Indian Institute of Technology
Bombay

Micro-Cloud Setup Using OpenShift

Principal Investigator
Prof. S. Sudarshan

Project In-Charge
Mr. Nagesh Karmali

Project Team

Member1	**Member2**	**Member3**
Mr. Suraj Goel	Mr. Saurabh Vijay	Ms. Anzum Bano

Summer Internship 2019

Project Approval Certificate

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

The project entitled submitted by Mr. Suraj Goel, Mr. Saurabh Vijay, Ms. Anzum Bano is approved for Summer Internship 2019 programme from 20th May 2019 to 9th July 2019, at Department of Computer Science and Engineering, IIT Bombay.

Prof. S. Sudarshan
Dept. of CSE, IIT Bombay
Principle Investigator

Mr. Nagesh Karmali
Sr. Project Manager (Research)
Dept. of CSE, IIT Bombay

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Suraj Goel
MNNIT Allahabad

Saurabh Vijay
MNNIT Allahabad

Anzum Bano
MNNIT Allahabad

Date

ACKNOWLEDGEMENT

We, the summer interns of this group, are overwhelmed in all humbleness and gratefulness to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection and have assigned tasks well above the level of simplicity and into something concrete and unique. We, wholeheartedly thank Mr. Nagesh Karmali for providing us the opportunity to work on this project.He was and is always there to show us the right track when needed help. With help of his brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned.Our sincere thanks to Mrs. Anjali , Mr. Gaurav Patil and Mr. Gaurav Ojha without whom the project would not have been successful . During the process, we got a chance to see the stronger side of our technical and nontechnical aspects and also strengthen our concepts. Last but not the least, we wholeheartedly thank all our other colleagues working in different projects under Prof. S.Sudarshan for helping us evolve better with their critical advice.

ABSTRACT

Micro-cloud Setup using openshift is aimed to provide an open source cloud-based user-friendly platform, which can be used to create, test, and run applications, and finally deploy them on cloud. Moreover, this project is capable of managing applications smoothly, written in various languages, such as Node.js, Ruby, Python, Perl, and Java.
One of the key features of this project, being focussed on OpenShift is, it is extensible, which helps the users support the application written in other languages. additionally, this project provides a common platform for enterprise units to host their applications on cloud without worrying about the underlying operating system. This makes it very easy to use, develop, and deploy applications on cloud. One of the key features is, it provides managed hardware and network resources for all kinds of development and testing. With OpenShift,this project gives PaaS developer the complete freedom to design their required environment with specification..

Contents

Chapter 1

INTRODUCTION

Micro-cloud setup using Openshift technology is the project undertaken by Eklavya summer internship program at IIT Bombay, which focuses on OpenShift Container Platform a microservices-based architecture of smaller, decoupled units that work together.

It runs on top of a Kubernetes cluster, with data about the objects stored in etcd, a reliable clustered key-value store.

The Docker service provides the abstraction for packaging and creating Linux-based, lightweight container images and the Kubernetes provides the cluster management and orchestrates containers on multiple hosts.

OpenShift Container Platform adds:

- Source code management, builds, and deployments for developers.
- Autoscaling and Automatic load balancing in an application.
- Managing and promoting images at scale as they flow through your system.
- Team and user tracking for organizing a large developer organization.

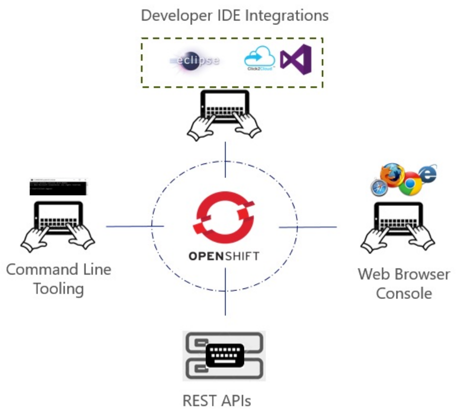


Fig.1:OpenShift Layout

1.1 Purpose

- To provide a platform so that both Developers and Operators can work together without having to sacrifice their individual concerns.
- Since deploying and managing containers at scale is a complicated process, OpenShift enables efficient container orchestration, allowing quicker container provisioning, deploying, scaling, and management.
- To allow applications to migrate their container processes to the new operating system quickly, while avoiding the extensive costs often involved in migrating.
- To automate the process of building new container images for all of your users. So that standard Docker builds based on the Dockerfiles manually provided can be run, and also provide a Source-to-Image feature which allows to specify the source from which image is to be generated.

1.2 Scope

The target audience of our project is not limited to any particular group or system rather the module(project) is compatible with every web application. This module can be used in context with any web application for creating, deleting ,managing, deploying etc.

Chapter 2

Motivation

- With OpenShift, developers have access to a self-service platform that allows them to create, modify, and deploy applications on demand with the click of a button .

- applications should be incredibly portable and hyper scalable.OpenShift's orchestration layer, Googles Kubernetes, automates the scheduling and replication of these containers meaning that they're highly available and able to accommodate whatever your users can throw at it.

- Data should be protected from any type of harm.OpenShift is built on those same principles and applications running on OpenShift have their own "container"allowing for the code and data to separated from each other by default.

- IT organizations need a vendor that can enable them through the entire stack, not just one aspect of it. Having a PaaS environment thats coupled together with supported IaaS services and middleware services, means better agility and interoperability.

- Operations needs to be able to maintain those applications easily and have them run at the appropriate scale. OpenShift enables them to do that with little to no manual intervention

2.1 Objective

Setting up Micro-cloud architecture with commodity storage and server nodes using OpenShift(PaaS). OpenShift will provide infrastructure for orchestration of dockerized images of applications like Open edX/ IIT-BombayX/Moodle/Drupal.

- Giving access to developer to create , modify and deploy applications on demand with the click of a button .
- To increase portability of applications by containerizing phenomena .
- To Make applications more scalable so that they can be scaled according to the traffic of users
- To make applications more reliable and secure by containerize code and data separately .
- Single click installation of applications despite of any kind of OS .

2.2 Technologies Used

- **Openshift** : OpenShift is an open source hybrid cloud application Platform as a Service (PaaS) .
- **Docker** : Docker is an open source software platform to create, deploy and manage virtualized application containers on a common operating system (OS), with an ecosystem of allied tools.
- **Kubernetes** : Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications .
- **Shell Scripting** : A shell script is a text file that contains a sequence of commands for a UNIX-based operating system.
- **MySql** : It is used for provisioning different services and to provide database for different services .

- **Git** : Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git repositories contain source code for different application and services .

Chapter 3

Open edX Platform

3.1 Open edX Versions

- **Ginko**
- **Hawthorn**
- **Ironwood**

3.2 Types Of Installations

- **Native Installation**
  The Native installation installs the Open edX software on your own Linux machine in a production-like configuration. Details are at the Open edX Native Installation page on the edX wiki.
  Installation Wiki
- **Docker Based Installation**
  The Docker Based Installation is done using Open edX Devstack. Devstack is a deployment of the Open edX platform within a set of Docker containers designed for local development. Running the Open edX platform locally allows you to discover and fix system configuration issues early in development.

3.3 openedX Components

Few key Components of openedX:

- **Learning Management System (LMS)**
  The LMS is the most visible part of the Open edX project. Learners take courses using the LMS. The LMS also provides an instructor dashboard that users who have the Admin or Staff role can access by selecting Instructor.
  The LMS uses a number of data stores. Courses are stored in MongoDB, with videos served from YouTube or Amazon S3. Per-learner data is stored in MySQL.
- **Studio**
  Studio is the course authoring environment. Course teams use it to create and update courses. Studio writes its courses to the same Mongo database that the LMS uses.
- **Forum**
  The LMS uses an API provided by the comments service to integrate discussions into the learners course experience.
- **Elastic Search**
  The Open edX project uses Elasticsearch for searching in multiple contexts, including course search and the comments service.

