**Question-1: Authenticating Users**

Enable “HTPasswdPasswordIdentityProvider" as an identity provider method.

Ensure no other user should

Create three users greg, joe, anthony to access oc gui and cli having password "**zaldebra**".

Use "HTPasswdPasswordIdentityProvider" as an identity provider method.

Ensure no other user should be allowed to access any oc projects.

User password information are stored under: /etc/origin/master/htpasswd

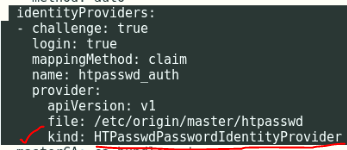
##################################################################################

**Solution-1:**

Login into Master Node open the file "/etc/origin/master/master-config.yaml" and

verify if "HTPasswdPasswordIdentityProvideridentity" is set under "identityProviders"





kind: HTPasswdPasswordIdentityProvider <---- Kind should be "HTPasswdPasswordIdentityProvider"

**Verify**

[root@master ~]# grep HTPasswdPasswordIdentityProvider /etc/origin/master/master-config.yaml

kind: HTPasswdPasswordIdentityProvider

[root@master ~]#

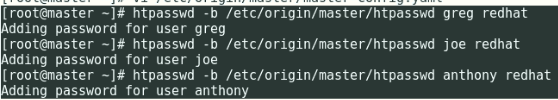


**Now create greg, joe, anthony users from master node**

# htpasswd -b /etc/origin/master/htpasswd greg zaldebra

# htpasswd -b /etc/origin/master/htpasswd joe zaldebra

# htpasswd -b /etc/origin/master/htpasswd anthony zaldebra

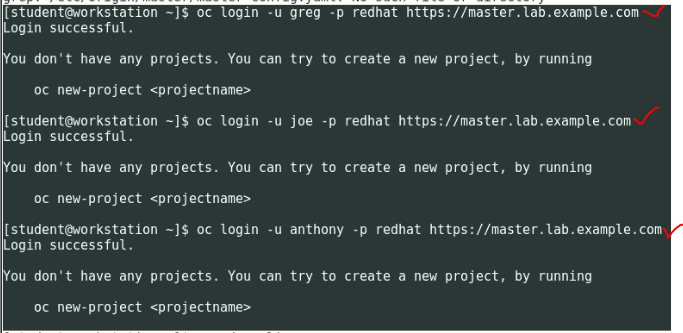


**Now, from workstation node, verify the login for the newly created users**

# oc login -u grep -p redhat <https://master.lab.example.com>

# oc login -u joe -p redhat <https://master.lab.example.com>

# oc login -u anthony -p redhat <https://master.lab.example.com>



##################################################################################

**Question-2: Projects and Roles to Users**

Create five projects

stanley

sydney

farm

tokyo

nginx

having a description "EX280 Project For Application Deployment".

greg should have admin access over "stanley" and "sydney".

anthony should be able to view "sydney" project but should not be able to modify/customize.

anthony should have admin access over project "farm"

joe has administrative access to tokyo nad nginx

##################################################################################

**Solution-2:**

**Login into Master node**

**Project Creation**

[student@workstation ~]$ ssh root@master

Last login: Sun Jul 14 10:44:49 2019 from workstation.lab.example.com

[root@master ~]#

[root@master ~]# oc whoami

system:admin

[root@master ~]#

# oc new-project stanley --description "EX280 Project For application deployment"

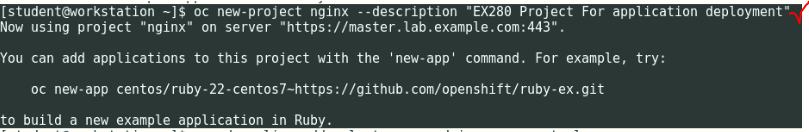
# oc new-project sydney --description "EX280 Project For application deployment"

# oc new-project farm --description "EX280 Project For application deployment"

# oc new-project tokyo --description "EX280 Project For application deployment"

# oc new-project nginx --description "EX280 Project For application deployment"





**Role Assignment**

greg should have admin access over "stanley" and "sydney".

anthony should be able to view "sydney" project but should not be able to modify/customize.

anthony should have admin access over project "farm"

joe has administrative access to tokyo nad nginx

[root@master ~]# oc whoami

system:admin

[root@master ~]#

# oc adm policy add-role-to-user admin greg -n stanley

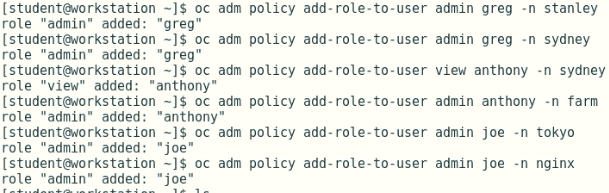
# oc adm policy add-role-to-user admin greg -n sydney

# oc adm policy add-role-to-user view anthony -n sydney

# oc adm policy add-role-to-user admin anthony -n farm

# oc adm policy add-role-to-user admin joe -n tokyo

# oc adm policy add-role-to-user admin joe -n nginx



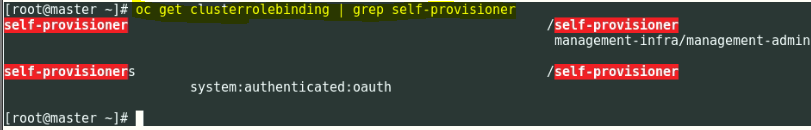
**Remove self-provisioner, system:authenticated, system:authenticated:oauth**

# oc get clusterrolebinding | grep self-provisioner

# oc adm policy remove-cluster-role-from-group self-provisioner system: authenticated system:authenticated:oauth

# oc get clusterrolebinding | grep self-provisioner





##################################################################################

**Question 3: PV Creation**

Create a persistent Volume of "exam-registry-volume" using /OSE\_registry "**exam-registry-volume**" to be bound with a PVC "**exam-registry-claim**" for substituting/replacing the default "**docker-registry**" volume.

##################################################################################

**Solution: 3**

Go to page no.**322** in **Developer Guide**. You need to add the last 3 lines of claimRef, name & namespace in exam-registry-volume.yml

# vi exam-registry-volume.yml

apiVersion: v1

kind: PersistentVolume

metadata:

  name: exam-registry-volume

spec:

  capacity:

    storage: 1Gi

  accessModes:

  - ReadWriteOnce

  nfs:

    path: /OSE\_registry

    server: [services.lab.example.com](http://workstation.lab.example.com/)

  persistentVolumeReclaimPolicy: Recycle

  claimRef:

    name: exam-registry-claim

    namespace: default

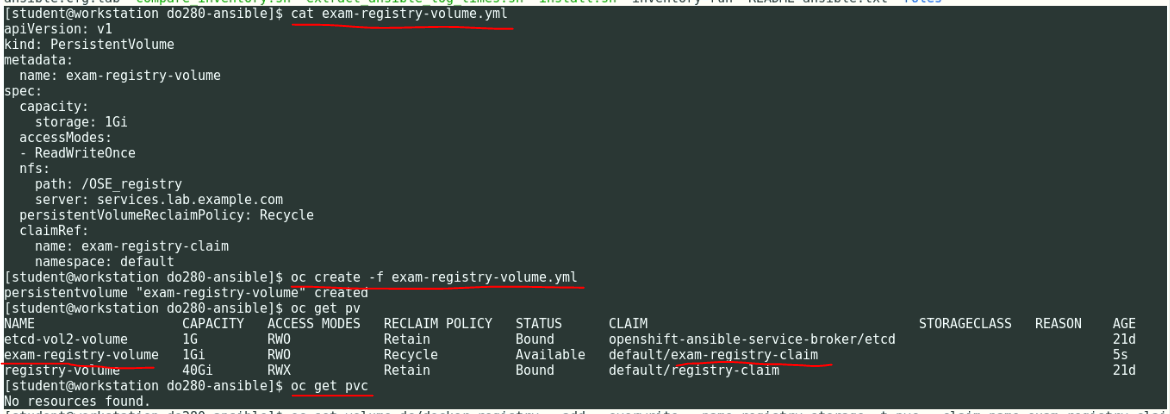
# oc create -f exam-registry-volume.yml

# oc get pv; oc get pvc

# oc set volume dc/docker-registry --add --overwrite --name=registry-storage -t pvc --claim-name=exam-registry-claim --claim-size=1Gi --claim-mode='ReadWriteOnce'

# oc get pvc

# oc set volume dc/docker-registry





**Question 4: Resource Management**

Create a resource quota "EX280-Quota" for defining limits for resource access for projects in "sydney" project.

i. Total memory amount for max usage is 2Gi

ii. Total CPU unit for max usage is "2" kubernetes units

iii. Total no of pods should be limited to 13

iv. Total no services should not exceed 10

v. Total no of replication controllers should be 3

Create another resource limit file "EX280-QuotaLimit" for defining range of resource access based on soft/hard limit.

i. For pods min cpu limit is "5m" and max is "500m"

ii. For containers min cpu limit is "100m" and max is "500m" and default request of "300m"

iii. For pods min memory is "300Mi" and max is "500Mi"

iv. For containers min memory is "200Mi" and max is "600Mi" and default request of "400Mi"

##################################################################################

**Solution-4:**

**Change to "sydney" project**

[student@workstation ~]$ oc whoami i

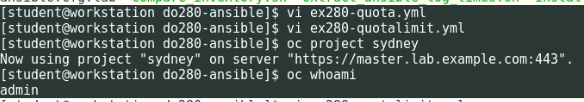
admin

[student@workstation ~]$ oc project sydney

Now using project "sydney" on server <https://master.lab.example.com:443>

[student@workstation ~]$

Go to page no. **213 & 214** in **Developer Guide**. Read the question and carefully add the respective entries in ex280-quota.yml file.



**Create the Quota with below YML file**

[student@workstation ~]$ cat ex280-quota.yml

apiVersion: v1

kind: ResourceQuota

metadata:

name: ex280-quota

spec:

hard:

pods: "13"

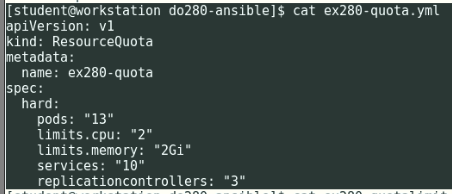
limits.cpu: "2"

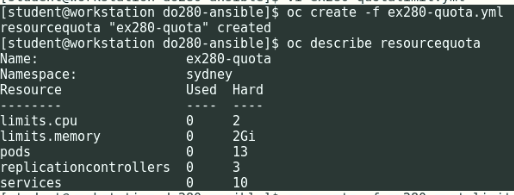
limits.memory: "2Gi"

replicationcontrollers: "3"

services: "10"

[student@workstation ~]$





**Create the Quota with below YML file**

Go to page no. **220** in **Developer Guide**. Read the question carefully and add the respective entries in below yaml file.

