## Q.1 Configure LDAP.

User: Samwell

Given: ldap yaml , secret name- idm-secret, configmap= idm-ca-configmap

1. Create a secret with the LDAP bind password, which is required for the identity provider

configuration.

*oc create secret generic* ***idm-secret*** *--from-literal bindPassword=****'Redhat123@!'*** *-n openshift-config*

1. Create a **ca-config-map** configuration map in the **openshift-config** namespace. Ca.crt file will be present in given directory.

*oc create configmap -n openshift-config* ***idm-ca-configmap*** *--from-file=****ca.crt***

1. Edit the **ldap-cr.yml** file. Ensure the file matches the following

* Take backup before apply oauth

*oc get oauth cluster -o yaml > oauth-bkp.yaml*

*vi ldap-cr.yml*

apiVersion: config.openshift.io/v1

kind: OAuth

metadata:

name: cluster

spec:

identityProviders:

- htpasswd:

fileData:

name: htpasswd-secret

mappingMethod: claim

name: htpasswd\_provider

type: HTPasswd

- name: ldapidp

mappingMethod: claim

type: LDAP

ldap:

attributes:

id:

- dn

preferredUsername:

- uid

bindDN: "uid=admin,cn=users,cn=accounts,dc=ocp4,dc=example,dc=com"

bindPassword:

name: **idm-secret** <-

ca:

name: **ca-config-map** <-

insecure: false

url: "ldaps://idm.ocp4.example.com/cn=users,cn=accounts,dc=ocp4,dc=example,dc=com?uid"

*oc apply -f ldap-cr.yml*

*watch oc get pod -n openshift-authentication*

* check login with Samwell user.

*oc login –u samwell –p apiurl*

*oc logout*

*oc login -u kubeadmin -p apiurl*

*oc adm policy add-cluster-role-to-user cluster-admin samwell*

then run command that is provides in question.

## Q.2 Ansible playbooks is given, 5 playbooks are there.

1. main yaml

2. deployment

3. sa.yaml -- problem --- **metadata:dkfdkf** <- remove characters in from of metadata

4. route

5. service

One yaml is having everything written inside it, basically it is calling all remaining playbooks to deploy the app in openshift.

1. indentation is wrong we need to fix it

2. We need to 3 module for deploying deployment, service, route 🡪 just copy paste from above module in YAML

*ansible-playbook main.yaml*

## Q.3 Alertmanager

We will get a yaml we just need to fill the values

"global":

"resolve\_timeout": "5m"

smtp\_smarthost: **192.168.50.254:25**

smtp\_from: **alerts-review@ocp4.example.com**

smtp\_require\_tls: **false**

"receivers":

- "name": default

- name: email-notification

email\_configs:

- to: **ocp-review@example.com**

"route":

"group\_by":

- "namespace"

"group\_interval": "5m"

"group\_wait": "30s"

"receiver": default

"repeat\_interval": **2m**

"routes":

- "match":

"alertname": **"Watchdog"**

"receiver": default

- match:

severity: warning

receiver: email-notification

Take backup before apply

*oc extract secrets/alertmanager-main -n openshift-monitoring --to /tmp*

* Set alert manager from yaml

*oc set data secret/alertmanager-main --from-file=/tmp/alertmanager.yaml -n openshift-monitoring*

*oc logs -f -c alertmanager alertmanager-main-0 -n openshift-monitoring*  🡪 this command should show completed configuration

* Then run **mailx** command on workstation machine it will show a mail to you.

## Q.4 Cronjob

create a project **elementum** and inside it create a cron job using image: hiiiiii and set a serviceaccount also magna

1. create a project -- oc new-project elementum

2. create sa -- oc create sa magna

3. create a yamlfile for deploying cronjob using - -- oc create cronjob job-runner --image=imagename --schedule="5 4 2 \* \*" --dry-run -o yaml > cron.yaml

4. open the file using vim and add **successfulJobsHistoryLimit**: 14 and **serviceaccount** name as well magna. Please see the below yaml file

YAML :

apiVersion: batch/v1

kind: CronJob

metadata:

creationTimestamp: null

name: my-job

spec:

jobTemplate:

metadata:

creationTimestamp: null

name: my-job

spec:

template:

metadata:

creationTimestamp: null

spec:

**serviceAccountName: magna**

containers:

- image: busybox

name: my-job

resources: {}

restartPolicy: OnFailure

schedule: '\*/1 \* \* \* \*'

**successfulJobsHistoryLimit: 14**

5. oc create -f cronjob.yml

## Q.5 Migration - you will a yaml and A tar file

1. load that tar –

podman load -i redhat-landing.tar.gz

1. TAG THE IMAGE AND chnage to same as given question --

podman tag localhost/redhat-landing:v1 registry.domain19.example.com/redhat-landing:latest

1. push this image --- (credentials will be given in important info page)

podman login -u regopsadm -p redhat registry.domain19.example.com

podman push registry.domain19.example.com/redhat-landing:latest

4. open the yaml and make some changes -- 1. change image name to image name that u have uploaded