3.4 Steps Of Installation

The following steps installs the ironwood-release of devstack

- git clone https://github.com/edx/devstack
- cd devstack
- git checkout open-release/ironwood.master
- export OPENEDX RELEASE=ironwood.master
- make dev.checkout
- make dev.clone
- make dev.provision

3.5 Errors Faced In Installation

- **./provision.sh: line 21: /usr/local/bin/docker-compose: Permission denied**
  Makefile:59: recipe for target 'dev.provision.run' failed
  make: *** [dev.provision.run] Error 126
  Solution :
  $sudo - i$
  curl -L
  https://github.com/docker/compose/releases/download/1.18.0/docker-compose-‘uname -s‘-‘uname -m‘ -o /usr/local/bin/docker-compose
  $chmod 755/usr/local/bin/docker - compose$
  exit

- **TASK [common : Update expired apt keys]** **************
  This error occurs due to the fetching to apt-keys from an expired link.
  Solution :
  1.sudo exec -it edx.devstack.lms bash
  2.cd app/edx_ansible/edx_ansible/playbooks/roles/common_vars/defaults/
  3.vi main.yml
  4.On entering the main.yml file in vim editor change the link for the keyword COMMON_EDX_PPA_KEY from keyserver.ubuntu.com to hkp://keyserver.ubuntu.com:80

3.6 Docker Containers after Installation

No.	Container Name	Docker Image
1	edx.devstack.chrome	edxops/chrome:ironwood.master
2	edx.devstack.elasticsearch	edxops/elasticsearch:devstack
3	edx.devstack.firefox	edxops/firefox:ironwood.master
4	edx.devstack.memcached	memcached.devstack.edx
5	edx.devstack.mongo	mongo:3.2.16
6	edx.devstack.mysql	mysql:5.6
7	edx.devstack.credentials	edxops/credentials:ironwood.master
8	edx.devstack.discovery	edxops/discovery:ironwood.master
9	edx.devstack.ecommerce	edxops/ecommerce:ironwood.master
10	edx.devstack.lms	edxops/edxapp:ironwood.master
11	edx.devstack.edx_notes_api	edxops/notes:ironwood.master
12	edx.devstack.studio	edxops/edxapp:ironwood.master
13	edx.devstack.forum	edxops/forum:ironwood.master
14	edx.devstack.devpi	edxops/devpi:ironwood.master
15	edx.devstack.gradebook	node:10

Table 3.1: Containers

Follow below links for more information:
Installation Steps
Installation Explanation

Chapter 4

Setting Up Openshift on Commodity Servers

4.1 OpenShift-Fundamentals

OpenShift is an open source hybrid cloud application Platform as a Service (PaaS). Community version of openshift is OKD. Major Components of Openshift are :-

1. **Docker** : Docker is an open-source software tool designed to automate and ease the process of creating, packaging, and deploying applications using an environment called a container.

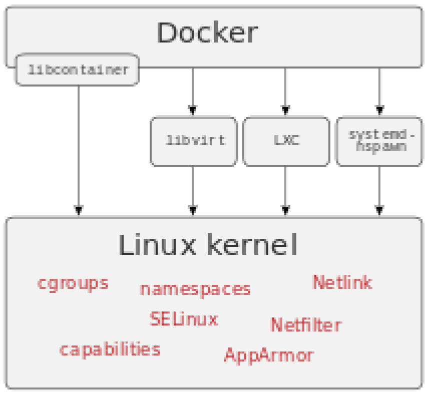


Fig. 2:Docker

Docker Terminology:

- **Docker Container**: A Docker Container encapsulates a Docker image and when live and running, is considered a container. Each container runs isolated in the host machine.

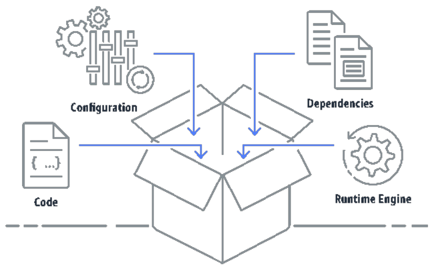


Fig. 3:Container

- **Docker Image**: A Docker Image is the basic unit for deploying a Docker container. A Docker image is essentially a static snapshot of a container, incorporating all of the objects needed to run a container.
- **Containerization** : Containerization is a lightweight alternative to full machine virtualization (like VMWare) that involves encapsulating an application within a container with its own operating environment.

- **Docker Registry**:The Docker Registry is a stateless, highly scalable server-side application that stores and distributes Docker images.

- **Docker Engine**: The Docker Engine is a layer which exists between containers and the Linux kernel and runs the containers.

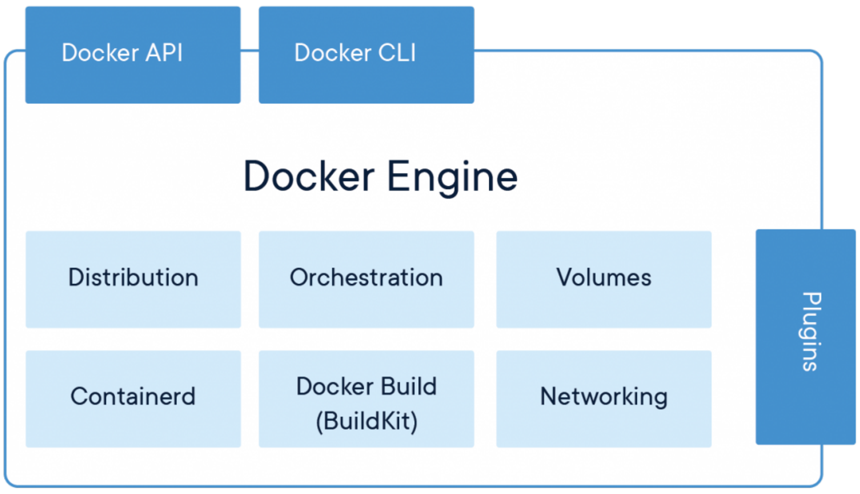


Fig. 4:Docker Engine

- **Dockerfiles**: Dockerfiles are merely text documents (.yaml files) that contains all of the configuration information and commands needed to assemble a container image.
- **Docker Compose**: Docker Compose is a tool that defines, manages and controls multi-container Docker applications. With Compose, a single configuration file is used to set up all of your applications services .

2. **Kubernetes**:Kubernetes is a container-orchestration tool which provides a smart way for orchestration in setting a multi-node cluster.Kubernetes comes with a lot of functionalities such as load balancing, pod scaling, process scheduling and process management.

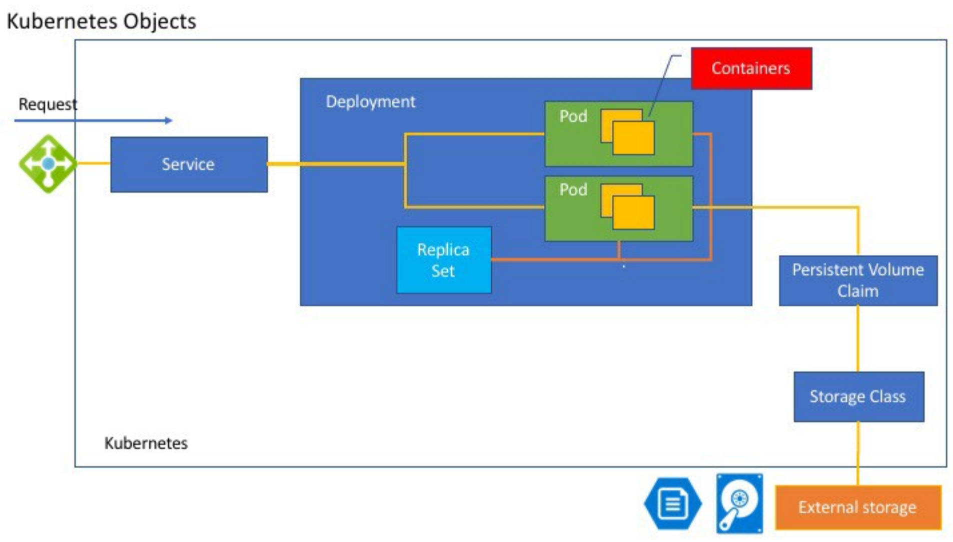


Fig. 5:Kubernetes Architecture

Kubernetes Terminology:

- **Pods** : The most fundamental unit in a kubernetes cluster is a pod. A pod is a collection of container and volumes. Each pod has an IP address.

- **Deployment** : A deployment can be seen as a stateless state of the pod. A deployment is used to provide pod definition and rolling updates to the pod. A deployment contains information such as number of replicas, image, volume .

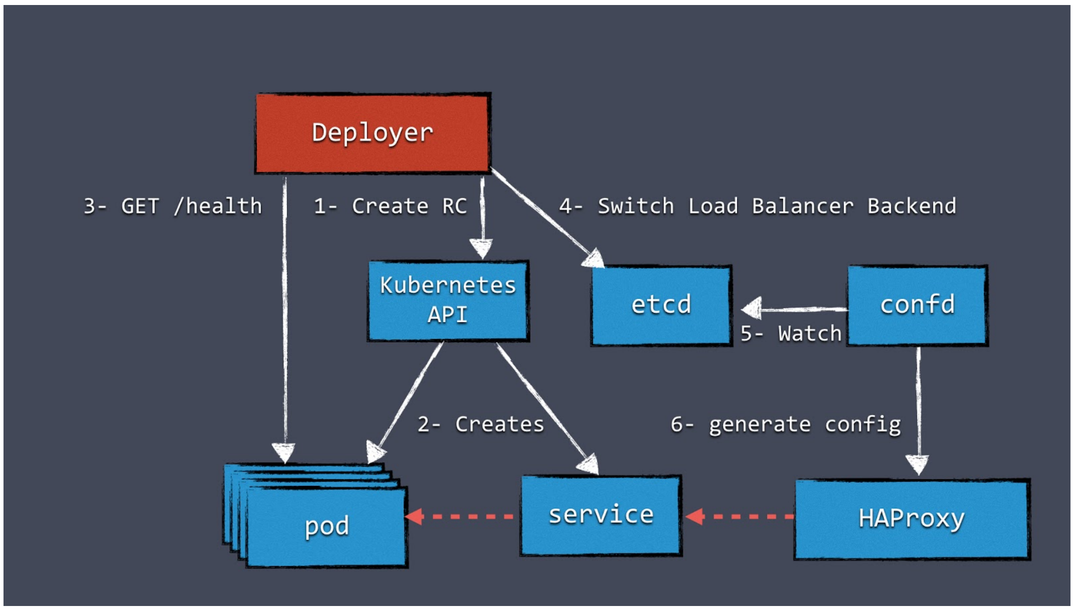


Fig. 6:Kubernetes Architecture

- **Service** : A service exposes a deployment as a network service. A deployment is stateless whereas a service can be considered as a stateful definition. The three types of services in kubernetes are : ClusterIP, NodePort and LoadBalancer.