[student@workstation ~]$ cat ex280-quotalimit.yml

apiVersion: "v1"

kind: "LimitRange"

metadata:

name: "core-resource-limits"

spec:

limits:

- type: "Pod"

max:

cpu: "500m"

memory: "300Mi"

min:

cpu: "10m"

memory: "5Mi"

- type: "Container"

max:

cpu: "500m"

memory: "1Gi"

min:

cpu: "10m"

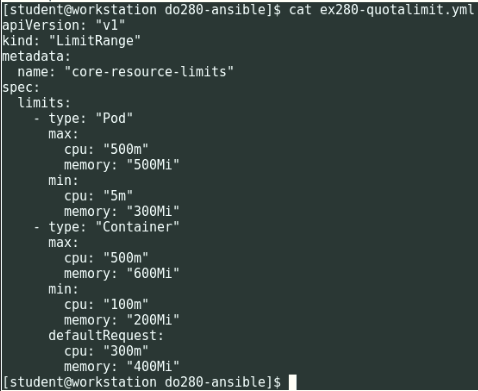
memory: "4Mi"

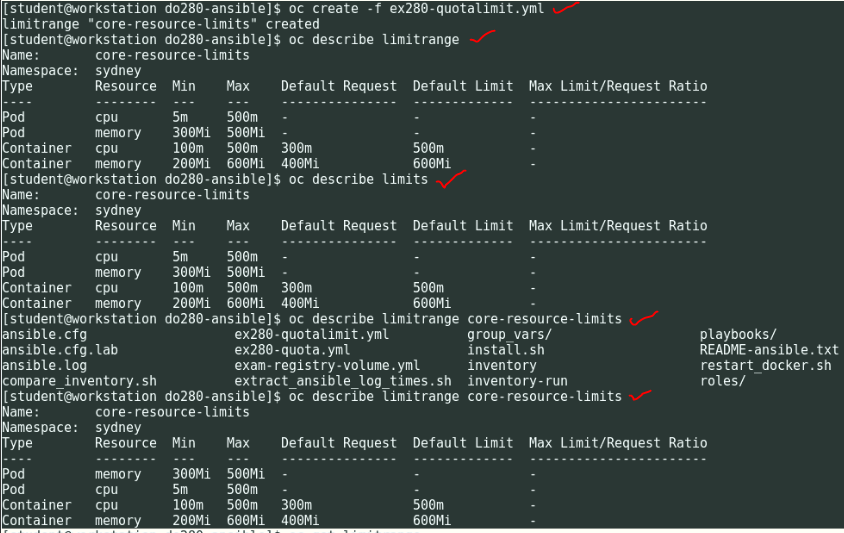
defaultRequest:

cpu: "300m"

memory: "400Mi"

[student@workstation ~]$



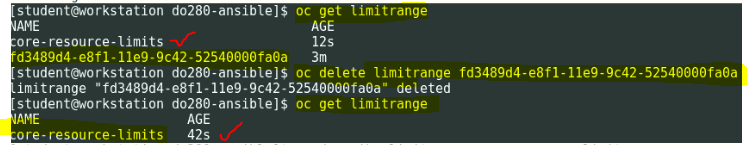




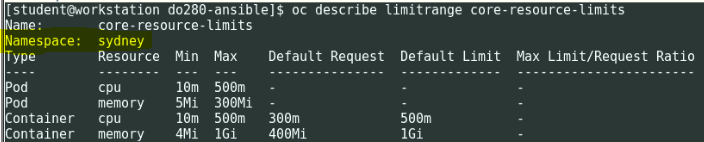
\*\*\*\*\*\*\*\*\* Make Sure Project is sydney \*\*\*\*\*\*\*\*\*

**Note:** By Mistake if you create quotas in different project, you can delete the quota using below commands and recreate it by using above commands quickly.









##################################################################################

**Question 5: Create an APP from a GIT Repository**

- As user anthony under "farm" project create an application using a GIT URL <http://registry.lab.example.com/version> accessible from registry.

- Make change in GIT FILE "index.php" to replace "PLACEHOLDER" with the text available at url http://rhelx.......

- Create a web hook to trigger build

- Application should be accessible with "http://php.apps.lab.example.com"

##################################################################################

**Solution-5:**

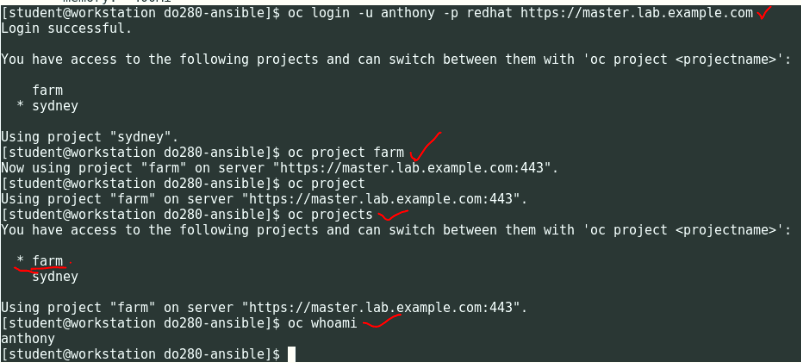
**Login as anthony user and move to farm project**

# oc login –u Anthony –p redhat <https://master.lab.example.com>

# oc project farm

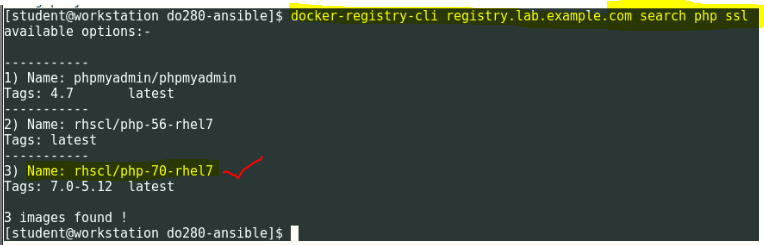
# oc projects

# oc whoami



**Now, search for PHP image in the docker-registry**

# docker-registry-cli registry.lab.example.com search php ssl



Name**: rhscl/php-70-rhel7** \*\*\*\*\*\*\*\*\*\*\* This is the Image to be used, make sure it is present in the registry \*\*\*\*\*\*\*

**Now create the APP**

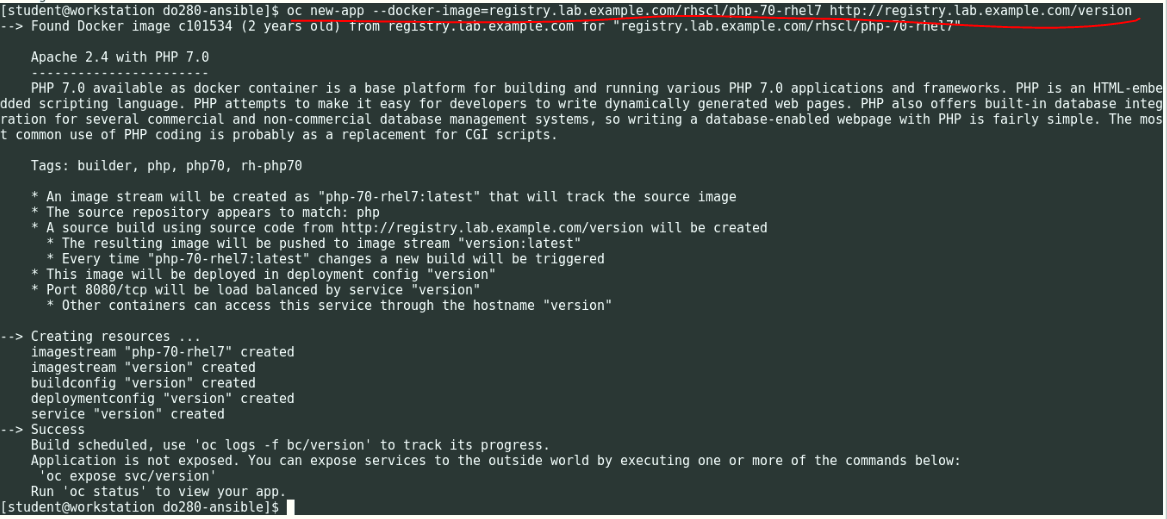
# oc new-app --docker-image=registry.lab.example.com/rhscl/php-70-rhel7 <http://registry.lab.example.com/version>

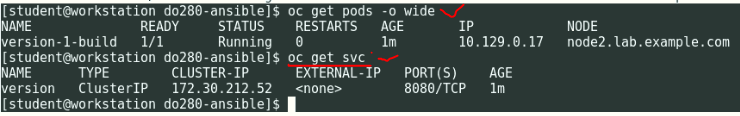
# oc get pod

# oc get pods -o wide

# oc get pods -o wide

# oc get svc



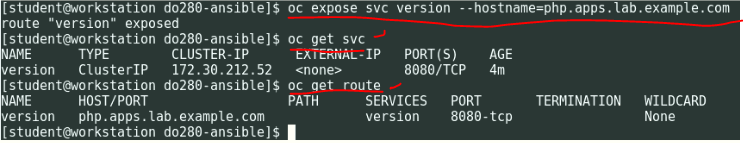


**Now, expose the service for version pod**

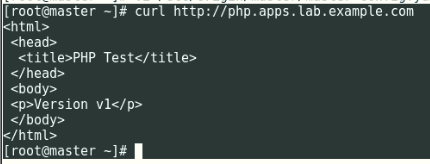
# oc expose svc version --hostname=php.apps.lab.example.com

