created a PostgreSQL database for the Sonarqube backend
* Deployed wkulhanek/sonarqueb”6.7.5 from Docker Repository
* Paused deployment to add persistent volume to mount to /opt/sonarqube/data
* Adjusted resource limits, setup readiness probes and resumed deployment

**Commands**

oc new-project cicd-sonarqube --display-name "Shared Sonarqube"

oc new-app --template=postgresql-persistent --param POSTGRESQL\_USER=sonar --param POSTGRESQL\_PASSWORD=sonar --param POSTGRESQL\_DATABASE=sonar --param VOLUME\_CAPACITY=4Gi --labels=app=sonarqube\_db

oc new-app --docker-image=wkulhanek/sonarqube:6.7.5 --env=SONARQUBE\_JDBC\_USERNAME=sonar --env=SONARQUBE\_JDBC\_PASSWORD=sonar --env=SONARQUBE\_JDBC\_URL=jdbc:postgresql://postgresql/sonar --labels=app=sonarqube

oc rollout pause dc sonarqube

oc expose service sonarqube

# Create persistent volume for mount

echo "apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: sonarqube-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 4Gi" | oc create -f -

oc set volume dc/sonarqube --add --overwrite --name=sonarqube-volume-1 --mount-path=/opt/sonarqube/data/ --type persistentVolumeClaim --claim-name=sonarqube-pvc

oc set resources dc/sonarqube --limits=memory=3Gi,cpu=2 --requests=memory=2Gi,cpu=1

oc patch dc sonarqube --patch='{ "spec": { "strategy": { "type": "Recreate" }}}'

oc set probe dc/sonarqube --liveness --failure-threshold 3 --initial-delay-seconds 40 -- echo ok

oc set probe dc/sonarqube --readiness --failure-threshold 3 --initial-delay-seconds 20 --get-url=http://:9000/about

oc rollout resume dc sonarqube

**Gogs Deployment**

* Created project named cicd-gogs
* Created a persistent PostgreSQL Database for the Gogs backend
* Deployed wkulhanek/gogs:11.35 from Docker Repository
* Paused deployment to add persistent volume to mount to /data
* Adjusted resource limits, setup readiness probes and resumed deployment
* Cloned the Parksmap repo to my local machine and then changed the nexus\_settings.xml file to point to the Nexus repo running on the Openshift cluster.
* Configured my local repo to link with the remote repo on Gogs and then pushed local repo to the remote Gogs repo.

**Commands**

oc new-project cicd-gogs --display-name "Shared Gogs"

oc new-app postgresql-persistent --param POSTGRESQL\_DATABASE=gogs --param POSTGRESQL\_USER=gogs --param POSTGRESQL\_PASSWORD=gogs --param VOLUME\_CAPACITY=4Gi -lapp=postgresql\_gogs

oc new-app wkulhanek/gogs:11.34 -lapp=gogs

# Create persistent volume for mount

echo "apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: gogs-data

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 4Gi" | oc create -f -

oc set volume dc/gogs --add --overwrite --name=gogs-volume-1 --mount-path=/data/ --type persistentVolumeClaim --claim-name=gogs-data

oc expose svc gogs

oc get route gogs

oc exec $(oc get pod | grep "^gogs" | awk '{print $1}') -- cat /opt/gogs/custom/conf/app.ini >$HOME/app.ini

oc create configmap gogs --from-file=/etc/ansible/app.ini

oc set volume dc/gogs --add --overwrite --name=config-volume -m /opt/gogs/custom/conf/ -t configmap --configmap-name=gogs

cd $HOME

git clone https://github.com/redhat-gpte-devopsautomation/openshift-tasks.git

cd $HOME/openshift-tasks

git remote add gogs http://user1:user1@$(oc get route gogs -n cicd-gogs --template='{{ .spec.host }}')/CICDLabs/openshift-tasks.git

git push -u gogs master

<?xml version="1.0"?>

<settings>

<mirrors>

<mirror>

<id>Nexus</id>

<name>Nexus Public Mirror</name>

<url>http://nexus3.cicd-nexus.svc.cluster.local:8081/repository/maven-all-public/</url>

<mirrorOf>\*</mirrorOf>

</mirror>

</mirrors>

<servers>

<server>

<id>nexus</id>

<username>admin</username>

<password>admin123</password>

</server>

</servers>

</settings>

git commit -m "Updated Settings" nexus\_settings.xml nexus\_openshift\_settings.xml

git push gogs master

**Jenkins Deployment**

* Created project named cicd-dev
* Deployed jenkins-persistent from Docker Repository with memory and volume limits
* Created jenkins workflow
* Adjusted resource limits, setup readiness probes and resumed deployment

**Commands**

oc new-project cicd-dev --display-name "Jenkins Workflow”

oc new-app jenkins-persistent --param ENABLE\_OAUTH=true --param MEMORY\_LIMIT=2Gi --param VOLUME\_CAPACITY=4Gi --param DISABLE\_ADMINISTRATIVE\_MONITORS=true

oc new-build -D $'FROM docker.io/openshift/jenkins-agent-maven-35-centos7:v3.11\n