- **Ingress Network**: An ingress network exposes the services in the cluster to the outside network i.e. the clients.

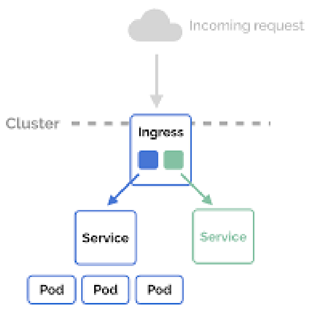


Fig. 7:Ingress

4.2 Openshift Architecture

OpenShift uses the kubernetes master/node architecture as a starting point .

Apart from k8s architecture, OpenShift has the following parts:

- **Routing layer**: The routing layer is a software load balancer.When an application is deployed, a DNS entry is created for it automatically.

- **Integrated container registry**:An image registry is a central location that can serve container images to multiple locations.In addition to providing tightly integrated image access,Openshift works to provide more efficiency.

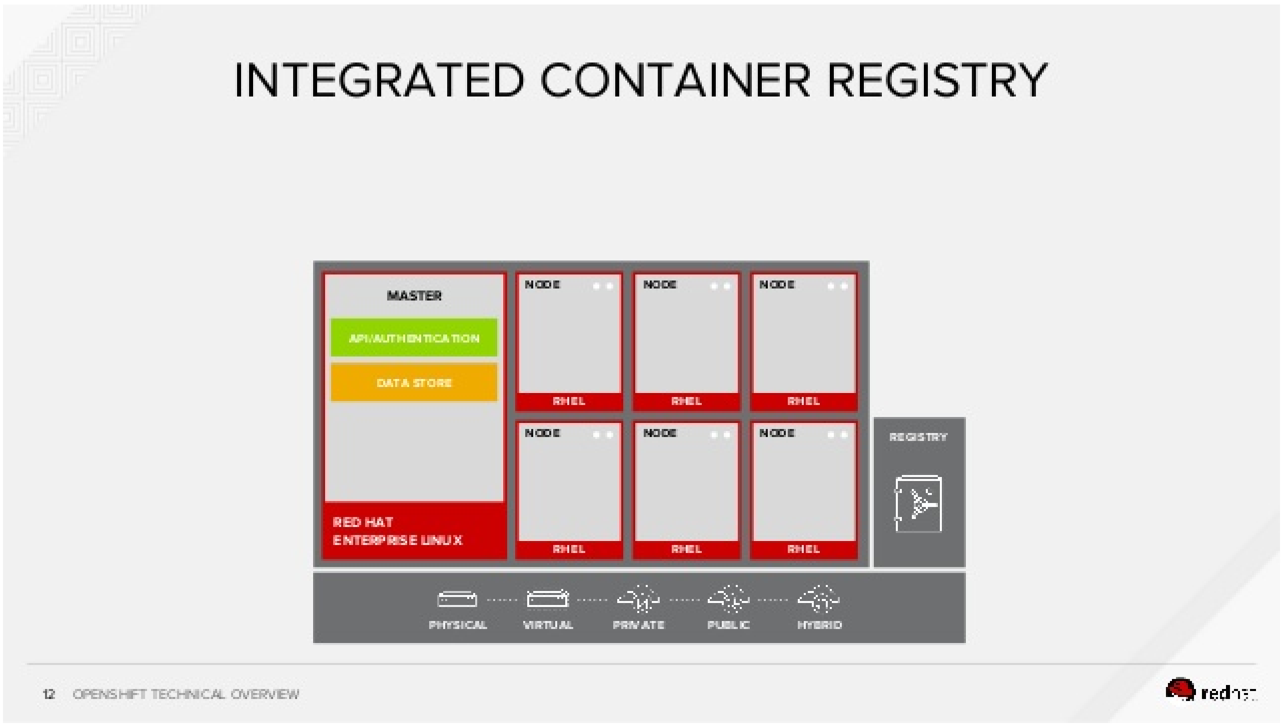


Fig. 8:Integrated Container Registry

Openshift Architecture OverView

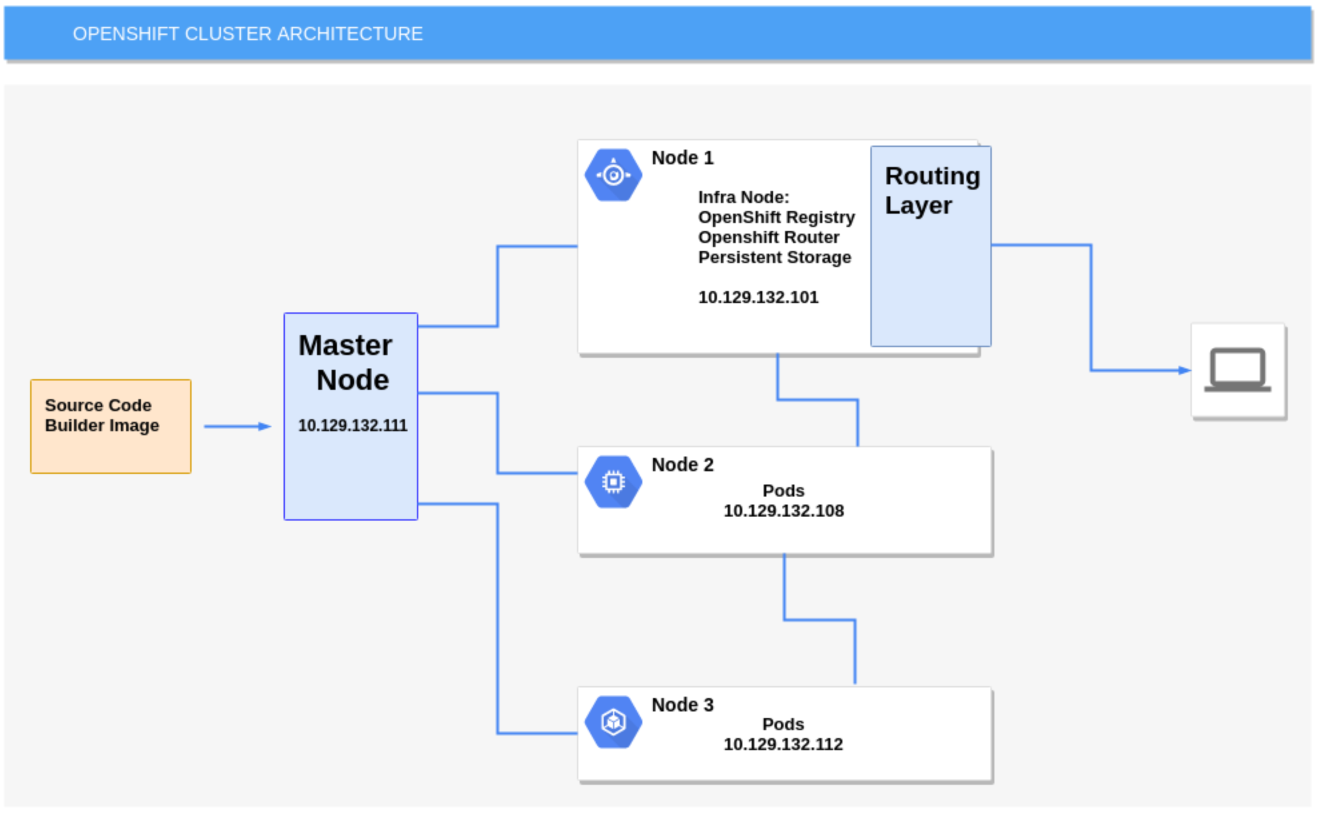


Fig. 9:Openshift cluster (Diagram Made using draw.io)

4.3 Openshift Application Components

:

- **Custom container images**:Each application deployment in OpenShift creates a custom container image to serve your application. This image is created using the applications source code and a custom base image called a builder image.

  Image streams:Image streams are used to automate actions in Openshift. They consist of links to one or more container images. Using image streams, you can monitor applications and trigger new deployments when their components are updated.
  Application pods:OpenShift Online leverages the Kubernetes concept of a pod, which is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed.
  Build configs:A build config contains all the information needed to build an application using its source code. This includes all the information required to build the application container image:

  – URL for the application source code
  – Name of the builder image to use
  – Name of the application container image thats created
  – Events that can trigger a new build to occur

  DeploymentConfigs:If an application is never deployed, it can never do its job. The job of deploying and upgrading the application is handled by the deploymentconfigs component.

  – Currently deployed version of the application.
  – Number of replicas to maintain for the application.
  – Trigger events that can trigger a redeployment. By default, configuration.
  – changes to the deployment or changes to the container image trigger an automatic application. redeployment
  – Upgrade strategy. app-cli uses the default rolling-upgrade strategy.

Deployments:A deployment represents a unique version of an application. Each deployment references a version of the application image that was created, and creates the replication controller to create and maintain the pods to serve the application.