# oc get svc

# oc get route



**Now, do Curl to** [**http://php.apps.lab.example.com**](http://php.apps.lab.example.com)



**Now, git checkout version and modify index.php file**

# mkdir php

# cd php

# git clone http://registry.lab.example.com/version

# cd version/

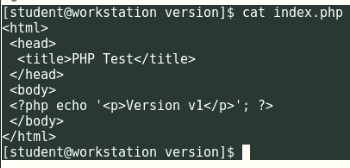
# ls

index.php

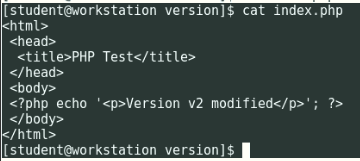
# vi index.php







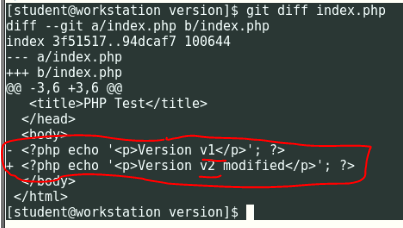


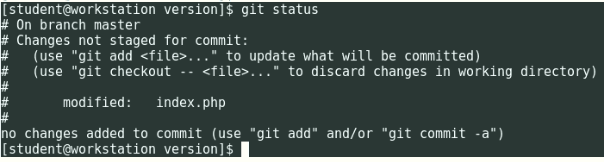


**Check the GIT Status**

# git diff index.php

# git status





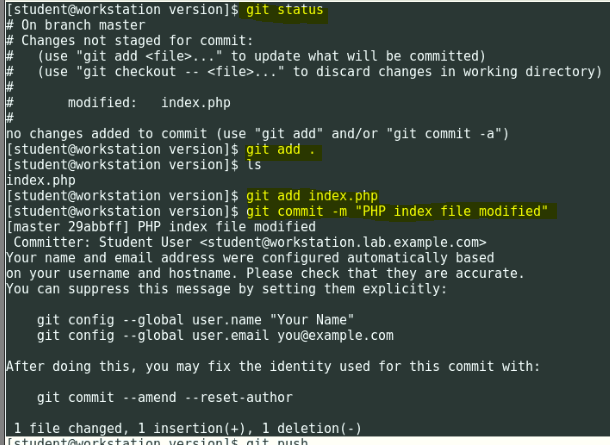
**Now, commit the file**

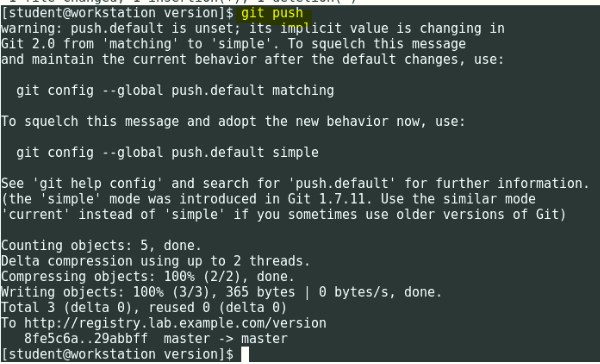
# git status

# git add index.php

# git commit –m “PHP index file modified”

# git push



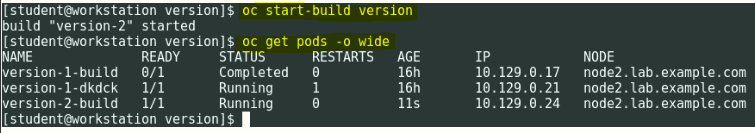


**Now, rebuild the version Build Config**

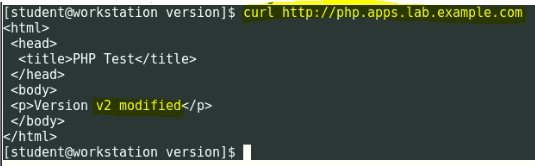
# oc start-build version

**Wait for 2-3 mins to build POD/Container:**

# oc get pods –o wide



**Now, do Curl to** [**http://php.apps.lab.example.com**](http://php.apps.lab.example.com)



You can also check in Firefox in student workstation GUI. Login into workstation and type[**http://php.apps.lab.example.com**](http://php.apps.lab.example.com) **and press Enter.**



##################################################################################

**Question 6: Create an APP using Edge Termination**

Create a new application name hello in project tokyo using the docker image

registry.lab.example.com/openshift/hello-openshift

Application could be accessible at <https://hello.apps.lab.example.com> where TLS termination occurs at the router.

You may use following file for creating TLS certificates at the router http://.../create-cert.sh

##################################################################################

**Solution-6:**

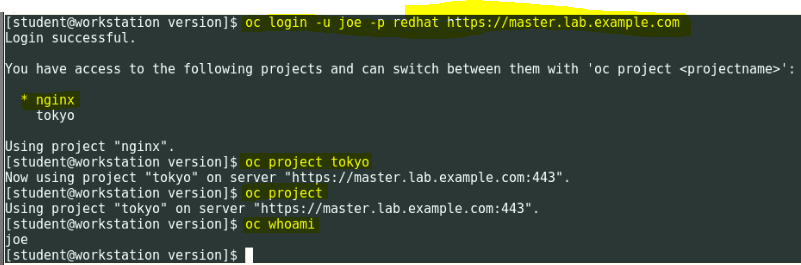
**Login as "joe" and move to project tokyo:**

# oc login –u joe –p redhat <https://master.lab.example.com>

# oc project tokyo

# oc project

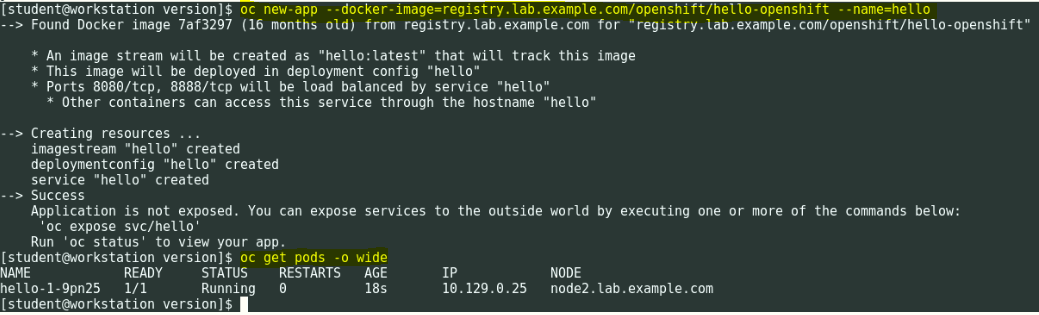
# oc whoami



**Create the APP**

# oc new-app --docker-image=registry.lab.example.com/openshift/hello-openshift --name=hello

# oc get pods –o wide

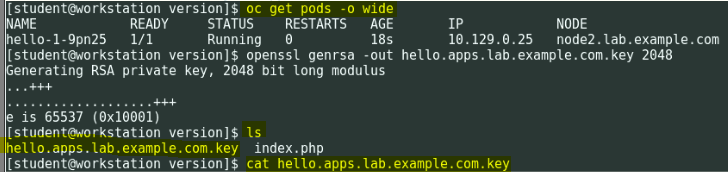


**Create the Certificate:**

**Generate private key**

# openssl genrsa -out hello.apps.lab.example.com.key 2048

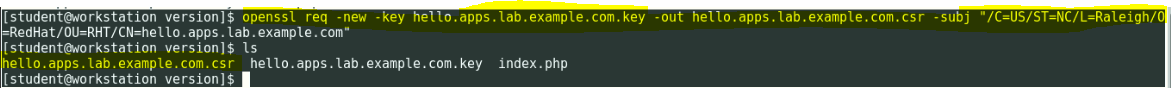
# ls hello.apps.lab.example.com.key



**Generate CSR**

# openssl req -new -key hello.apps.lab.example.com.key -out hello.apps.lab.example.com.csr -subj "/C=US/ST=NC/L=Raleigh/O=RedHat/OU=RHT/CN=hello.apps.lab.example.com"

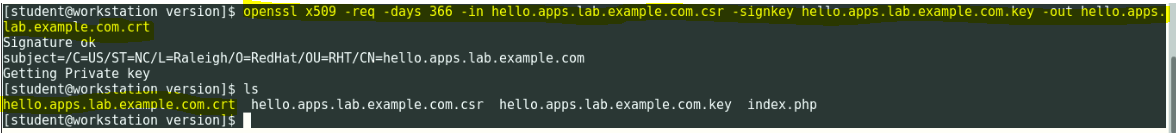
# ls hello.apps.lab.example.com.csr



**Generate a certificate**

# openssl x509 -req -days 366 -in hello.apps.lab.example.com.csr -signkey hello.apps.lab.example.com.key -out hello.apps.lab.example.com.crt

# ls hello.apps.lab.example.com.crt

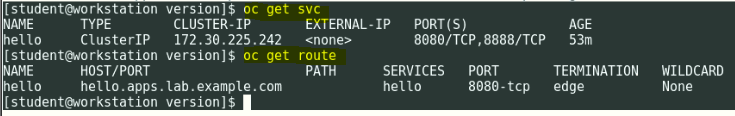


**Create a secure edge route**

# oc create route edge --service=hello --hostname=hello.apps.lab.example.com --key=hello.apps.lab.example.com.key --cert=hello.apps.lab.example.com.crt



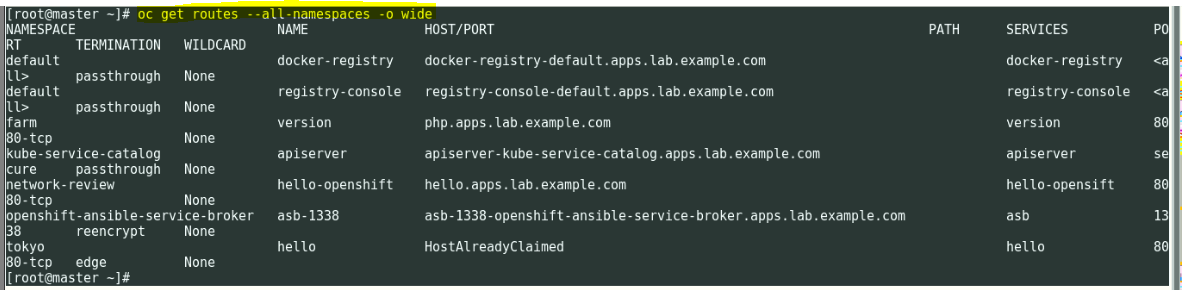
**Verify Route**

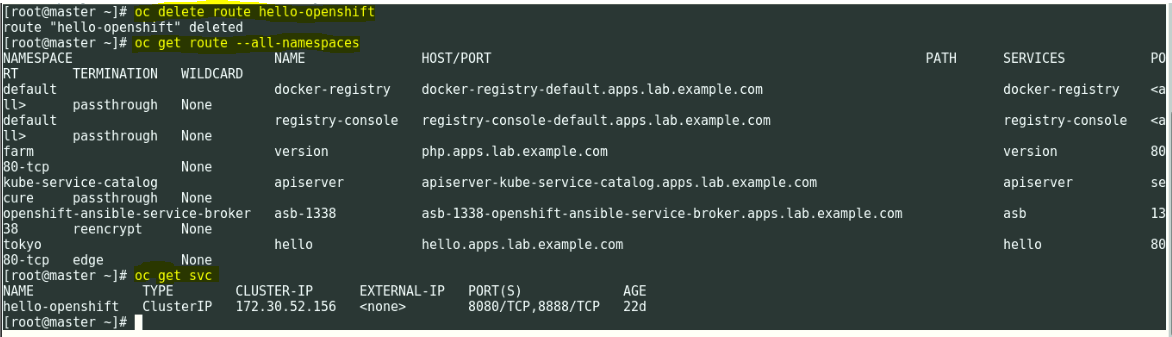


If you found **HostAlreadyClaimed** in HOST column. Please run the below commands:

# oc get route --all-namespaces -o wide 🡪 Run from Master Node.



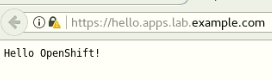




**Verify the APP**



You can also check in Firefox in student workstation GUI. Login into workstation and type[**http://hello.apps.lab.example.com**](http://hello.apps.lab.example.com) **and press Enter.**



##################################################################################

**Question 7: Replication**

Create Five Replication of HELLO Application POD create in Question No: 6 for the tokyo project.