2. from service kind remove 2 lines of cluster IP

3. in ingress section give the correct URL that you will get in question.

mkdir base

cp migration.yml base/

vim kustomization.yml

---

resources:

- migration.yml

oc apply -k base

5. create it . -- oc create -f filename

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

namespace: prod

bases:

- ../../base

images:

- name: registry.ocp4.example.com:8443/redhattraining/versioned-hello

newTag: v1.1

* Command to check desired output

oc kustomize <foldername>

* Apply kustomize folder.

oc apply –k <foldername> 🡪 makesure **kustomization.yml** is present in that directory

## Q.6 Install operator local-storage in specific project called ex380-logging

cluster monitoring should be enable for the ex380-logging project

1. go to dashboard

2. create a project ex380-logging -- using yaml

apiVersion: v1

kind: Namespace

metadata:

labels:

openshift.io/cluster-monitoring: "true"

name: ex380-logging

* Go to Operator-hub and install local-storage to same namespace.

## Q.7 NTP machine config -- need to configure NTP using machine config

the name of machine config should be 60-master-ntpd-config, 60-worker-ntpd-config

cat chrony.conf

server 101.101.101.101 trust

driftfile /var/lib/chrony/drift

makestep 1.0 -1

rtcsync

keyfile /etc/chrony.keys

leapsectz right/UTC

logdir /var/log/chrony

cat chrony.conf | base64 -w0 -

jAuMTA1IHRydXN0CnNlcnZlciAxMC4xNDYuMTMwLjExMiAKZHJ

oc get machineconfig

* Open any machineconfig file and only leave following content.

Existing:

spec:

config:

ignition:

version: 3.2.0

storage:

files:

- contents:

source: >-

data:,unqualified-search-registries%20%3D%20%5B'registry.access.redhat.com'%2C%20'docker.io'%5D%0A

mode: 420

overwrite: true

path: /etc/containers/registries.conf

extensions: null

fips: false

kernelArguments: null

kernelType: ''

osImageURL: >-

quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:368d9b871acb9fc29eea6a4f66e42894677594e91834958c015ed15c03ebe79e

Required:

spec:

config:

ignition:

version: 3.2.0

storage:

files:

- contents:

source: **data:text/plain;charset=utf-8;base64**,c2VydmVyIDEwLjEwMC4xMjAuMTA1IHRydXN0CnNlcnZlciA

mode: 420

overwrite: true

path: /etc/chrony.conf

osImageURL: "" <<-

* Remember highlighted to add and create two machineconfig {master, worker}

oc apply -f worker-chrony.yaml

oc apply -f master-chrony.yaml

## Q.8 PV,PVC,DEPLOYMENT

PV - Details will be given -

pv name -- landing-pv size, access mode and policy should be same as storage class (1GI,READONLYMANY,RETAIN)

pvc details will be given -- pvcname --- landing-pvc , accessmode Readonlymany size 1 gi

deployment details will be given -- CREATE A DEPLOYMENT USING THIS IMAGE AND MOUNT THE VOLUME AT nginxunpriviledge /usr/share/nginx/html/

oc get sc

oc get sc nfs-storage –o yaml --- remember the path where nfs is going to create dir and nfs server hostname also

create pv -- below yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: landing-pv

spec:

capacity:

storage: 1Gi

volumeMode: Filesystem

accessModes:

- ReadOnlyMany

persistentVolumeReclaimPolicy: Retain

storageClassName: nfs-storage

nfs:

path: /tmp

server: nfs.domain19.example.com

4. create pvc -- below yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: landing-pvc

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

storageClassName: nfs-storage

NOTE: Get yaml from Openshift console.

5. If both are bounded then create deployment yaml and mount pvc in container in this location /usr/share/nginx/html/

6. after running expose it on 8080 port

7. expose a svc to get route but https they need to use -- oc create route edge --service=page

8. accessible on given URL

oc new-app --name http --image=registry.apps.uatintocp.adcbcp.local/dataplatform/httpd-24:latest

oc set volume deploy/http --add --name=v1 --type=persistentVolumeClaim --claim-name=pvc-test --mount-path=/data

oc get po

oc expose deployment –h

oc expose deploy http --port=8080

oc create route edge --service=http

8. accessible on given URL

## Q.9 schedule a pod in worker 2 machine.

1. ssh core@worker2.lab.example.com "sudo systemctl is-active crio"

2. ssh core@worker2.lab.example.com "sudo systemctl is-active kubelet"

3. ssh core@worker2.lab.example.com "sudo systemctl is-enabled crio"

4. ssh core@worker2.lab.example.com "sudo systemctl is-enabled kubelet"

5. ssh core@worker2.lab.example.com "sudo systemctl enable --now crio"

oc status --suggest

6. oc edit deployment deployment name -- check nodeselector they are using as region=redhat-na

7. oc adm uncordon worker2.domain19.example.com

8. oc label node worker2.domain19.exameple.com region=redhat-na

The pod will start creating container but there will be memory issue

10. oc edit dpeloyment name -- in yaml chnage the limit value to 200Mi

It will start running and you can verify that on ROUTE url that is given