Services:A service uses the labels applied to pods when theyre created to keep track of all pods associated with a given application.

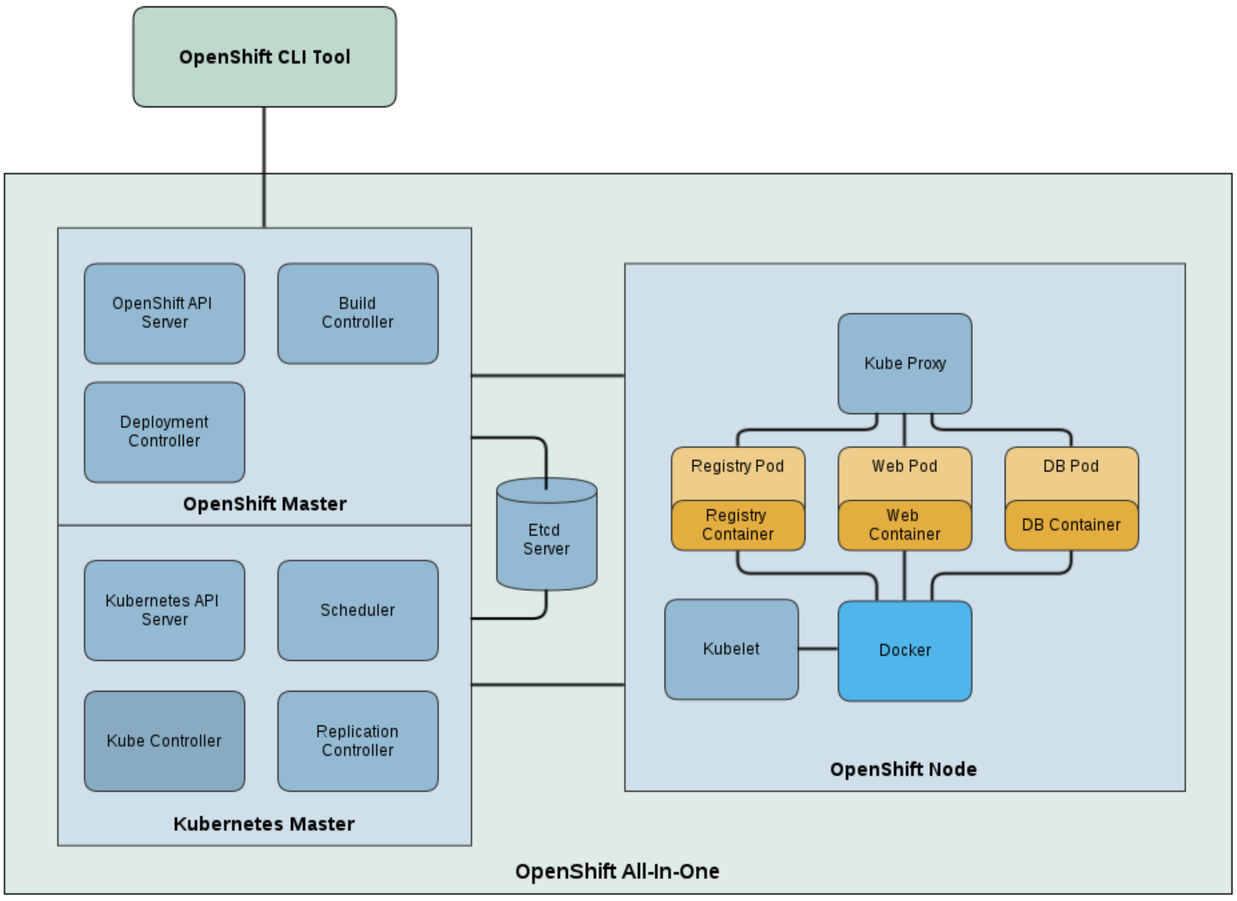


Fig. 10:Application Components

Openshift application components working together

4.4 Openshift Installation

PreInstallation Step - 4 systems are required out of which 1 is acting as master, 1 as infranode and other 2 as nodes. Configuration of these machines are:-

Host Name	IP Address	CPUs	RAM
master.cse.iitb.ac.in	10.129.132.111	4	16GB
node1.cse.iitb.ac.in	10.129.132.101	4	8GB
node2.cse.iitb.ac.in	10.129.132.108	4	8GB
node3.cse.iitb.ac.in	10.129.132.112	4	2GB

Table 4.1: Nodes

Installation Steps:

Step 1: Set the hostname:
hostnamectl set-hostname master.cse.iitb.ac.in (For master node)

Step 2: Configure /etc/hosts file for name resolution on all nodes

Step 3: Update System on all node
yum update -y

Step 4: Install the following Packages on all nodes:

yum install -y wget git nano net-tools docker-1.13.1 bind-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct openssl-devel httpd-tools NetworkManager python-cryptography python-devel python-passlib java-1.8.0-openjdk-headless "@Development Tools"

Step 5:Configure Ansible Repository on master Node only, in etc/yum.repos.d/ansible.repo

Step 6:Start and Enable NetworkManager and Docker Services on all nodes

Step 7:Install Ansible Package and Clone Openshift-Ansible Git Repo on Master Machine

Step 8:Generate SSH Keys on Master Node and install it on all nodes

Step 9:Now Create Your Own Inventory file for Ansible as following on master Node

Step 10:Use the ansible playbook command to check the prerequisites to deply OpenShift Cluster on master Node

Step 11:use the ansible playbook to Deploy OpenShift Cluster on master Node

installation Document.