##################################################################################

**Solution-7:**

# oc whoami

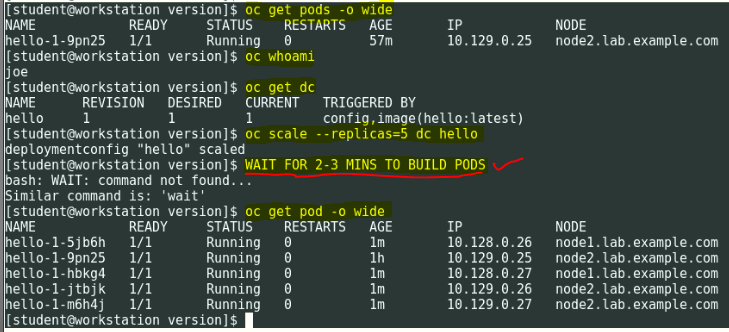
# oc get pod –o wide

# oc get dc

# oc scale –replicas=5 dc hello

Wait for 2-3 mins

# oc get pod –o wide



##################################################################################

**Question 8: Metric Application**

Configure Openshift Metrics subsystem in terms of (run this one workstation in classroom but in exam you have to run it in master node) following requirements.

a) Use the host file /root/hosts and yml available here

b) Metrics subsystem data storage should

/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml

Configured as persistent volume in nfs /OSE\_cassandra shared directory

b) Your metrics subsystem should be created with following parameters:

i. openshift\_metrics\_image\_version=v3.9

ii. openshift\_metrics\_heapster\_requests\_memory=300M

iii. openshift\_metrics\_hawkular\_requests\_memory=750M

iv. openshift\_metrics\_cassandra\_requests\_memory=750M

v. openshift\_metrics\_cassandra\_storage\_type=pv

vi. openshift\_metrics\_cassandra\_pvc\_prefix=cassandra

vii. openshift\_metrics\_cassandra\_pvc\_size=5Gi

viii. openshift\_metrics\_install\_metrics=True

##################################################################################

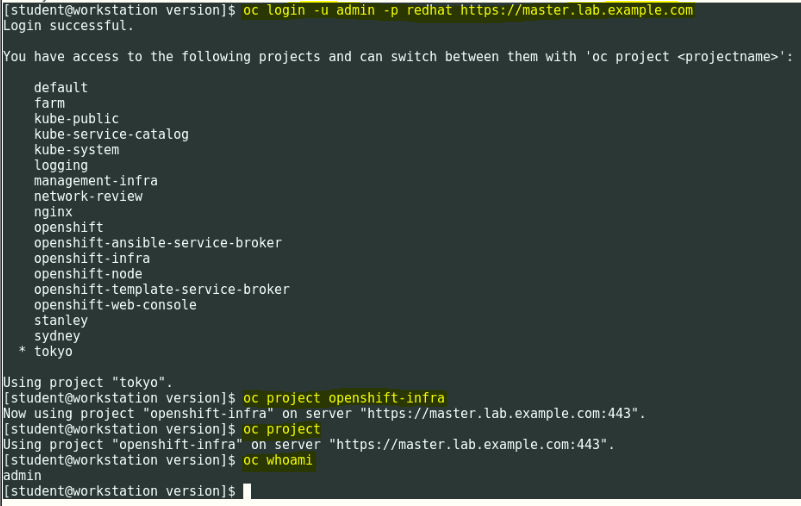
**Solution-8:**

**Use Openshift-infra project and with admin user: Master Node**

# oc login –u admin –p redhat <https://master.lab.example.ccom>

# oc project Openshift-infra

# oc whoami



**Now, on the services machine create the NFS directory for**

# ssh root@services

# mkdir -p /var/export/OSE\_cassandra

# chown nfsnobody:nfsnobody /var/export/OSE\_cassandra

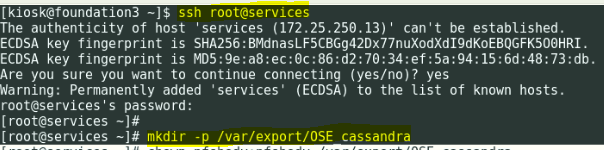
# echo "/var/export/OSE\_cassandra \*(rw,async,all\_squash)" > /etc/exports.d/OSE\_cassandra.exports

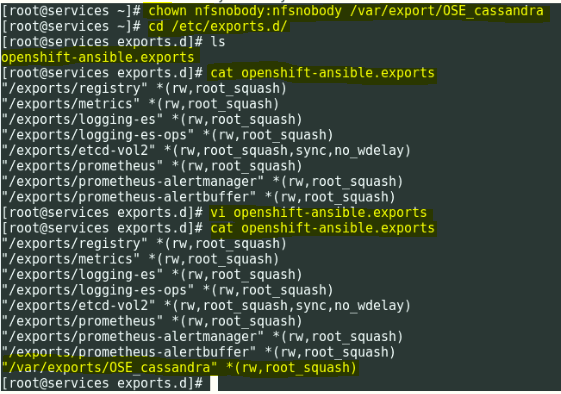
# exportfs

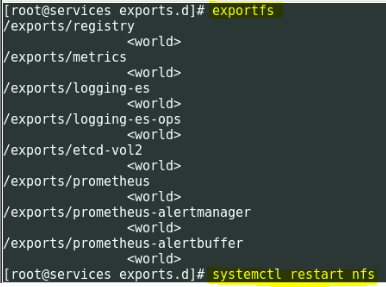
# systemctl restart nfs

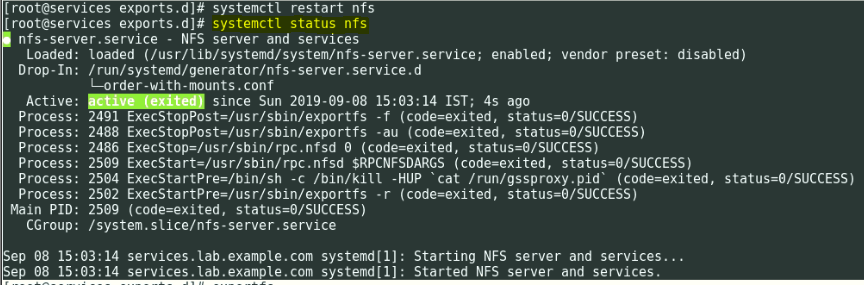
# systemctl status nfs

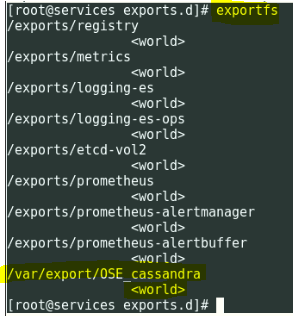
# exportfs











**Now, create the PV**

# oc whoami

# oc get pv

# oc project

[student@workstation ~]$ cat metrics-pv.yml

apiVersion: "v1"

kind: "PersistentVolume"

metadata:

name: "cassandra-pv"

spec:

capacity:

storage: "5Gi"

accessModes:

- "ReadWriteOnce"

nfs:

path: "/var/export/OSE\_cassandra"

server: "services.lab.example.com"

persistentVolumeClaimPolicy: "Recycle"

claimRef:

name: "cassandra-1"

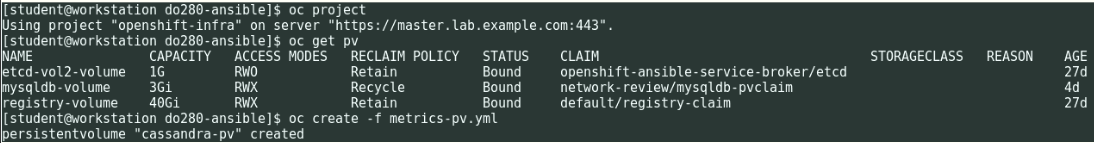
namespace: "openshift-infra"

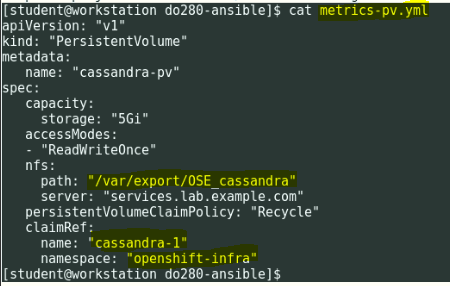
[student@workstation ~]$

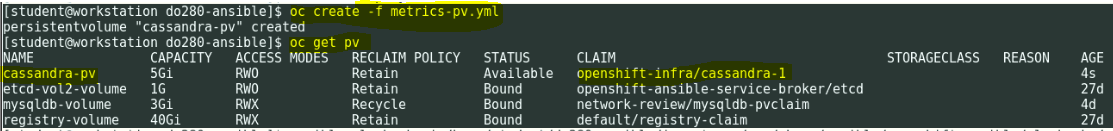
**Note:** In above Cassandra/metrics PV volume always use the claim reference name as “Cassandra-1” / ”metrics-1”. Use (-1) only.

# oc create –f metrics-pv.yml

# oc get pv







**Create the inventory file**

[student@workstation ~]$ cat inventory

[workstations]

workstation.lab.example.com

[nfs]

services.lab.example.com

[masters]

master.lab.example.com

[etcd]

master.lab.example.com

[nodes]

master.lab.example.com

node1.lab.example.com openshift\_node\_labels="{'region':'infra', 'node-role.kubernetes.io/compute':'true'}"

node2.lab.example.com openshift\_node\_labels="{'region':'infra', 'node-role.kubernetes.io/compute':'true'}"

[OSEv3:children]

masters

etcd

nodes

nfs

#Variables needed by the prepare\_install.yml playbook.

[nodes:vars]

registry\_local=registry.lab.example.com

use\_overlay2\_driver=true

insecure\_registry=false

run\_docker\_offline=true

docker\_storage\_device=/dev/vdb

[OSEv3:vars]

#General Variables

openshift\_disable\_check=disk\_availability,docker\_storage,memory\_availability

openshift\_deployment\_type=openshift-enterprise

openshift\_release=v3.9

openshift\_image\_tag=v3.9.14

#OpenShift Networking Variables

os\_firewall\_use\_firewalld=true

openshift\_master\_api\_port=443

openshift\_master\_console\_port=443

openshift\_master\_default\_subdomain=apps.lab.example.com

#Cluster Authentication Variables

openshift\_master\_identity\_providers=[{'name': 'htpasswd\_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]

openshift\_master\_htpasswd\_users={'admin': '$apr1$4ZbKL26l$3eKL/6AQM8O94lRwTAu611', 'developer': '$apr1$4ZbKL26l$3eKL/6AQM8O94lRwTAu611'}

