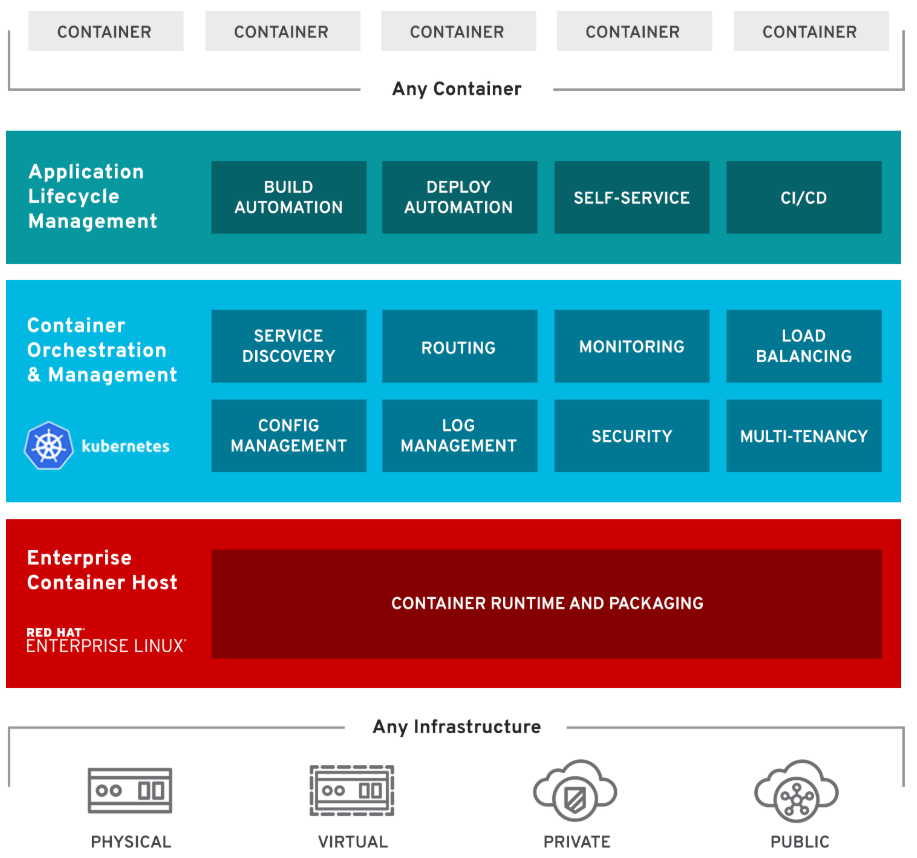
# Deploying OpenShift OKD Cluster setup

# OpenShift Layered Structure­



OpenShift Features

1. **Traditional and Cloud-Native**: Handles cloud-native and traditional applications on a single platform
2. **Multi-tenancy**: Gives your teams self-service to consistent environments across your organization, from development through production.
3. **Hybrid Infra Support**: Offers trusted, proven Kubernetes on any infrastructure.
4. **Security**: Integrated throughout OpenShift, from infrastructure to services, and throughout the operations and application lifecycle
5. **Service on demand**: Seamlessly configure, deploy and consume private and public cloud services with OpenShift Service Catalog.
6. **Built-in Automation**: Remove routine tasks from application teams to focus on business value.
7. **Pluggable Architecture**: Your choice of supported, container runtimes, networking, storage, CI/CD solutions.

# OpenShift Components:

Server roles

There are two types of roles a server can play in OpenShift:

* **Master**: Holding the key services, managing nodes, load distribution, management of OpenShift platform, orchestration. Typically, a master server has the API server, etcd, controller manager, and HAProxy (load balance)
* **Node**: Handles the actual load, act as compute provider. Provides runtime environment for containers. Typically, a node has docker service, kubelet, network proxy services.
* **ETCD**: The **openshift** Master, **openshift** provides a REST endpoint for interacting with the system. An **etcd** server, **Openshift** uses **etcd** to store system configuration and state. Controllers: Controllers are the components that run with the masters that makes sure the running system matches the desired state as stored in **etcd**.

## Services

**API Service**:

* Validates and configures the data for pods, services, and replication controllers.
* Synchronizes pod information with service configuration

**Etcd service**: Stores the persistent master state

**Controller Manager service**: Watches etcd for changes to replication controller objects and then uses the API to enforce the desired state.