Openshift web console

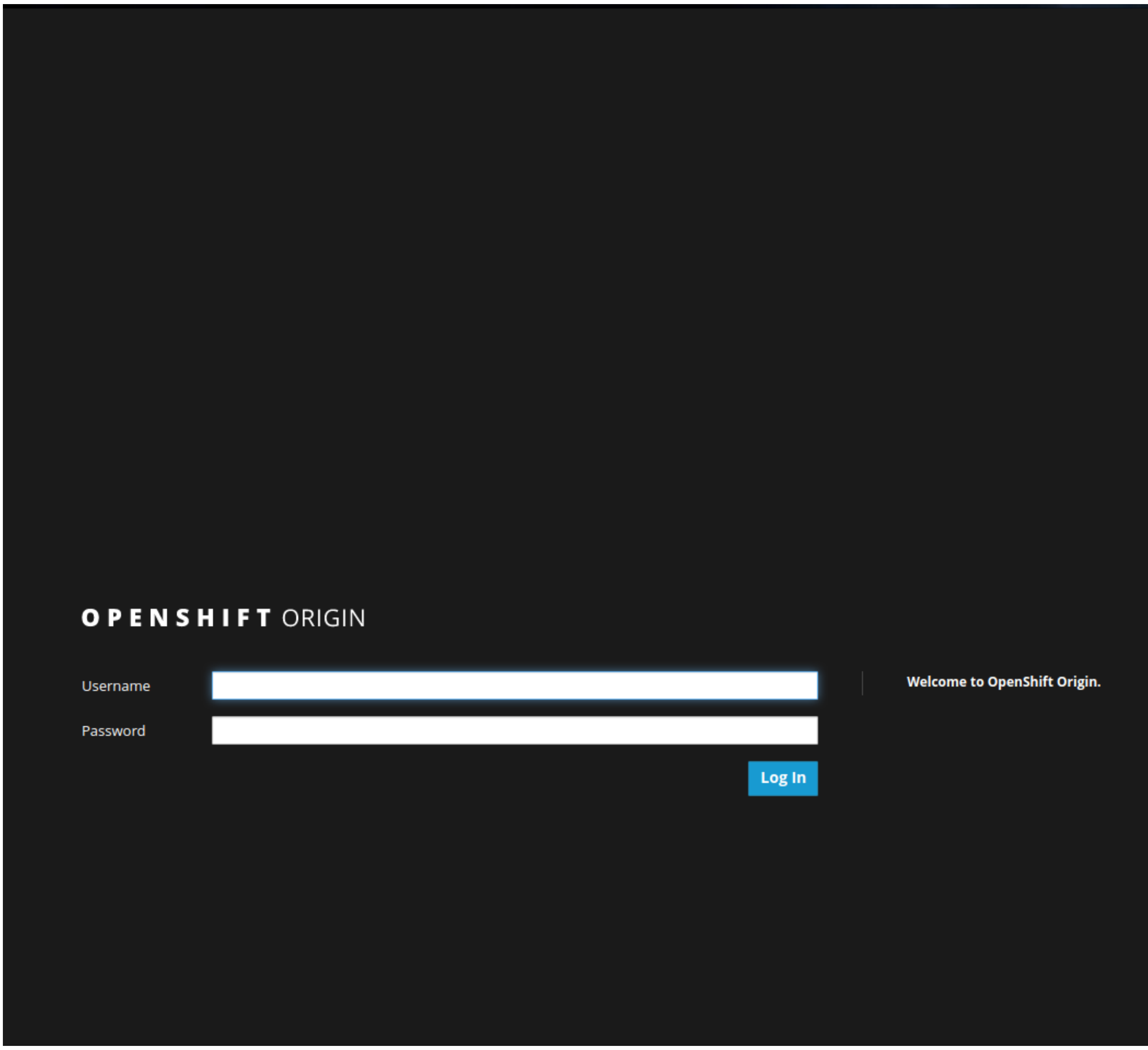


Fig. 11 **Openshift dashboard**

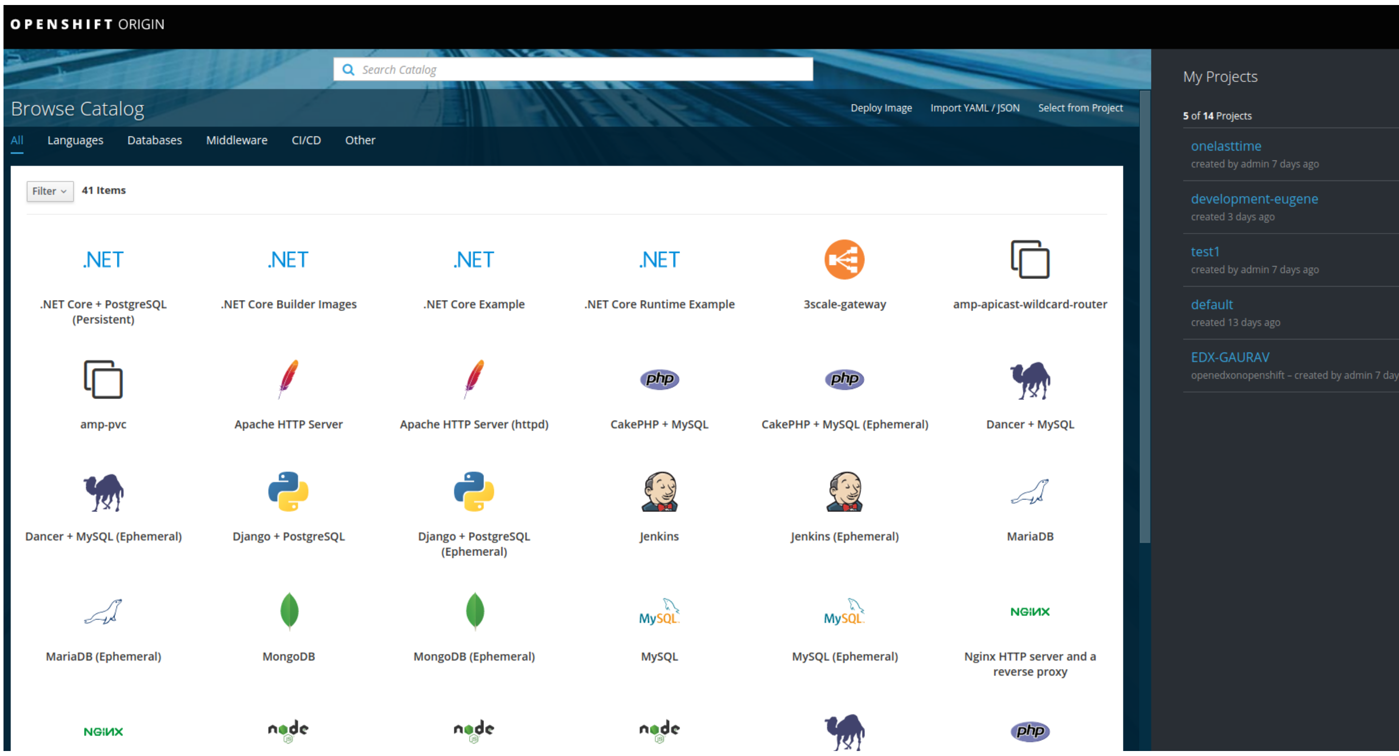


Fig. 12:OpenShift web console

Chapter 5

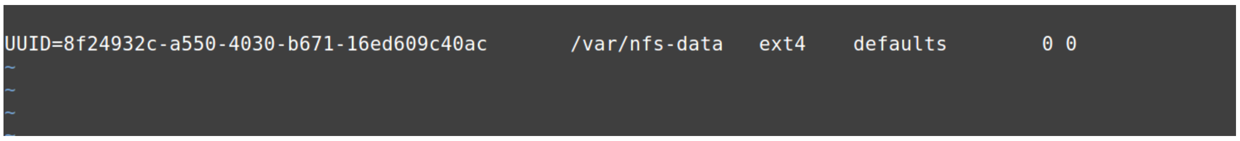
Devstack Installation on Openshift

5.1 Introduction

Devstack is the docker based installation of openedX.This chapter deals with the process of deployment of various services of openedX on openShift Cluster

5.2 Setting up a persistent storage source

1. **Installing the NFS server software**
   - yum -y install nfs-utils
2. **Configuring storage for NFS**
   - **To see all the block devices on master server**
     # lsblk
   - **Creating a filesystem on your storage disk**
     # mkfs.ext4 /dev/sda2
3. **Mounting your storage disk at startup**
   - **Creating a mountpoint directory**
     # mkdir /var/nfs-data
   - **Getting your storage drives block ID**
     # blkid
   - **Editing /etc/fstab to include your volume**



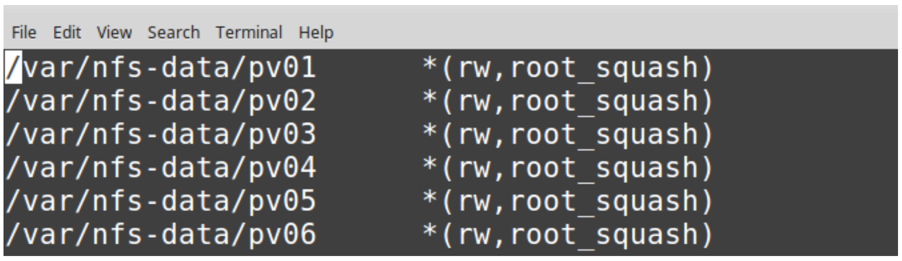
```
UUID=8f24932c-a550-4030-b671-16ed609c40ac       /var/nfs-data   ext4    defaults        0 0
~
~
~
~
```

Fig. 13:/etc/fstab file

- **Activating your new mount point**
  # mount -a
  # mount

4. **Configuring NFS**

- **We'll need to export twenty five different NFS volumes.**
  # mkdir -p /var/nfs-data/{pv01,pv02,pv03,pv04,pv05..}
- **Edit /etc/exports configration file to add all the volumes.**



```
/var/nfs-data/pv01        *(rw,root_squash)
/var/nfs-data/pv02        *(rw,root_squash)
/var/nfs-data/pv03        *(rw,root_squash)
/var/nfs-data/pv04        *(rw,root_squash)
/var/nfs-data/pv05        *(rw,root_squash)
/var/nfs-data/pv06        *(rw,root_squash)
```

Fig. 14:/etc/exports

- **Setting ownership of the mountpoint**
  # chown -R nfsnobody.nfsnobody /var/nfs-data/
  # chmod -R 0770 /var/nfs-data/

- **Setting firewall rules to allow NFS traffic**
  # iptables -I INPUT -p tcp –dport 2049 -j ACCEPT
  # service iptables save