#Need to enable NFS

openshift\_enable\_unsupported\_configurations=true

#Registry Configuration Variables

openshift\_hosted\_registry\_storage\_kind=nfs

openshift\_hosted\_registry\_storage\_access\_modes=['ReadWriteMany']

openshift\_hosted\_registry\_storage\_nfs\_directory=/exports

openshift\_hosted\_registry\_storage\_nfs\_options='\*(rw,root\_squash)'

openshift\_hosted\_registry\_storage\_volume\_name=registry

openshift\_hosted\_registry\_storage\_volume\_size=40Gi

#etcd Configuration Variables

openshift\_hosted\_etcd\_storage\_kind=nfs

openshift\_hosted\_etcd\_storage\_nfs\_options="\*(rw,root\_squash,sync,no\_wdelay)"

openshift\_hosted\_etcd\_storage\_nfs\_directory=/exports

openshift\_hosted\_etcd\_storage\_volume\_name=etcd-vol2

openshift\_hosted\_etcd\_storage\_access\_modes=["ReadWriteOnce"]

openshift\_hosted\_etcd\_storage\_volume\_size=1G

openshift\_hosted\_etcd\_storage\_labels={'storage': 'etcd'}

#Modifications Needed for a Disconnected Install

oreg\_url=registry.lab.example.com/openshift3/ose-${component}:${version}

openshift\_examples\_modify\_imagestreams=true

openshift\_docker\_additional\_registries=registry.lab.example.com

openshift\_docker\_blocked\_registries=registry.access.redhat.com,docker.io

openshift\_web\_console\_prefix=registry.lab.example.com/openshift3/ose-

openshift\_cockpit\_deployer\_prefix='registry.lab.example.com/openshift3/'

openshift\_service\_catalog\_image\_prefix=registry.lab.example.com/openshift3/ose-

template\_service\_broker\_prefix=registry.lab.example.com/openshift3/ose-

ansible\_service\_broker\_image\_prefix=registry.lab.example.com/openshift3/ose-

ansible\_service\_broker\_etcd\_image\_prefix=registry.lab.example.com/rhel7/

[student@workstation ~]$

**Now, install the Hawkular Metrics using Ansible from Master node:**

# ansible-playbook -i /home/student/do280-ansible/inventory /usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml -e openshift\_metrics\_image\_version=v3.9 -e openshift\_metrics\_heapster\_requests\_memory=300M -e openshift\_metrics\_hawkular\_requests\_memory=750M -e openshift\_metrics\_cassandra\_requests\_memory=750M -e openshift\_metrics\_cassandra\_storage\_type=pv -e openshift\_metrics\_cassandra\_pvc\_size=5Gi -e openshift\_metrics\_cassandra\_pvc\_prefix=metrics -e openshift\_metrics\_install\_metrics=True -e openshift\_metrics\_hawkular\_hostname=hawkular-metrics.apps.lab.example.com –e openshift\_metrics\_image\_prefix=registry.lab.example.com/openshift3/ose-

**OR**

[student@workstation ~]$ ansible-playbook -i /home/student/do280-ansible/inventory /usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml \

-e openshift\_metrics\_image\_version=v3.9 \

-e openshift\_metrics\_heapster\_requests\_memory=300M \

-e openshift\_metrics\_hawkular\_requests\_memory=750M \

-e openshift\_metrics\_cassandra\_requests\_memory=750M \

-e openshift\_metrics\_cassandra\_storage\_type=pv \

-e openshift\_metrics\_cassandra\_pvc\_prefix=metrics \

-e openshift\_metrics\_cassandra\_pvc\_size=5Gi \

-e openshift\_metrics\_install\_metrics=True \

-e openshift\_metrics\_hawkular\_hostname=hawkular-metrics.apps.lab.example.com \

-e openshift\_metrics\_image\_prefix=registry.lab.example.com/openshift3/ose-

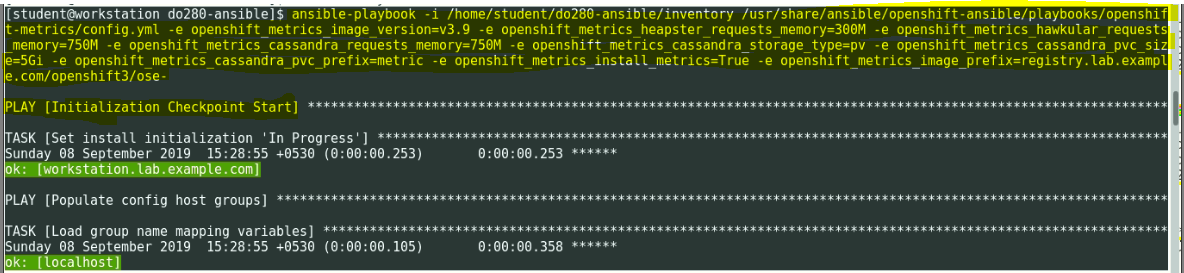
**Note:** V.V.IMP (Remember these 2 parameters).

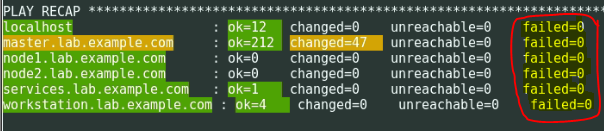
“-e openshift\_metrics\_hawkular\_hostname=hawkular.apps.lab.example.com \

-e openshift\_metrics\_image\_prefix=registry.lab.example.com/openshift3/ose-“)

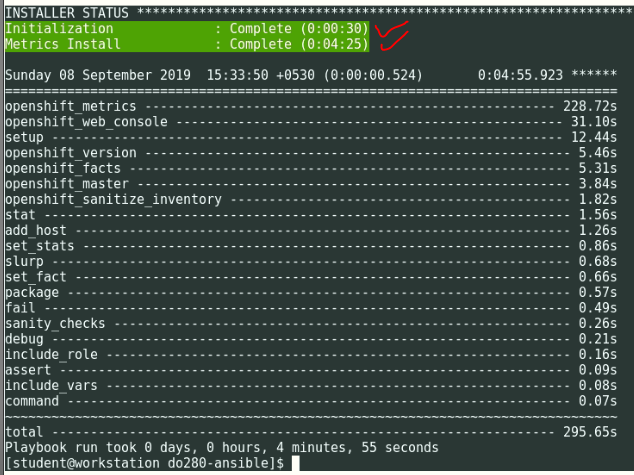
# oc get pv

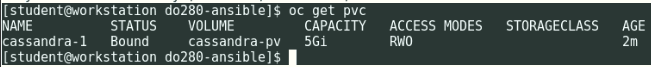
# oc get pvc

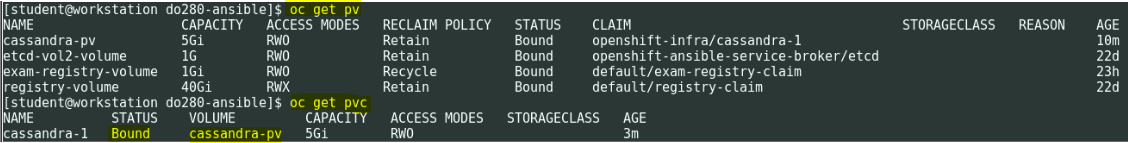




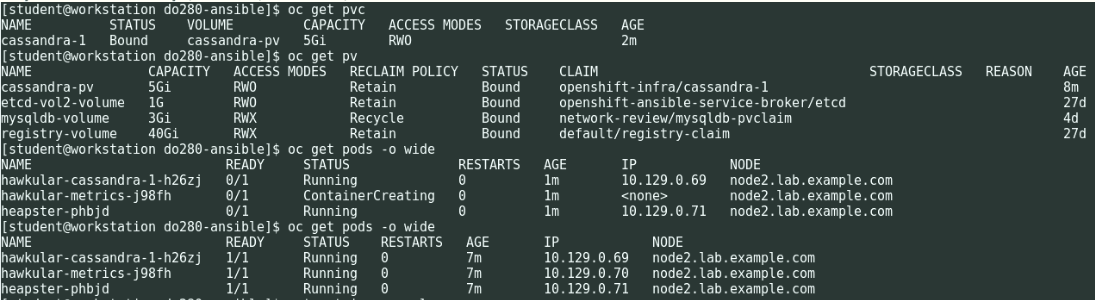
**Make sure in the PLAY RECAP "failed" should ZERO and not ONE**







**Wait for 5-6 mins till all PODS REDAY state is 1/1**



**Now get the route info**

# oc get route



[student@workstation ~]$ curl -k https://hawkular-metrics.apps.lab.example.com

<!--

~ Copyright 2014-2015 Red Hat, Inc. and/or its affiliates

~ and other contributors as indicated by the @author tags.

~

~ Licensed under the Apache License, Version 2.0 (the "License");

~ you may not use this file except in compliance with the License.

~ You may obtain a copy of the License at

~

~ http://www.apache.org/licenses/LICENSE-2.0

~

~ Unless required by applicable law or agreed to in writing, software

~ distributed under the License is distributed on an "AS IS" BASIS,

~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

~ See the License for the specific language governing permissions and

~ limitations under the License.

-->

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<meta http-equiv="refresh" content="0; url=hawkular/metrics">

<title>Hawkular Metrics</title>

</head>

<body>

<h1>Hawkular Metrics</h1>

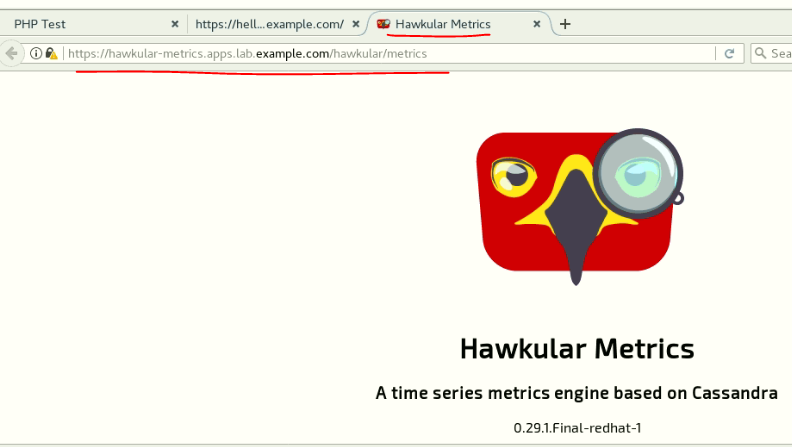
<h3>A time series metrics engine based on Cassandra</h3>

</body>

</html>

[student@workstation ~]$

You can also check in **Firefox** in student workstation GUI.



**To Delete Metrics/Cassandra POD’s:**

oc get all

oc delete all --all

oc delete pvc <pvc-name>

oc delete pv <pv-name>

oc delete pod recycler-for-metrics-pv -n openshift-infra --grace-period=0 --force

oc get pod --> if anything you found delete POD again.

**Manually delete one by one then follow the below procedure.**

oc delete svc <svc-name

oc delete route <route-name>

oc delete pvc <pvc-name>

oc delete all --all

oc delete pod recycler-for-metrics-pv -n openshift-infra --grace-period=0 --force

oc delete pv <pv-name>

##################################################################################

**Question 9: Create WORDPRESS APP Site**

Create a worpress site using http://classroom.example.com/wordpress.tar.gz for image.

Use wordpress-pod.yml and mysql-pod.yml

Wordpress pod should use /OSE\_wordpress for persistent volume

Mysql pod should use /OSE\_mysql for persistent volume

Wordpress admin user greg and password faqreg

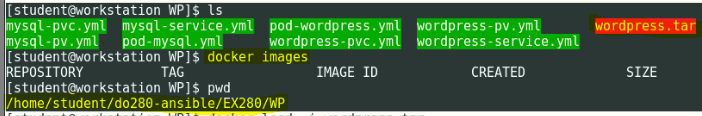
Wordpress site should use persistent mysql database use /OSE\_mysql storagea and use /OSE\_wordpress for hosting.