**HAProxy service**: Provides load balancing for master servers.

**Kubelet**: Updates the node as specified by a container manifest

**Network proxy**: Reflects the services defined in the API on that node.

## Important Components

**OCR**: Provides users with a built-in location for their application builds to push the resulting images

**Web console**: A user interface accessible from a web browser

**OC CLI**: A CLI-based interface to manage OpenShift and use services.

**Pod**:

* Pods are the rough equivalent of a machine instance (physical or virtual) to a container.
* Each pod is allocated its own internal IP address, therefore owning its entire port space.
* Containers within pods can share their local storage and networking.
* Pod is a smallest compute unit that can be defined, deployed, and managed.

## Users

Users are those who usage and consumed the services.

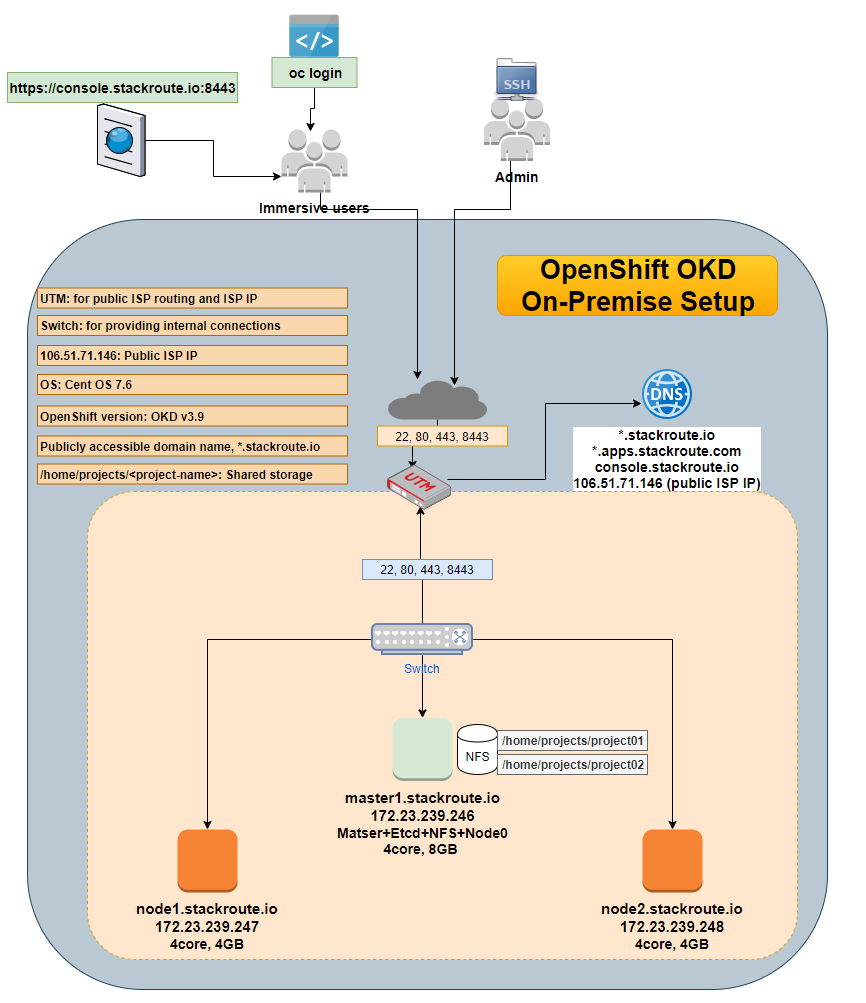
1. **Regular users** can login and can management their applications as per need.
2. **System users** are created automatically for the internal services usage, authentication and platform management.

## Setup Requirements

1. Operating System: Red Hat Enterprise Linux 7 (64-bit)
2. Compatibility: ESXi 6.5 and later (VM version 13)
3. VMware Tools: Running, version:10309 (Guest Managed)
4. Master: 2core and 16GB RAM.
5. Nodes: Two nodes with 2 core and 16GB RAM.
6. Domain name: console.Hobbylobby.com, \*.apps.Hobbylobby.com
7. Private IP Scheme: Any private scheme except 172.16.x.x/16
8. Connectivity: Private IP connectivity with password-less SSH access from master to all servers.