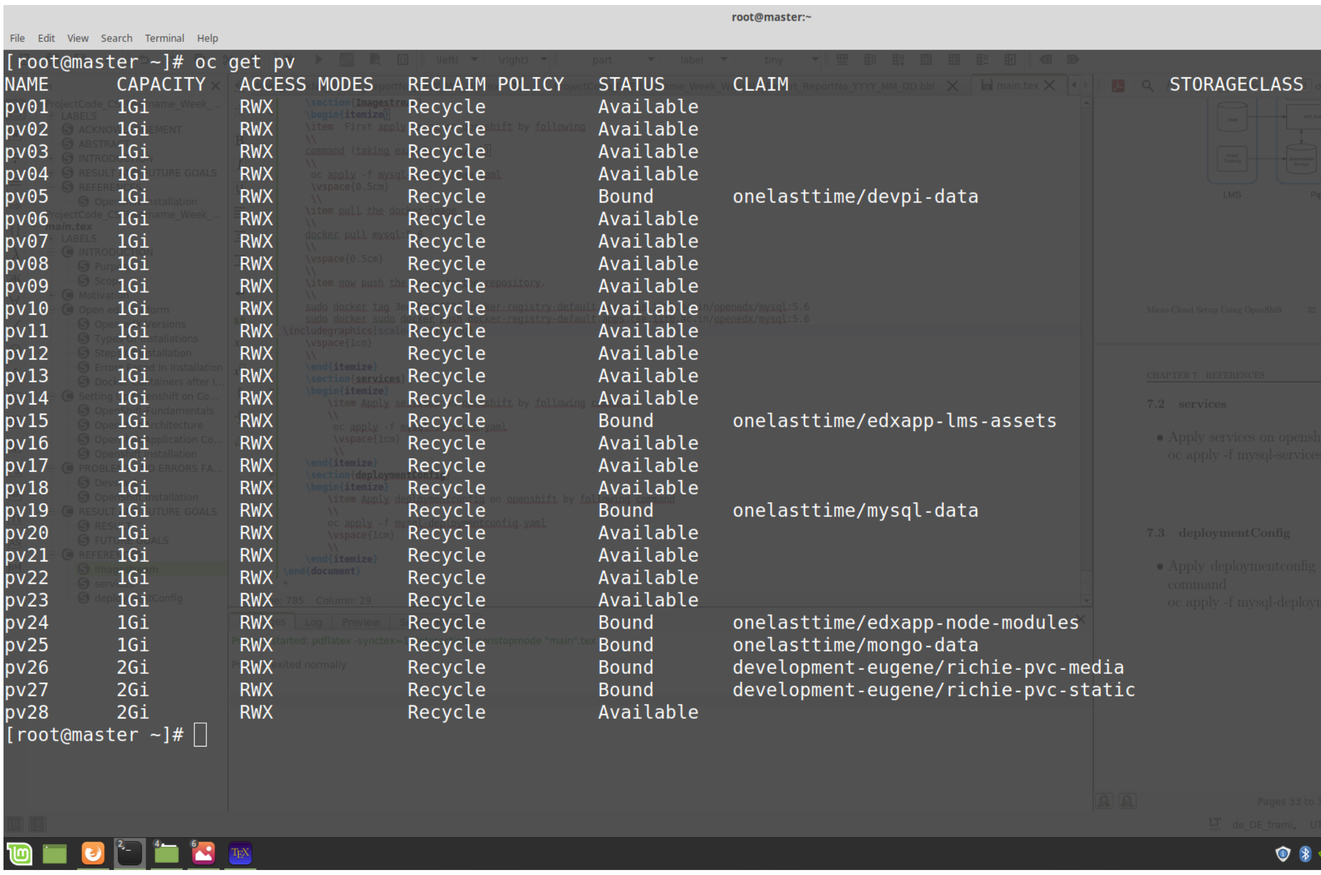
5. **Enabling and starting NFS**

- **Starting NFS services**
  for i in rpcbind nfs-server nfs-lock nfs-idmap;do systemctl restart

$i;done
for i in rpcbind nfs-server nfs-lock nfs-idmap;do systemctl enable

- **Confirming that your NFS volume is exported and ready to use**
  # exportfs

6. **Creating a physical volume**
   To create a resource from a YAML template, use the oc create command along with the -f parameter, which specifies the template file you want to process.
   # oc –config /etc/origin/master/admin.kubeconfig create -f pv01.yaml
   persistentvolume "pv01" created
   To get all active PVs on the server # oc get pv

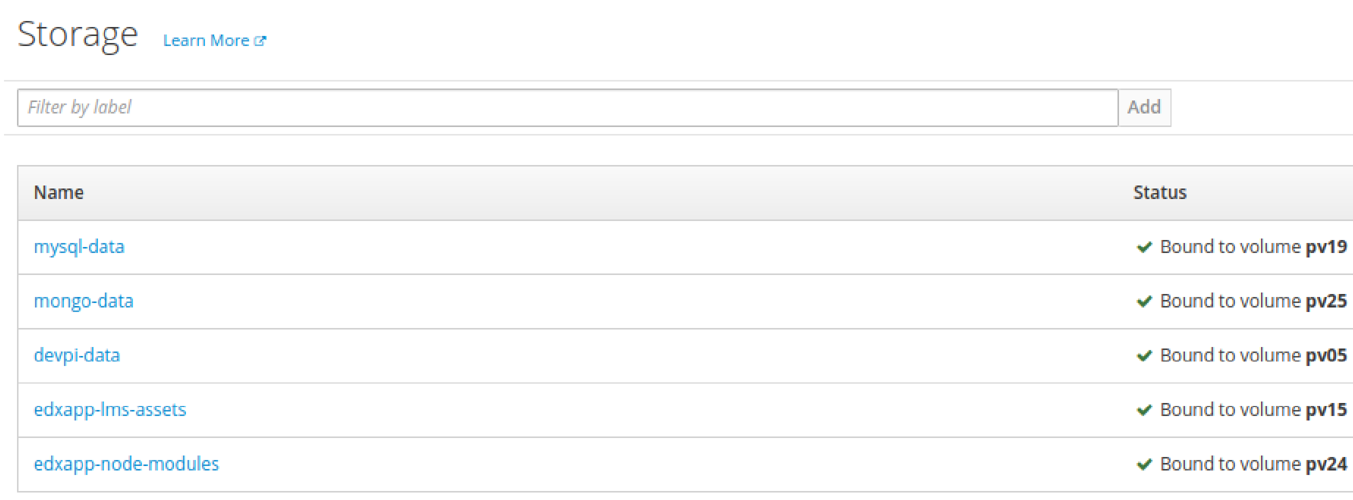


```
[root@master ~]# oc get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                                STORAGECLASS
pv01      1Gi        RWX            Recycle          Available
pv02      1Gi        RWX            Recycle          Available
pv03      1Gi        RWX            Recycle          Available
pv04      1Gi        RWX            Recycle          Available
pv05      1Gi        RWX            Recycle          Bound       onelasttime/devpi-data
pv06      1Gi        RWX            Recycle          Available
pv07      1Gi        RWX            Recycle          Available
pv08      1Gi        RWX            Recycle          Available
pv09      1Gi        RWX            Recycle          Available
pv10      1Gi        RWX            Recycle          Available
pv11      1Gi        RWX            Recycle          Available
pv12      1Gi        RWX            Recycle          Available
pv13      1Gi        RWX            Recycle          Available
pv14      1Gi        RWX            Recycle          Available
pv15      1Gi        RWX            Recycle          Bound       onelasttime/edxapp-lms-assets
pv16      1Gi        RWX            Recycle          Available
pv17      1Gi        RWX            Recycle          Available
pv18      1Gi        RWX            Recycle          Available
pv19      1Gi        RWX            Recycle          Bound       onelasttime/mysql-data
pv20      1Gi        RWX            Recycle          Available
pv21      1Gi        RWX            Recycle          Available
pv22      1Gi        RWX            Recycle          Available
pv23      1Gi        RWX            Recycle          Available
pv24      1Gi        RWX            Recycle          Bound       onelasttime/edxapp-node-modules
pv25      1Gi        RWX            Recycle          Bound       onelasttime/mongo-data
pv26      2Gi        RWX            Recycle          Bound       development-eugene/richie-pvc-media
pv27      2Gi        RWX            Recycle          Bound       development-eugene/richie-pvc-static
pv28      2Gi        RWX            Recycle          Available
[root@master ~]#
```

Fig. 15:List of Persistent Volumes

5.3 Storage binding to persistent volume

Persistent Volume Claims can be made after persistent volumes are created.
The PVC can be created by using the yaml file.
for example:
oc apply -f devpi-data-persistentvolumeclaim.yaml

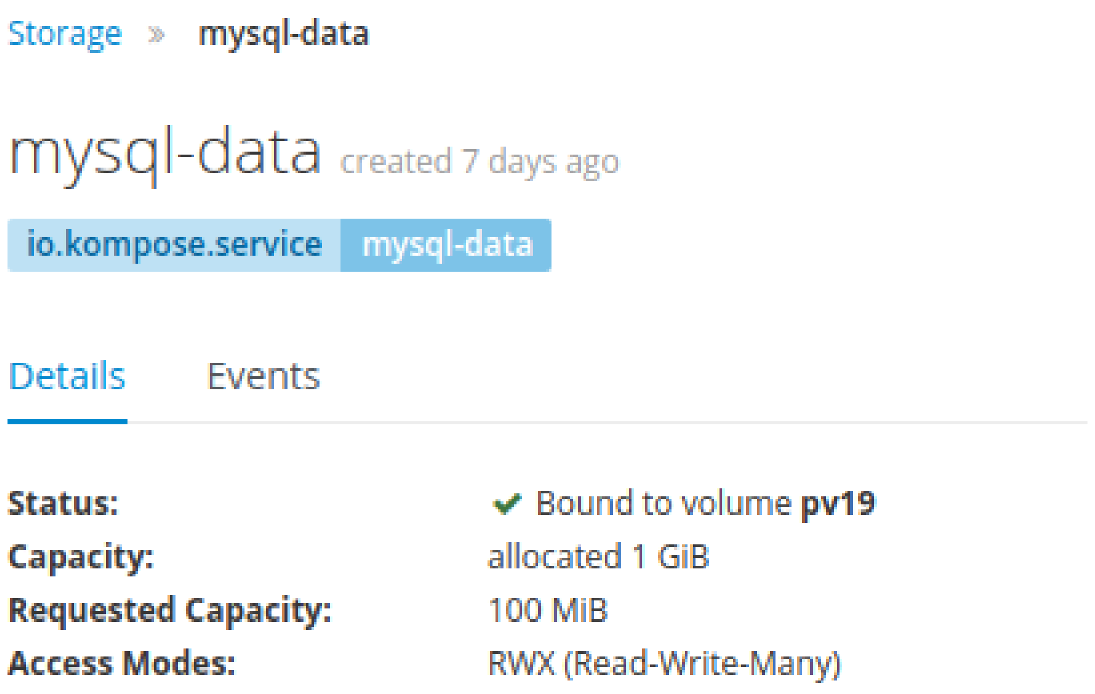

Storage Learn More

Filter by label Add

Name	Status
mysql-data	✓ Bound to volume **pv19**
mongo-data	✓ Bound to volume **pv25**
devpi-data	✓ Bound to volume **pv05**
edxapp-lms-assets	✓ Bound to volume **pv15**
edxapp-node-modules	✓ Bound to volume **pv24**