##################################################################################

**Solution-9:**

**Download the wordpress.tar file from docker HUB and save the image using docker save cmd**

Upload the tar file into registry.lab.example.com



**Load wordpress.tar file using docker:**

# docker images

# docker load –i wordpress.tar

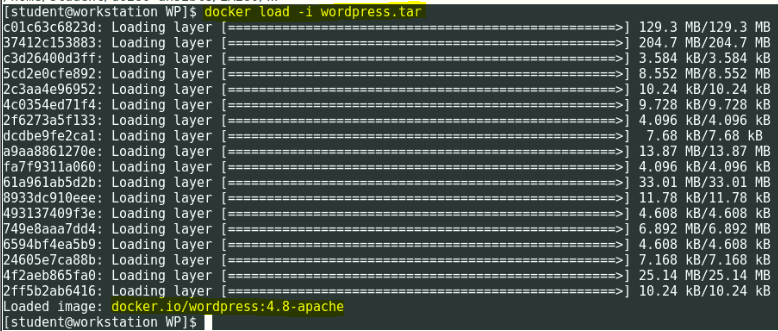
# docker tag fcf3e41b8864 registry.lab.example.com/wordpress:4.8

# docker push fcf3e41b8864 registry.lab.example.com/wordpress:4.8

# docker images

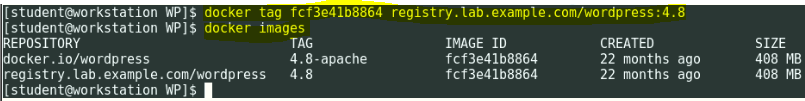
# docker-registry-cli registry.lab.example.com search wordpress ssl

# docker-registry-cli registry.lab.example.com search mysql ssl



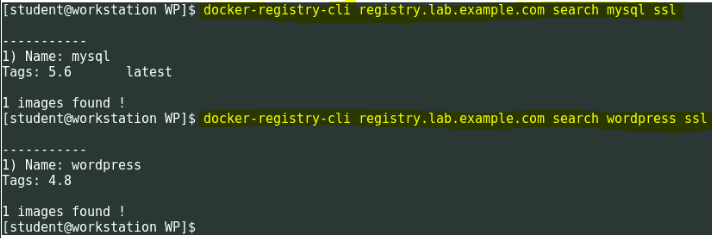


**Tag the word press docker image**



**PUSH the tar image into WordPress into registry.lab.example.com**





**Login into services/storage machine and execute below commands to enable NFS directories for MySQL and WordPress mount points**

# mkdir -p /var/export/OSE\_mysql

# mkdir -p /var/export/OSE\_wordpress

# chown nfsnobody:nfsnobody /var/export/OSE\_mysql

# chown nfsnobody:nfsnobody /var/export/OSE\_wordpress

# echo "/var/export/OSE\_mysql \*(rw,async,all\_squash)" > /etc/exports.d/OSE\_mysql.exports

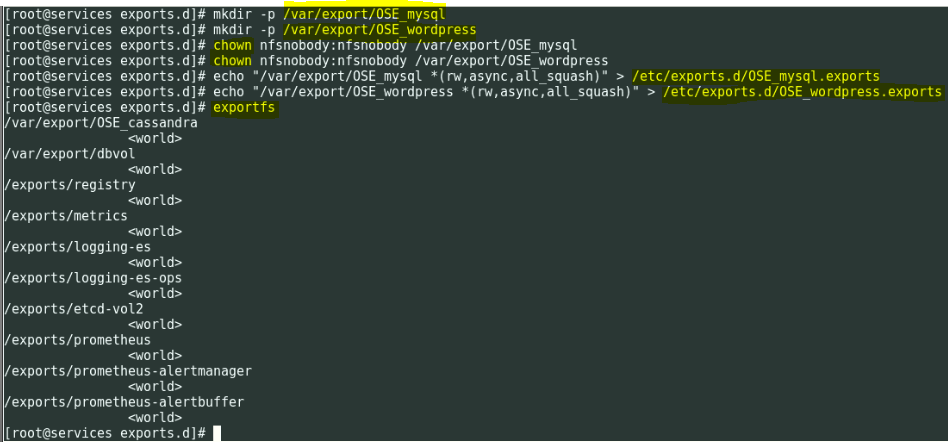
# echo "/var/export/OSE\_wordpress \*(rw,async,all\_squash)" > /etc/exports.d/OSE\_wordpress.exports

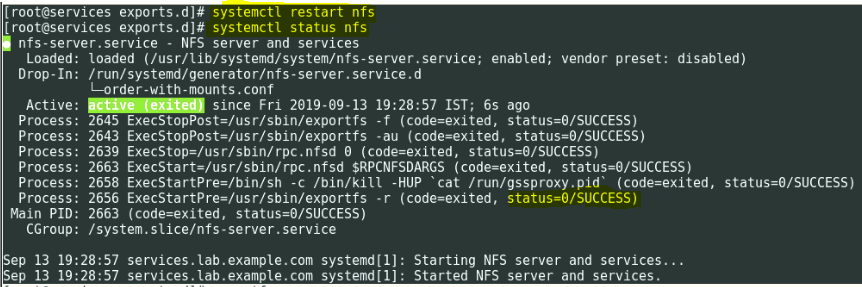
# exportfs

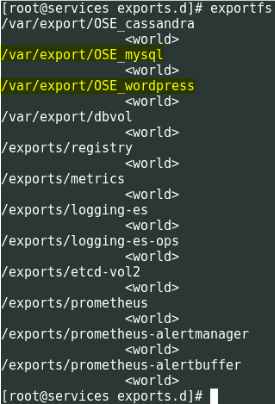
# systemctl restart nfs

# systemctl status nfs

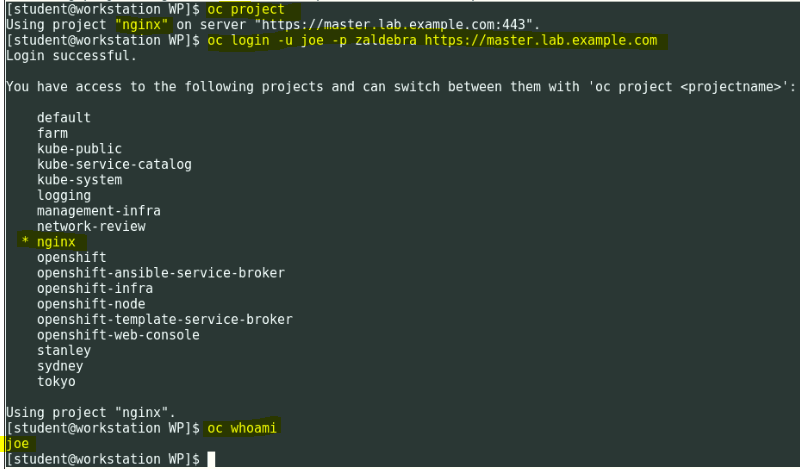
# exportfs







**Login as joe from workstation**



Make sure the active project is "**ngnix**"

**Following are the YML files to be used for PresistentVolumeCreation:**

[student@workstation ~]$ **cat mysql-pv.yml**

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv

spec:

capacity:

storage: 5Gi

accessModes:

- ReadWriteOnce

nfs:

path: /var/export/OSE\_mysql

server: services.lab.example.com

persistentVolumeReclaimPolicy: Recycle

claimRef:

name: claim-mysql

namespace: nginx

[student@workstation ~]$

[student@workstation ~]$ **cat mysql-pvc.yml**

apiVersion: "v1"

kind: "PersistentVolumeClaim"

metadata:

name: "claim-mysql"

spec:

accessModes:

- "ReadWriteOnce"

resources:

requests:

storage: "5Gi"

volumeName: "mysql-pv"

[student@workstation ~]$

[student@workstation ~]$ **cat wordpress-pv.yml**

apiVersion: v1

kind: PersistentVolume

metadata:

name: wordpress-pv

spec:

capacity:

storage: 5Gi

accessModes:

- ReadWriteOnce

nfs:

path: /var/export/OSE\_wordpress

server: services.lab.example.com

persistentVolumeReclaimPolicy: Recycle

claimRef:

name: claim-wordpress

namespace: nginx

[student@workstation ~]$

[student@workstation ~]$ **cat wordpress-pvc.yml**

apiVersion: "v1"

kind: "PersistentVolumeClaim"

metadata:

name: "claim-wordpress"

spec:

accessModes:

- "ReadWriteOnce"

resources:

requests:

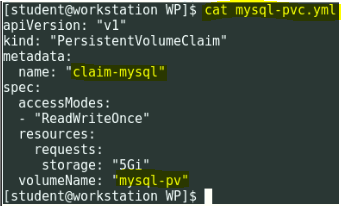
storage: "5Gi"

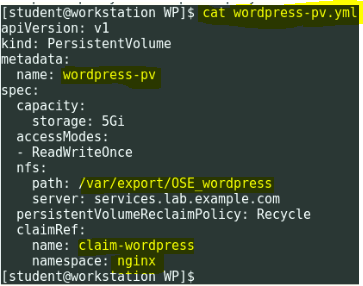
volumeName: "wordpress-pv"

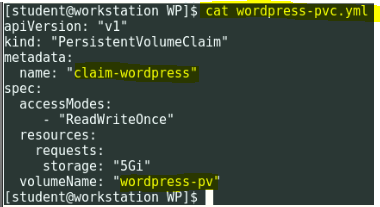
[student@workstation ~]$



**Page No. 320 in Developer Guide: Yaml code will be available**







**Following are the YML files to be used for Service Creation**

[student@workstation ~]$ **cat mysql-service.yml**

apiVersion: v1

kind: Service

metadata:

labels:

name: mysql

name: mysql

spec:

ports:

- name: 3306-mysql

port: 3306

selector:

name: mysql

[student@workstation ~]$

[student@workstation ~]$ **cat wordpress-service.yml**

apiVersion: v1

kind: Service

metadata:

labels:

name: wordpress

name: wordpress

spec:

ports:

- name: 5055-wordpress

port: 5055

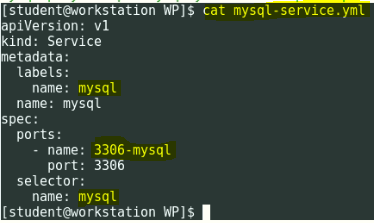
targetPort: wordpress

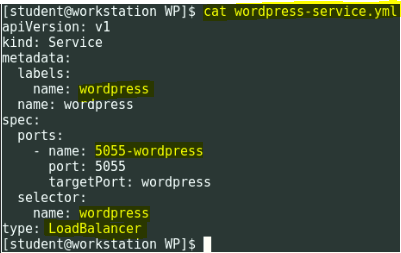
selector:

name: wordpress

type: LoadBalancer

[student@workstation ~]$





**Following are the YML files to be used for POD Creation**

[student@workstation ~]$ **cat pod-mysql.yml**

apiVersion: v1

kind: Pod

metadata:

name: mysql

labels:

name: mysql

spec:

containers:

- image: registry.lab.example.com/rhscl/mysql-57-rhel7:latest

name: mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

value: mysql\_user

- name: MYSQL\_USER

value: wp\_user

- name: MYSQL\_PASSWORD

value: wp\_pass

- name: MYSQL\_DATABASE

value: wp\_db

ports:

