OCP HA DEPLOYEMENT

Prerequisites:

1. Make sure the systems to be part of the OCP cluster are on network and with RHEL OS installed, registered to RHSM/Satellite and configured with FQDN hostnames.

In my example, hostnames are:

```
loadbalancer1.00d0.internal
master1.18a3.internal
master2.18a3.internal
master3.18a3.internal
infranode1.18a3.internal
infranode2.18a3.internal
node1.18a3.internal
node2.18a3.internal
node3.18a3.internal
node4.18a3.internal
support1.00d0.internal  <==NFS Volumes
```

2. From your laptop / bastion machine, run the following command to connect to github and download all the configuration playbooks and scripts to setup OCP HA Cluster.

```
# git clone https://github.com/jaideena/ocp-homework
```

3. Copy the file by name “inventory_host” to */etc/ansible/host*

4. Run the following commands to prepare the hosts:

```
# ansible localhost,all -m shell -a 'export GUID=`hostname | cut -d"." -f2`; echo "export GUID=$GUID" >> $HOME/.bashrc'

# ansible all -m ping

# ansible nodes -m shell -a"systemctl status docker | grep Active"

# ansible nodes -m shell -a"docker version|grep Version"

# ansible all -m shell -a"yum repolist"

# ansible nfs -m shell -a"exportfs"

# yum -y install atomic-openshift-utils atomic-openshift-clients
```

Installation of OCP in HA mode

Basic Requirements

- Ability to authenticate at the master console
- Registry has storage attached and working
- Router is configured on each infranode
- PVs of different types are available for users to consume
- Ability to deploy a simple app (nodejs-mongo-persistent)
-

HA Requirements

- There are three masters working
- There are three etcd instances working
- There is a load balancer to access the masters called loadbalancer.$GUID.$DOMAIN
- There is a load balancer/DNS for both infranodes called *.apps.$GUID.$DOMAIN
- There are at least two infranodes, labeled env=infra

Environment Configuration

- NetworkPolicy is configured and working with projects isolated by default
- Aggregated logging is configured and working
- Metrics collection is configured and working
- Router and Registry Pods run on Infranodes
- Metrics and Logging components run on Infranodes
- Service Catalog, Template Service Broker, and Ansible Service Broker are all working

1. Run the OCP prerequisites playbook

```
# ansible-playbook -f 20 /usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml
```

2. Run the OCP deployment playbook

```
# ansible-playbook -f 20 /usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml
```

3. Once the OCP HA deployment is successful, setp the nfs mount
- Login to **support1.00d0.internal** and switch to root.
- Execute the script 01_create_nfs_directories.sh that you find it in git repo locally.
- Exit and return to your laptop / bastion system
- Execute 02_pv_df_5GB_rwo.sh
- Execute 03_pv_df_10GB_rwm.sh

4. Perform smoke test by deploying an application with persistent storage – Use nodejs-mongo-persistent

```
# oc new-project smoke-test
```

# oc new-app nodejs-mongo-persistent

User creation

The following users are already created as a part of HA cluster.
- admin
- amy
- andrew
- betty
- Brian

The auth method used for this deployment is htpasswd. The password for all the user account is r3dh4t1!

1. User “admin” should be given cluster admin role

# oc adm policy add-role-to-user cluster-admin admin

Project Creation

Three projects (clients) were created by name “Alpha”, “Beta” and “Common” and each project are allocated a dedicated node to run their pods.

# oc adm new-project alpha --node-selector='hostname=node1.18a3.internal' --display-name='Alpha Corp'

# oc adm new-project beta --node-selector='hostname=node2.18a3.internal' --display-name='Beta Corp'

# oc adm new-project common --node-selector='hostname=node3.18a3.internal' --display-name='Common Customers'

# oc label node node1.18a3.internal client=alpha

# oc label node node2.18a3.internal client=beta

# oc label node node3.18a3.internal client=common

# oc label node node4.18a3.internal client=common

Assign roles to users to control projects

# oc adm policy add-role-to-user admin brian -n beta

# oc adm policy add-role-to-user admin betty -n beta

```
# oc adm policy add-role-to-user admin andrew -n alpha
```

```
# oc adm policy add-role-to-user admin amy -n alpha
```

CI / CD Pipeline

Requirement:

- Jenkins pod is running with a persistent volume
- Jenkins deploys openshift-tasks app
- Jenkins OpenShift plugin is used to create a CICD workflow
- HPA is configured and working on production deployment of openshift-tasks

1. Create three projects by name jdeenada-pipeline-tasks-dev, jdeenada-pipeline-tasks-test jdeenada-pipeline-tasks-prod for CI/CD ldeployment by lifecycle.

```
# oc new-project jdeenada-pipeline-tasks-dev

# oc new-project  jdeenada-pipeline-tasks-test

# oc new-project  jdeenada-pipeline-tasks-prod
```