Fig. 16:List Of PVC

Storage Example


Storage » mysql-data

mysql-data created 7 days ago

io.kompose.service mysql-data

Details Events

Status:	✓ Bound to volume **pv19**
Capacity:	allocated 1 GiB
Requested Capacity:	100 MiB
Access Modes:	RWX (Read-Write-Many)

Fig. 17:mysql-data PVC

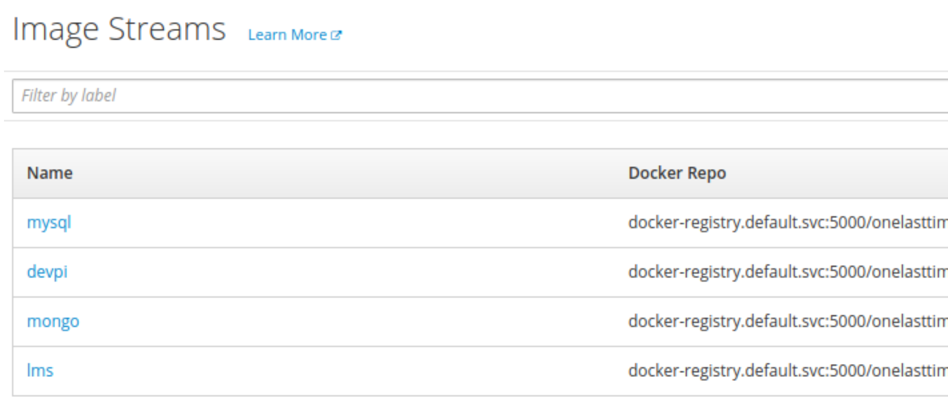
5.4 Imagestream

Image streams are used to automate actions in Openshift. They consist of links to one or more container images. Using image streams, you can monitor applications and trigger new deployments when their components are updated.

- First apply image on openshift by following command (taking example of mysql)
  oc apply -f mysql-imagestream.yaml

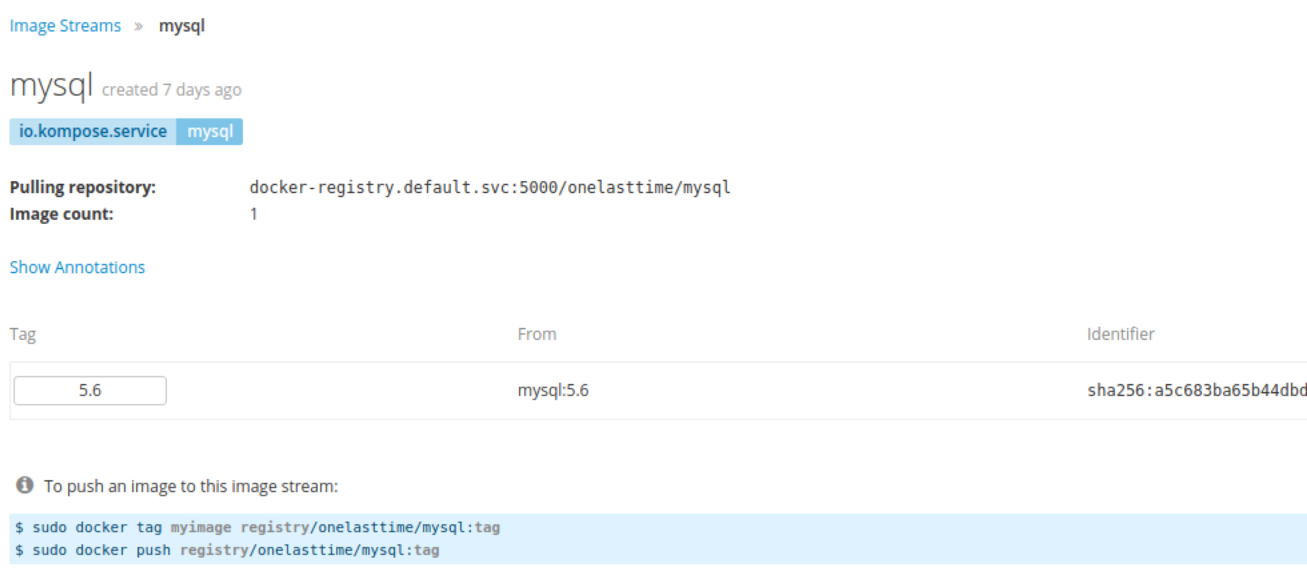
- pull the docker image
  docker pull mysql:5.6

- now push the image to the repository.
  sudo docker tag 3ed1080b793f docker-registry-default.apps.cse.iitb.ac.in/openedx/
  sudo docker sudo docker push docker-registry-default.apps.cse.iitb.ac.in/openedx/



Name	Docker Repo
mysql	docker-registry.default.svc:5000/onelasttim
devpi	docker-registry.default.svc:5000/onelasttim
mongo	docker-registry.default.svc:5000/onelasttim
lms	docker-registry.default.svc:5000/onelasttim

Fig. 18:Image Streams



Pulling repository: docker-registry.default.svc:5000/onelasttime/mysql
Image count: 1

Tag	From	Identifier
5.6	mysql:5.6	sha256:a5c683ba65b44dbd

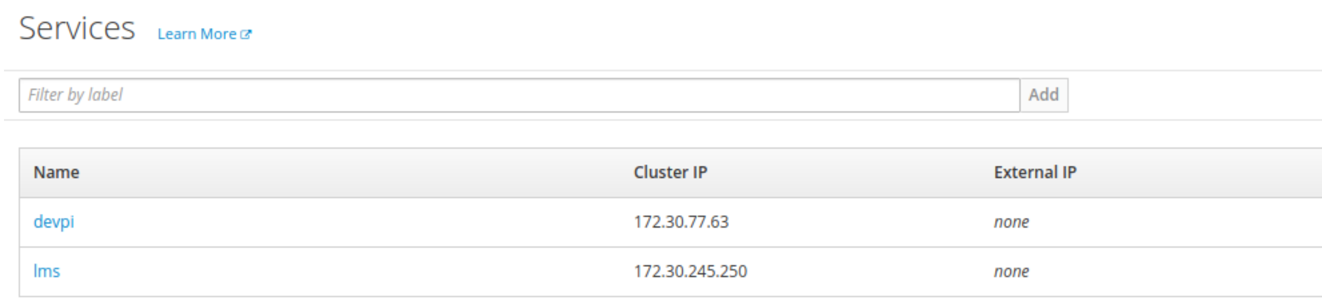
To push an image to this image stream:

```
$ sudo docker tag myimage registry/onelasttime/mysql:tag
$ sudo docker push registry/onelasttime/mysql:tag
```

Fig. 19:mysql Image Stream

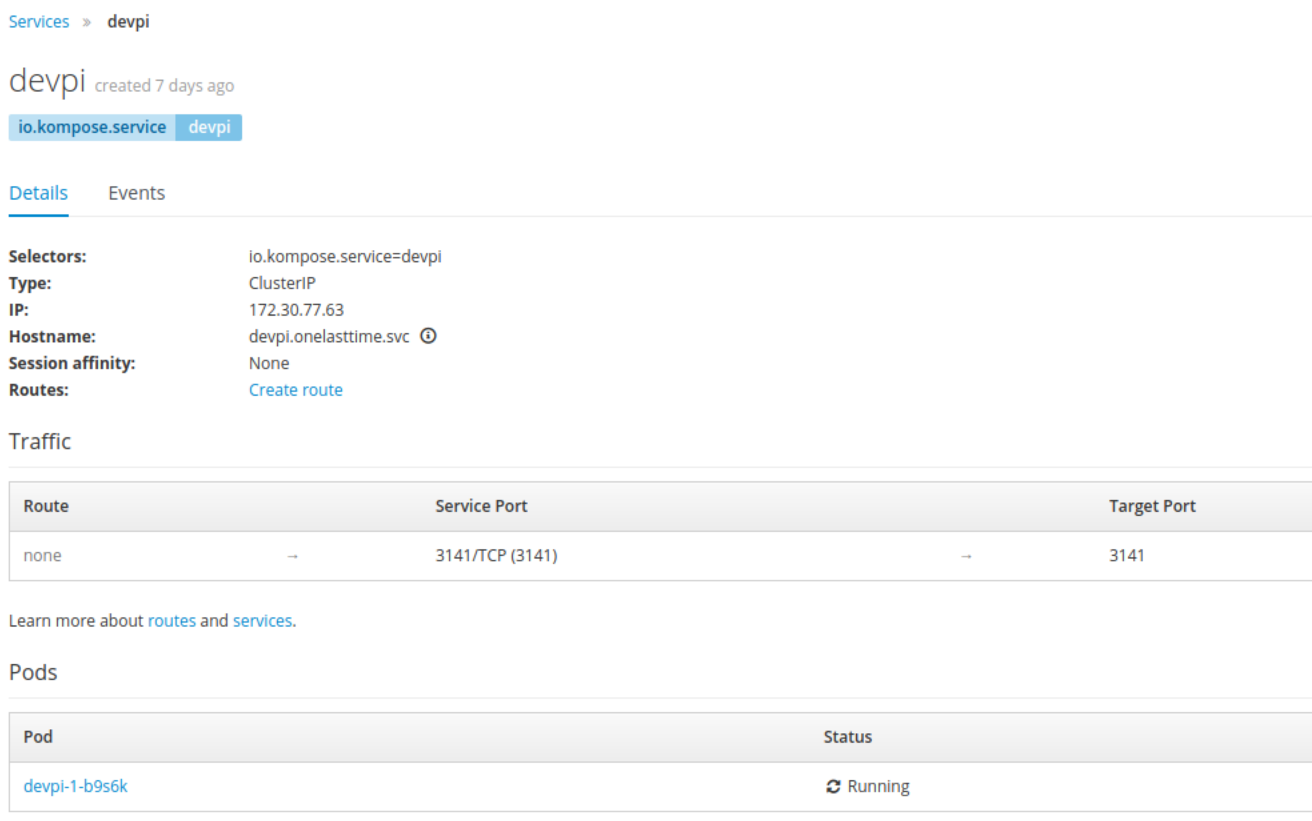
5.5 services

- Apply services on openshift by following command
  oc apply -f mysql-services.yaml