- containerPort: 3306

name: mysql

volumeMounts:

- name: mysql-persistent-storage

mountPath: /var/lib/mysql/data

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: claim-mysql

[student@workstation ~]$

[student@workstation ~]$ **cat pod-wordpress.yml**

apiVersion: v1

kind: Pod

metadata:

name: wordpress

labels:

name: wordpress

spec:

containers:

- image: registry.lab.example.com/wordpress:4.8

name: wordpress

env:

- name: WORDPRESS\_DB\_USER

value: wp\_user

- name: WORDPRESS\_DB\_PASSWORD

value: wp\_pass

- name: WORDPRESS\_DB\_NAME

value: wp\_db

- name: WORDPRESS\_DB\_HOST

# this is the name of the mysql service fronting the mysql pod in the same namespace

# expands to mysql.<namespace>.svc.cluster.local - where <namespace> is the current namespace

value: mysql

ports:

- containerPort: 80

name: wordpress

volumeMounts:

- name: wordpress-persistent-storage

mountPath: /var/www/html

serviceAccountName: wordpress

volumes:

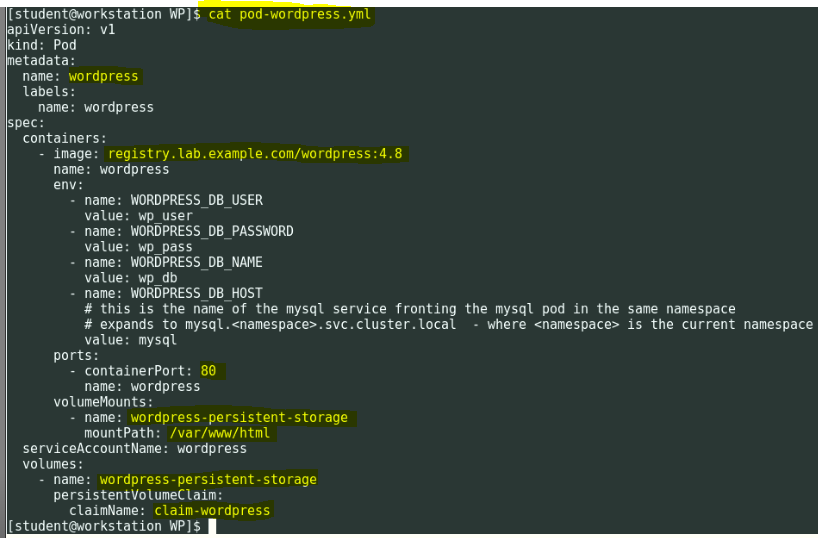
- name: wordpress-persistent-storage

persistentVolumeClaim:

claimName: claim-wordpress

[student@workstation ~]$





**Now, Create PV and PVCs**

**Login as admin:**

# oc login –u admin –p redhat <https://master.lab.example.com>

# oc whoami

# oc create -f mysql-pv.yml

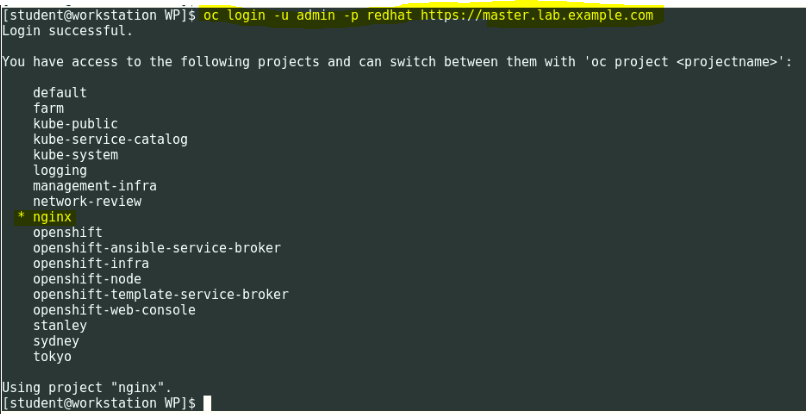
# oc create -f wordpress-pv.yml

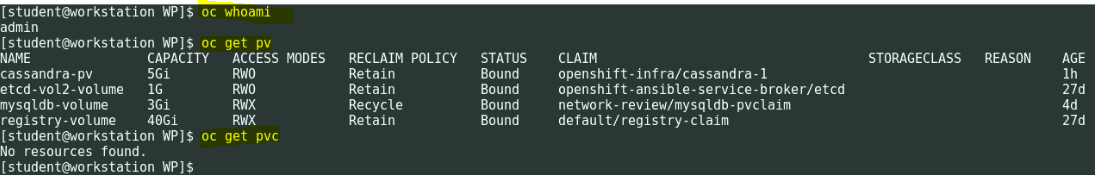
# oc get pv

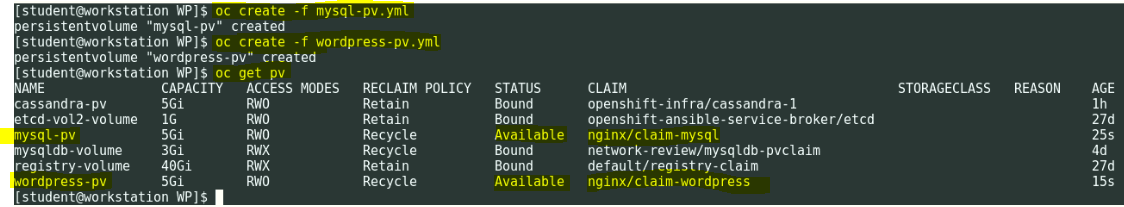
# oc create -f mysql-pvc.yml

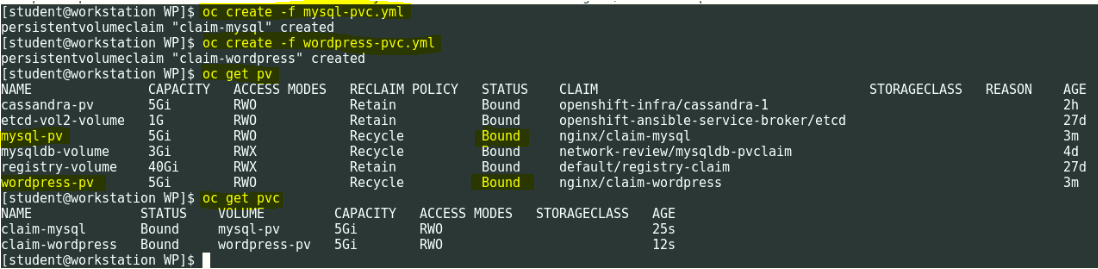
# oc create -f wordpress-pvc.yml

# oc get pvc









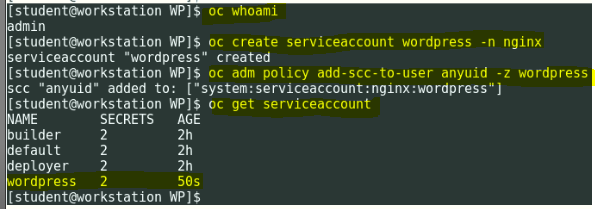
**Now, create a service account by logging as admin:**

# oc whoami

# oc create serviceaccount wordpress -n nginx

# oc adm policy add-scc-to-user anyuid -z wordpress

# oc get serviceaccount

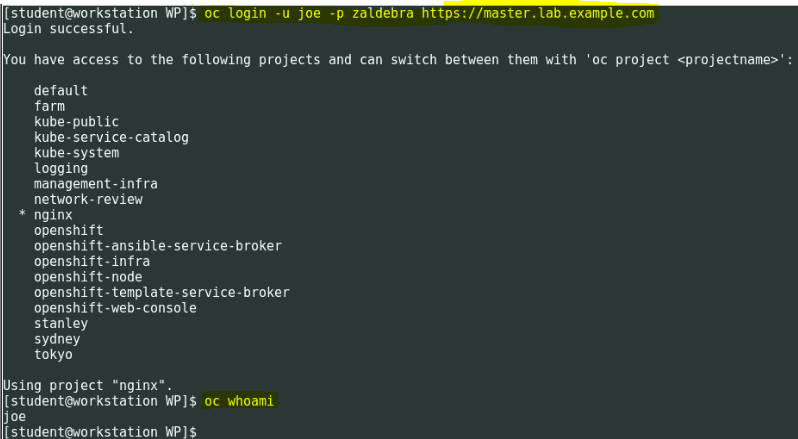


**Login as Joe to create the wordpress app:**

# oc login –u joe –p redhat <https://master.lab.example.com>

# oc get project

# oc get svc



**Now, create the service for mysql and wordpress**

# oc get svc

# oc create -f mysql-service.yml

# oc create -f wordpress-service.yml

# oc get svc



**Now, create mysql and wordpress PODs**

# oc get pod

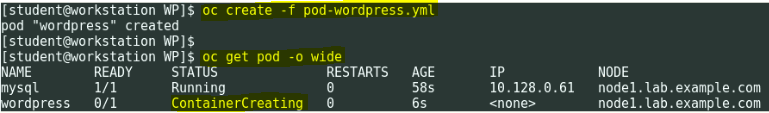
# oc create -f pod-mysql.yml

# oc create -f pod-wordpress.yml

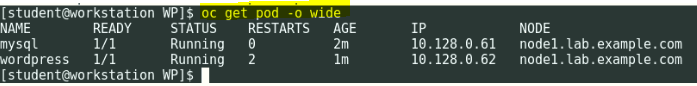
# oc get pod -o wide



**Create Wordpress POD**



**Wait for 1 min till mysql POD READY state is 1/1**

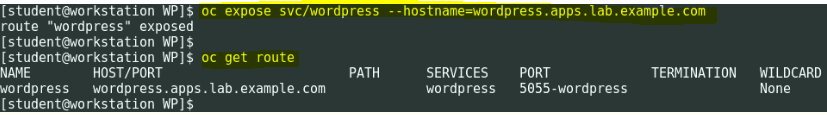


**Now create the route for the wordpress application**

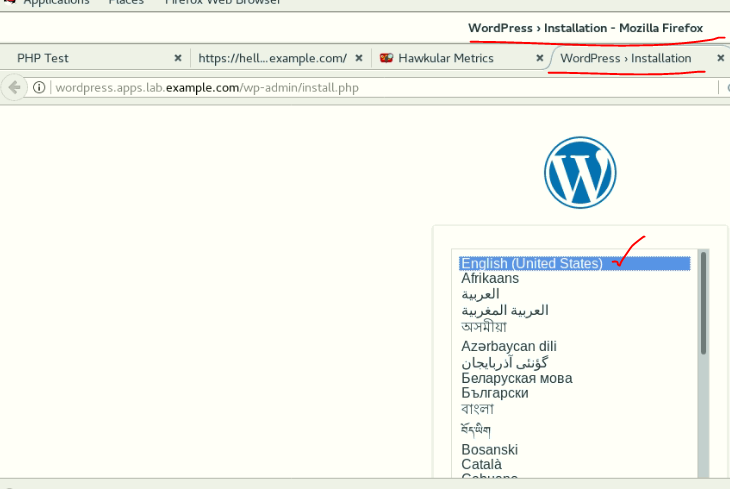
# oc get route

# oc expose svc/wordpress --hostname=wordpress.apps.lab.example.com

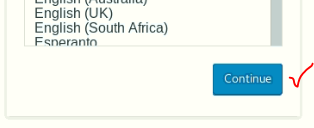
# oc get route



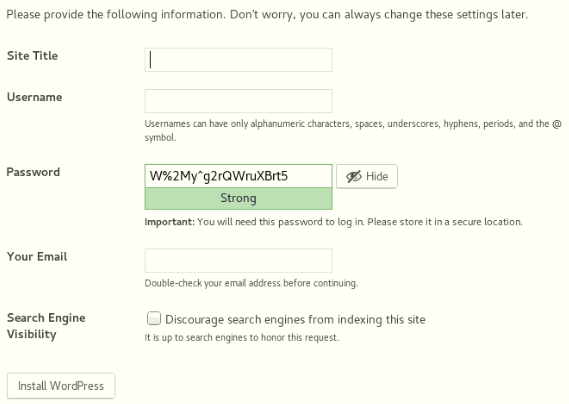
Now, from the workstation GUI open Firefox browser and type <http://wordpress.apps.lab.example.com> and complete the wordpress installation.