2. Install Jenkins app (usrname: admin / PWD: openshiftpipelines)

```
# oc new-app jenkins-persistent -p ENABLE_OAUTH=false -e
JENKINS_PASSWORD=openshiftpipelines -n jdeenada-pipeline-tasks-dev

# oc policy add-role-to-user edit system:serviceaccount:jdeenada-pipeline-tasks-dev:jenkins -n
jdeenada-pipeline-tasks-test

# oc policy add-role-to-user edit system:serviceaccount:jdeenada-pipeline-tasks-dev:jenkins -n
jdeenada-pipeline-tasks-prod

# oc policy add-role-to-group system:image-puller system:serviceaccounts:jdeenada-pipeline-tasks-
test -n jdeenada-pipeline-tasks-dev

# oc policy add-role-to-group system:image-puller system:serviceaccounts:jdeenada-pipeline-tasks-
prod -n jdeenada-pipeline-tasks-dev
```

3. Jenkin pipeline by name “pipeline-demo” should be configured in project **j**deenada-pipeline-tasks-dev

4. Use playbook pipeline.yaml from gitrepo to create jenkin pipeline.

```
# oc create -f  pipeline.yaml -n jdeenada-pipeline-tasks-dev
```

Deploy application Openshift-tasks

1. The application is already build with S2I and the image is already available in image stream by name "openshift-tasks"

2. Create HPA
Use hpa.yaml to create horizontal auto scaler

```
# oc create -f hpa.yaml -n  jdeenada-pipeline-tasks-dev
```

3. If you want set limit to projects. You limits.yaml to set limits.. Adjust the value according to your needs.

```
# oc create -f limits.yaml  -n  jdeenada-pipeline-tasks-dev
```

Project template

A default project template has been made to openshift namespace by name "tester-project"
This template will create project and also set resource limits.

```
# oc get template -n openshift
tester-project
```

Example to create project with template:

```
# oc process -f tester-project -p PROJECT_NAME=tester PROJECT_DISPLAYNAME=tester
PROJECT_DESCRIPTION=tester_client PROJECT_ADMIN_USER=gamma
PROJECT_REQUESTING_USER=gamma | oc create -f -
```

User Creation template

Use the template "user-creation.yaml" is available to create user and associate user with htpasswd auth.

Eg:

```
# oc create -f user-creation.yaml -p <username>
```

Credentials for all the apps

OpenShift Web console: https://loadbalancer.00d0.example.opentlc.com/console/catalog
username: admin / PWD: r3dh4t1!

GIT: http://gogs-jdeenada-gogs.apps.00d0.example.opentlc.com/CICDLabs/openshift-tasks.git

gogs: http://gogs-jdeenada-gogs.apps.00d0.example.opentlc.com/CICDLabs/openshift-tasks
username: jdeenada / PWD: jdeenada

Jenkins: https://jenkins-jdeenada-pipeline-tasks-dev.apps.00d0.example.opentlc.com
usrname: admin / PWD: openshiftpipelines

All the users have password r3dh4t1!

Appendex

===Setup Nexus===

```
oc new-project jdeenada-nexus --display-name "Shared Nexus"

oc new-app sonatype/nexus3:latest

oc expose svc nexus3

oc rollout pause dc nexus3
```

===Change the deployment strategy from Rolling to Recreate and set requests and limits for memory.=====

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

oc set resources dc nexus3 --limits=memory=2Gi --requests=memory=1Gi
```

===Create a persistent volume claim (PVC) and mount it at /nexus-data.=====

```
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

==================Set up liveness and readiness probes for Nexus.===========

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/repository/maven-public/

===setup nexus repository===========

curl -o setup_nexus3.sh -s https://raw.githubusercontent.com/wkulhanek/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 rollout resume dc nexus3

===Setup sonarqube======

oc new-project jdeenada-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:6.7.4 --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

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
```

======Setup Gogs===============

```
oc new-project jdeenada-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
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=$HOME/app.ini
oc set volume dc/gogs --add --overwrite --name=config-volume -m /opt/gogs/custom/conf/ -t
configmap --configmap-name=gogs
git commit -m "Updated Settings" nexus_settings.xml nexus_openshift_settings.xml
git push gogs master
```

```
cd $HOME
git clone https://github.com/wkulhanek/openshift-tasks.git
cd $HOME/openshift-tasks
git remote add gogs http://jdeenada:redhat123@$(oc get route gogs -n jdeenada-gogs
--template='{{ .spec.host }}')/CICDLabs/openshift-tasks.git
git push -u gogs master
```