USER root\nRUN yum -y install skopeo && yum clean all\n

USER 1001' --name=jenkins-agent-appdev -n skp-jenkins

skopeo copy --dest-tls-verify=false --dest-creds=admin:admin123 docker://docker-registry-default.apps.c54c.example.opentlc.com/rk-jenkins/jenkins-slave-maven-appde v:v3.11 docker://$(oc get route nexus-registry -n rk-nexus --template='{{ .spec.host }}')/rk-jenkins/jenkins-slave-maven-appdev:v3.11

**Development / Test / Production Environment Setup**

- Created projects called tasks-dev, tasks-test and tasks-prod

- Created a MongoDB deployment config

- Gave the Jenkins project permission to manipulate objects all three projects

- Created a build configuration

- Created a deployment config, configmap and set environment variables for each of the three environments

- Removed all triggers

- Created services for each of the deployment configs

- Exposed each service as a route.

- Created readiness and liveness probes for each deployment.

**Commands**

**Development Environment**

oc new-project tasks-dev --display-name "mitzicom-dev”

oc policy add-role-to-user edit system:serviceaccount:cicd-jenkins:jenkins -n tasks-dev

oc new-build --binary=true --name="task-build" jboss-eap70-openshift:1.6 -n

oc new-app openshift-task/task-build:0.0-0 --name=mitzicom --allow-missing-imagestream-tags=true -l type=mitzicom-backend -n tasks-dev

oc create configmap mitzicom-config \

--from-literal=DB\_HOST=mongodb\

--from-literal=DB\_PORT=27017\

--from-literal=DB\_PORT=27017 \

--from-literal=DB\_USERNAME=dev \

--from-literal=DB\_PASSWORD=dev \

--from-literal=DB\_NAME=dev \

--from-literal=APPNAME="mitzicom" \

-n cicd-dev

oc set env --from=configmap/mitzicom-config dc/mitzicom

oc set triggers dc/mitzicom --remove-all

oc expose dc/mitzicom --port 8080

oc expose svc/mitzicom

oc set probe dc/mitzicom --readiness --get-url=http://:8080/ws/healthz/ --failure-threshold=3 --initial-delay-seconds=60

oc set probe dc/mitzicom --liveness --failure-threshold 3 --initial-delay-seconds 60 -- echo ok

**Test Environment**

oc new-project tasks-test --display-name "mitzicom-test”

oc policy add-role-to-user edit system:serviceaccount:cicd-jenkins:jenkins -n tasks-test

oc new-build --binary=true --name="task-build" jboss-eap70-openshift:1.6 -n

oc new-app openshift-task/task-build:0.0-0 --name=mitzicom --allow-missing-imagestream-tags=true -l type=mitzicom-backend -n tasks-test

oc create configmap mitzicom-config \

--from-literal=DB\_HOST=mongodb\

--from-literal=DB\_PORT=27017\

--from-literal=DB\_PORT=27017 \

--from-literal=DB\_USERNAME=test \

--from-literal=DB\_PASSWORD=test \

--from-literal=DB\_NAME=test \

--from-literal=APPNAME="mitzicom" \

-n tasks-test

oc set env --from=configmap/mitzicom-config dc/mitzicom

oc set triggers dc/mitzicom --remove-all

oc expose dc/mitzicom --port 8080

oc expose svc/mitzicom

oc set probe dc/mitzicom --readiness --get-url=http://:8080/ws/healthz/ --failure-threshold=3 --initial-delay-seconds=60

oc set probe dc/mitzicom --liveness --failure-threshold 3 --initial-delay-seconds 60 -- echo ok

**Production Environment**

oc policy add-role-to-user edit system:serviceaccount:cicd-jenkins:jenkins -n tasks-prod

oc new-build --binary=true --name="task-build" jboss-eap70-openshift:1.6 -n

oc new-app openshift-task/task-build:0.0-0 --name=mitzicom --allow-missing-imagestream-tags=true -l type=mitzicom-backend -n tasks-prod

oc create configmap mitzicom-config \

--from-literal=DB\_HOST=mongodb\

--from-literal=DB\_PORT=27017\

--from-literal=DB\_PORT=27017 \

--from-literal=DB\_USERNAME=prod \

--from-literal=DB\_PASSWORD=prod \

--from-literal=DB\_NAME=prod \

--from-literal=APPNAME="mitzicom" \

-n tasks-prod

oc set env --from=configmap/mitzicom-config dc/mitzicom

oc set triggers dc/mitzicom --remove-all

oc expose dc/mitzicom --port 8080

oc expose svc/mitzicom

oc set probe dc/mitzicom --readiness --get-url=http://:8080/ws/healthz/ --failure-threshold=3 --initial-delay-seconds=60

oc set probe dc/mitzicom --liveness --failure-threshold 3 --initial-delay-seconds 60 -- echo ok