Note: Whatever commands and values are mentioned with italic and in yellow color need to be replaced with your own requirements.

## Setup Design



## On master and all nodes servers

Once the setup is ready with basic requirements, proceed for the below steps. The following commands need to be executed on all master and node servers included in your cluster.

1. Switch to root user using the following command.

sudo su -

cd

1. Update the yum packages using the following command.

yum update

reboot

1. Install the prerequisites with basic tools using the following command.

yum install -y wget git zile 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 python2-pip python-devel python-passlib java-1.8.0-openjdk-headless "@Development Tools"

1. Install the latest RHEL 7 epel repository using the following command.

yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

1. Enable the added RHEL 7 epel repository using the following command.

sed -i -e "s/^enabled=1/enabled=0/" /etc/yum.repos.d/epel.repo

1. Start the network manager service

systemctl start NetworkManager

systemctl enable NetworkManager

## Only on Master server or Ansible Admin server

The following steps only need to be performed on your server you deiced to act as ansible deployment server.

1. Installing ansible on master server using the following command.

yum -y --enablerepo=epel install ansible pyOpenSSL

1. Create the project directory using the following command.

*mkdir openshift*

*cd openshift*

1. Cloning the OpenShift ansible git repository using the following command.

git clone https://github.com/openshift/openshift-ansible.git

1. Check out the desired stable version ( e.g v3.9) using the following command.

cd openshift-ansible && git fetch && git checkout release-3.9 && cd ..

## On all master and nodes servers

OpenShift OKD needs a working DNS server that can resolve the master and node servers FQDNs. If your OpenShift OKD also need to be accessed from public network, then you must have proper domain name registered with the following records:

<Public ip of master server> <record type> <public domain name>

<x.x.x.x> A <master.domian.com>

<x.x.x.x> A <console.domian.com>

<x.x.x.x> A <\*.domian.com>

<x.x.x.x> A <\*apps.domian.com>

<x.x.x.x> A <node01.domian.com>

<x.x.x.x> A <node02.domian.com>

Alternatively, you can add all host entries in host file of all servers. However, \*.domain.com does not work for host file so either you must have proper DNS domain name, or you must register each project’s public domain name in your host file.

1. Make the appropriate changes in the /etc/hosts file on all servers.

vi /etc/hosts

*10.100.37.88 Sndomaster01.hobbylobby.corp Sndomaster01*

*10.100.37.89 Sndomaster02.hobbylobby.corp Sndomaster02*

*10.100.37.90 Sndomaster03.hobbylobby.corp Sndomaster03*

*10.100.37.91 Sndosinfra01.hobbylobby.corp Sndosinfra01*

*10.100.37.92 Sndosinfra02.hobbylobby.corp Sndosinfra02*

*10.100.37.93 Sndosinfra03.hobbylobby.corp Sndosinfra03*

*10.100.37.94 Sndoslb01.hobbylobby.corp Sndoslb01*

*10.100.37.95 Sndospod01.hobbylobby.corp Sndospod01*

*10.100.37.96 Sndospod02.hobbylobby.corp Sndospod02*

Note: You need to replace the appropriate IPs and Domain names as per your setup.

## Only on master server.

1. Restart and enable the docker service on all servers

systemctl stop docker && systemctl restart docker && systemctl enable docker

1. Generating the ssh keys and coping the public key to all servers for ansible connection from loadbalancer.

ssh-keygen

for host in *10.100.37.88*\

*10.100.37.89* \

*10.100.37.90* \

*10.100.37.91\*

*10.100.37.92\*

*10.100.37.93\*

*10.100.37.95\*

*10.100.37.96;\cat*

do ssh-copy-id -i ~/.ssh/id\_rsa.pub $host; \

done

1. Creating an inventory file for ansible playbook, paste the content of inventory.ini file in this file.

vi inventory.ini

The sample of inventory.ini file can be found here.

<https://github.com/vipin-k/Openshift-Cluster/blob/master/inventory.ini>

1. Execute the prerequisites ansible playbook using the following command.

cd /usr/share/ansible/openshift-ansible/playbooks

ansible-playbook -i /kumar/inventory prerequisites.yml

Note: Make sure the ansible user mentioned in the inventory file is able to access all master and node servers before to execute playbook.