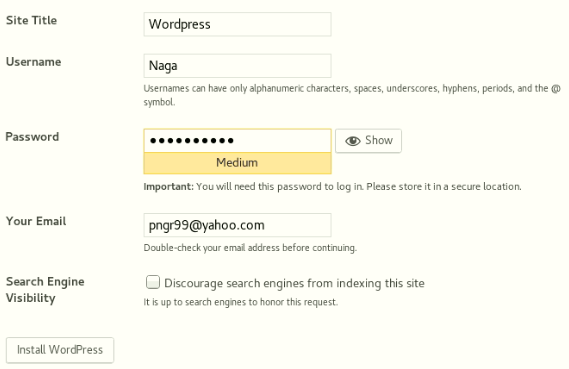
**Select “English (United States) and Scroll down and click on continue.**

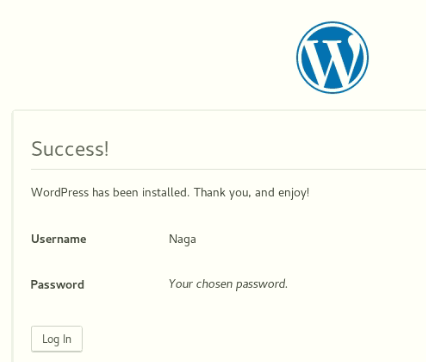


Update the below details and click on install wordpress.



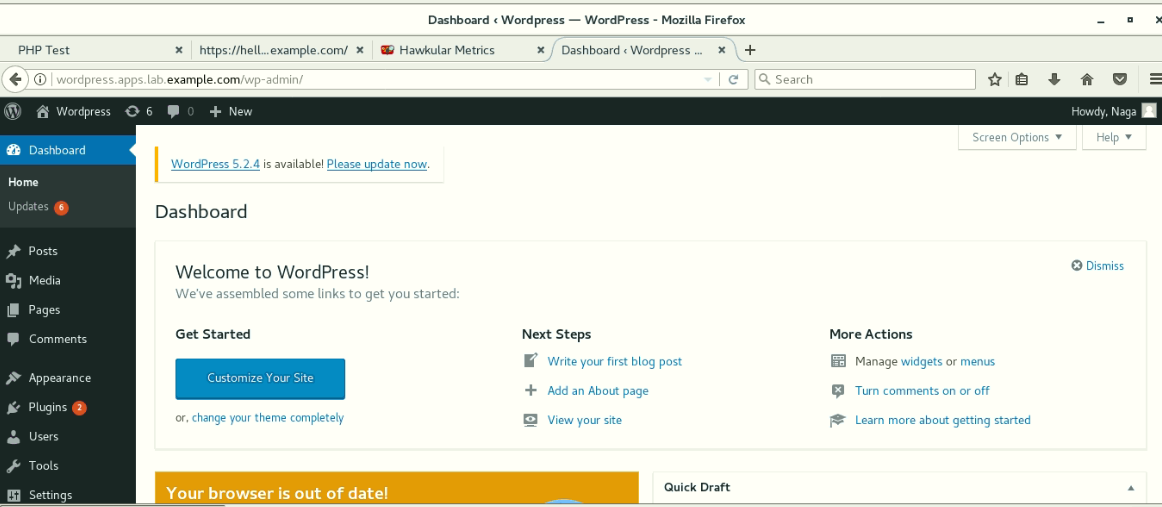
**See the below details..**





**Above you created a user, using same credentials you login. You will be able to see the wordpress dashboard.**





##################################################################################

**Question 10: Create GOGS APP Site**

Create gogs application from template gogs-template.yaml

a) Change the git url to <http://classroom.example.com/git/gogs>

b) Postgres database should be using imagestream postgres:9.5

c) Application should be accessible at <http://template.apps.realX.example.com>

##################################################################################

**Solution-10:**

# docker images

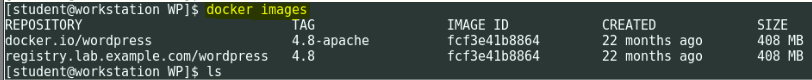
# docker load –i gogs\_latest.tar

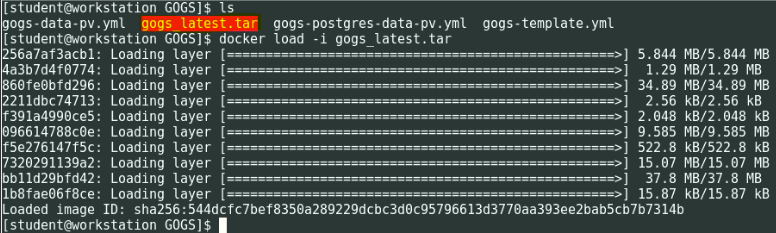
# docker tag 544dcfc7bef8 registry.lab.example.com/gogs:9.0.7

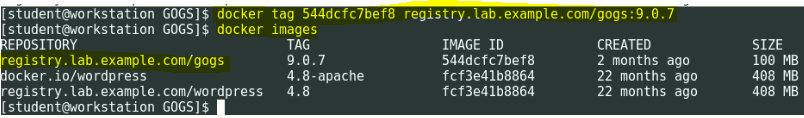
# docker push registry.lab.example.com/gogs:9.0.7

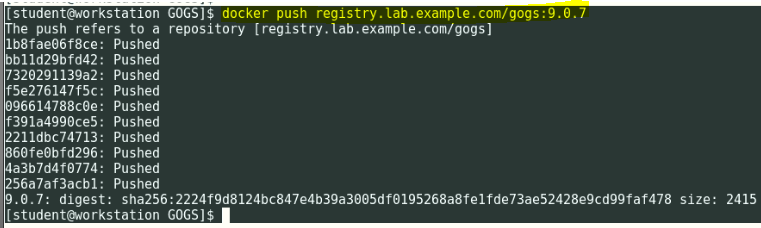
# docker images

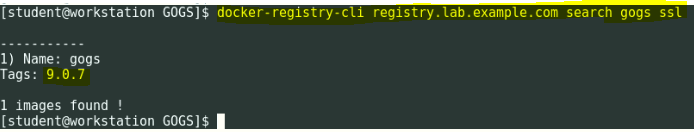
# docker-registry-cli registry.lab.example.com search gogs ssl











**Login into services/storage machine and execute below commands to enable NFS directories for GOGS and Postgres mount points**

# mkdir -p /var/export/OSE\_gogs

# mkdir -p /var/export/OSE\_postgres

# chown nfsnobody:nfsnobody /var/export/OSE\_gogs

# chown nfsnobody:nfsnobody /var/export/OSE\_postgres

# echo "/var/export/OSE\_gogs \*(rw,async,all\_squash)" > /etc/exports.d/OSE\_gogs.exports

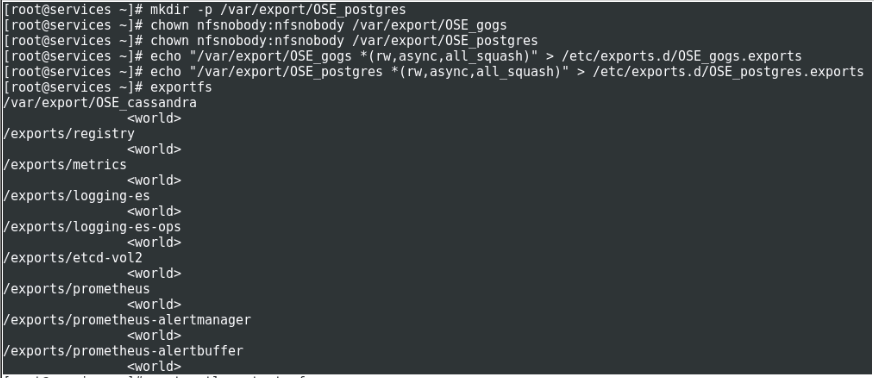
# echo "/var/export/OSE\_postgres \*(rw,async,all\_squash)" > /etc/exports.d/OSE\_postgres.exports

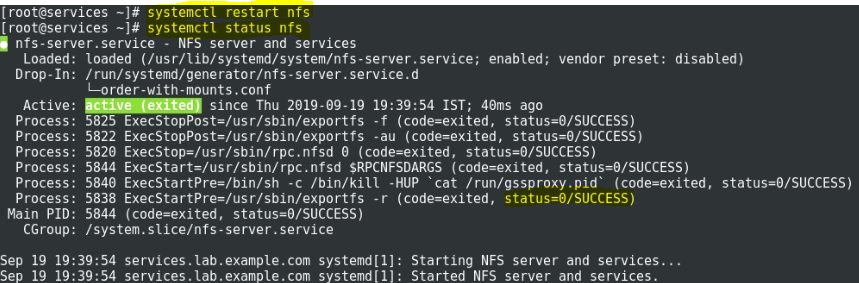
# exportfs

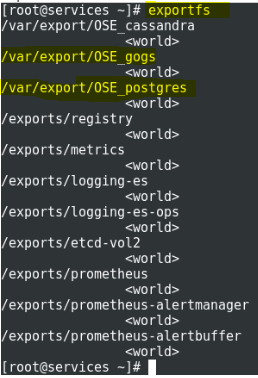
# systemctl restart nfs

# systemctl status nfs

# exportfs







Please refer PDF attachment. 

**Delete commands:**

oc delete svc gogs

oc delete svc gogs-postgresql

oc delete serviceaccount gogs

oc delete route gogs

oc delete dc gogs

oc delete dc gogs-postgresql

oc delete imagestream gogs

oc delete pvc gogs-data

oc delete pvc gogs-postgres-data

oc delete configmap gogs-config

oc delete pv gogs-data-pv

oc delete pv gogs-postgres-data-pv

oc create -f gogs-postgres-data-pv.yml

oc create -f gogs-data-pv.yml