Name	Cluster IP	External IP
devpi	172.30.77.63	none
lms	172.30.245.250	none

Fig. 20:Services


Services » devpi

devpi created 7 days ago

io.kompose.service devpi

Details Events

Selectors:	io.kompose.service=devpi
Type:	ClusterIP
IP:	172.30.77.63
Hostname:	devpi.onelasttime.svc
Session affinity:	None
Routes:	Create route

Traffic

Route		Service Port		Target Port
none	→	3141/TCP (3141)	→	3141

Learn more about routes and services.

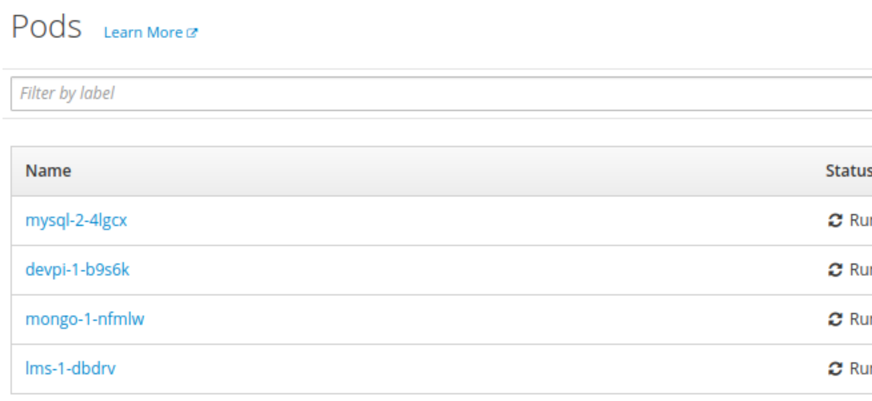
Pods

Pod	Status
devpi-1-b9s6k	Running

Fig. 21:devpi service

5.6 deploymentConfig

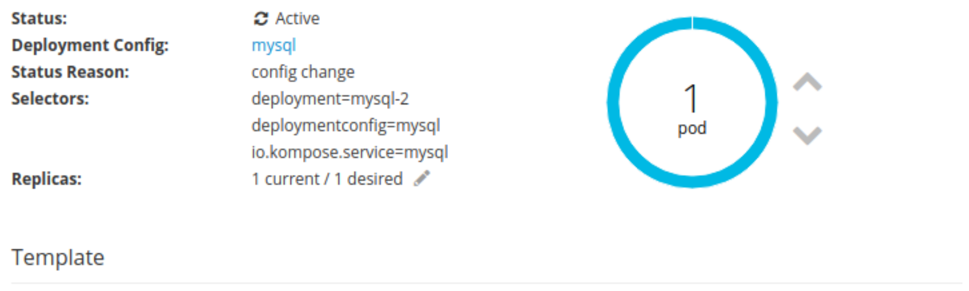
- Apply deploymentconfig on openshift by following command
  oc apply -f mysql-deploymentconfig.yaml


Pods Learn More

Filter by label

Name	Status
mysql-2-4lgcx	Ru
devpi-1-b9s6k	Ru
mongo-1-nfmlw	Ru
lms-1-dbdrv	Ru

Fig. 22:List of deployments



Status:	Active
Deployment Config:	mysql
Status Reason:	config change

Selectors:	deployment=mysql-2
	deploymentconfig=mysql
	io.kompose.service=mysql

Replicas:	1 current / 1 desired

Fig. 23:mysql deployment

Chapter 6

PROBLEMS AND ERRORS FACED

6.1 Devstack

- **./provision.sh: line 21: /usr/local/bin/docker-compose: Permission denied**
  Makefile:59: recipe for target 'dev.provision.run' failed
  make: *** [dev.provision.run] Error 126
  Solution :
  sudo − i
  curl -L
  https://github.com/docker/compose/releases/download/1.18.0/docker-compose-‘uname -s‘-‘uname -m‘ -o /usr/local/bin/docker-compose
  *chmod*755*/usr/local/bin/docker − compose*
  exit

- **TASK [common : Update expired apt keys]** ***************
  This error occurs due to the fetching to apt-keys from an expired link.
  Solution :
  1.sudo exec -it edx.devstack.lms bash
  2.cd app/edx_ansible/edx_ansible/playbooks/roles/common_vars/defaults/

  3.vi main.yml
  4.On entering the main.yml file in vim editor change the link for the keyword COMMON_EDX_PPA_KEY from keyserver.ubuntu.com to hkp://keyserver.ubuntu.com:80

6.2 OpenShift Installation

- **wget-1.14-18.el7_6.1.x86_64:No more mirrors to try.**
  Fix: sudo yum install wget

- **TLS Handshake Error/Connection Timed out**
  Fix: Run the command again.

- **API Server Error:Get https://127.0.0.1:8443/healthz?timeout=32s:**

  Connect : connection refused () Probable Reason: Lower Resources on PC (Weak VM):
  (Encountered This Error when installation was done on Single Machine Using VMs)
  Fix: used 4 dedicated machines for multi-node installation.

- **Error :- Starting openshift.io/sdn**
  IP: 10.128.0.0 conflicts with host network : 10.129.132.0/24
  While Running deploy_cluster.yml
  In the task Running Handler [openshift_master:restart master-controllers]

  Resolution
  vi /etc/origin/master/master-config.yaml
  under networkconfig-
  change clusternetwork CIDR from 14 to 16.

Chapter 7

RESULT AND FUTURE GOALS

7.1 RESULT

In the project Micro-Cloud setup using Openshift, we have

- Configured system hardware according to the Openshift requirement of setup involving one master and 3 nodes.
- Successfully setup a system consisting of 4 machines, one serving as master and other 3 as nodes.
- Installed Openshift on local server with 1 master, 1 infra node and 2 nodes.
- Accessed openshift through terminal as well as openshift console i.e. **master-console**.
- Successfully build and deployed various applications based on Ruby,PHP, NodeJs etc.
- Did Successful Scaling and Descaling of pods.
- Tried Installation of OpenEdx platform on openshift cluster.
- Deployed various services of open-edx like mysql,mongodb, chrome,lms,cms etc on openshift.

7.2 FUTURE GOALS

- Complete deployment of open-edx on openshift by resolving conflicts, being faced in provisioning.
- Installation of open-edx on openshift, using Arnonld tool. here is github repository for Arnold **Arnold-openshift**.
- Openshift installation on multiple machines.

REFERENCES

- The github link for all documentations related to project is **https://github.com/fresearchgroup/Micro-Cloud-setup-using-Openshift-on-commodity-servers**.

- The Origin Community Distribution of Kubernetes that powers Red Hat Openshift.
  https://www.okd.io/.

- Setting up of openshift in multinodes.
  https://youtu.be/JhlzSoayksY.

- Complete steps to install openshift
  https://github.com/fresearchgroup/Micro-Cloud-setup-using-Openshift-on-commodity-servers

- Deploying open-edx using Arnold
  https://github.com/openfun/arnold.

- Openshift in Action by Jamie Duncan, John Osborne.

- Image Sources
  Fig. 1:http://http://protoinformatico.com/"Accessed on July 4 2019"

  Fig. 2:https://en.wikipedia.org/wiki/Docker_(software)"Accessed on July 4 2019"
  Fig. 3:https://www.mindfireit.com/cyber-security/introduction-to-container-services/attachment/introduction-to-container-services-docker-kubernetes-openshift-configuration-code-runtime-engine/"Accessed on July 4 2019"

  Fig. 4:https://blog.docker.com/2018/09/join-the-beta-for-docker-engine-18-09/docker-website-2018-diagrams-071918-v5_a-docker-engine-page-first-panel-2/"Accessed on July 4 2019"
  Fig. 5:https://cdn-images-1.medium.com "Accessed on July 4 2019"
  Fig. 6:https://techbeacon.com "Accessed on July 4 2019"
  Fig. 7:https://www.networkslayer.com/ingress/ "Accessed on July 4

2019”
Fig. 8:https://image.slidesharecdn.com ”Accessed on July 4 2019”
Fig. 10:https://blog.openshift.com/openshift-v3-deep-dive-docker-kubernetes/
”Accessed on July 4 2019”