Introduction to Kubernetes

1. 1. Introduction to Kubernetes 1.0 Compiled by Rajdeep Twitter : @rajdeepdua July 2015
2. [2.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-2-638.jpg?cb=1437831998) Agenda • Introduction • Key Components • Architecture
3. [3.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-3-638.jpg?cb=1437831998) What is Kubernetes • Service for Container Cluster Management • Open Sourced by Google • Supports GCE, CoreOS, Azure, vSphere, • Used to manage Docker containers as a default implementation
4. [4.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-4-638.jpg?cb=1437831998) High Level Components
5. [5.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-5-638.jpg?cb=1437831998) Key Concepts • Concepts – Master – Nodes – Pod – Service and Labels – Container – Node • Kubelet • Kubernetes Proxy • Kubernetes Control Panel – API Server – Controller Manager – Persistent store : etcd
6. [6.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-6-638.jpg?cb=1437831998) Master • Master maintains the State of the Kubernetes Server runtime • State is maintained in the etcd backend • It is the point of entry for all the client calls to configure and manage Kubernetes components like Nodes, Pods, ReplicationControllers, Services
7. [7.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-7-638.jpg?cb=1437831998) Master • Master is also made up of following components – API Server – Scheduler – Registries (Internal Mechanism to Persist data) • Minon Registry • Pod Registry • Service Registry • Binding Registry – Storage
8. [8.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-8-638.jpg?cb=1437831998) Master • restful.Container – Container for webservices exposed • Storage Objects – PodStorage – NodeStorage – ReplicationControllerStorage – ServicesStorage – PersistVolumeStorage
9. [9.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-9-638.jpg?cb=1437831998) Master – Key Components
10. [10.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-10-638.jpg?cb=1437831998) Node • Represents the resource provided for provisioning pods • Node runs a Docker etcd and a kubelet daemon
11. [11.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-11-638.jpg?cb=1437831998) Node Registry • Registry for keeping track of the nodes in the Kubernetes cluster • It is a Set implementation • Actual implementation fetches list of hosts from the underlying cloudprovider • Referenced from the Master – Actions performed on a Minon Registry – Insert a Node – Delete a Node – Contains a Node – List of Nodes
12. [12.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-12-638.jpg?cb=1437831998) Pod
13. [13.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-13-638.jpg?cb=1437831998) Pod Pod is a collection of containers that can run on a host. This resource is created by clients and scheduled onto hosts. • Pod represents a logical construct to bundle one or more applications together • It represents a Logical Host • Volumes can be shared within the application in the same pod • In the docker world pod represents a bundle of containers with shared volumes • Pods are ephemeral in nature and never re-scheduled on other nodes
14. [14.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-14-638.jpg?cb=1437831998) Relation between a Node and a Pod
15. [15.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-15-638.jpg?cb=1437831998) Pod Structure
16. [16.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-16-638.jpg?cb=1437831998) Pod Registry • Wrapper on top of etcd persistent store • Keeps track of Pods and their mapping to minions • Actions Performed on a Pod Registry – List Pods – based on a Selector – Watch Pods – Create a Pod – Update a Pod – Delete a Pod
17. [17.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-17-638.jpg?cb=1437831998) What is a Service? • A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access them - sometimes called a micro-service. • The set of Pods targeted by a Service is (usually) determined by a Label Selector
18. [18.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-18-638.jpg?cb=1437831998) Service • A service defines a TCP or UDP port reservation. • Provides a way for applications running in containers to connect to each other without requiring that each one be configured with the end-point IP addresses. • Allows for abstracted configuration and for mobility and load balancing of the providing containers. • When a Kubernetes service, the service providers will be labeled to receive traffic and the service consumers will be given the access information in the environment so that they can reach the providers.
19. [19.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-19-638.jpg?cb=1437831998) Services • Elements of a Service – Name – Port of the proxy – Labels of a Service – Selector – Uses LoadBalancer – Container Port
20. [20.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-20-638.jpg?cb=1437831998) Example Service { "kind": "Service", "apiVersion": "v1", "metadata": { "name": "my-service" }, "spec": { "selector": { "app": "MyApp" }, "ports": [ { "protocol": "TCP", "port": 80, "targetPort": 9376 } ] } }
21. [21.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-21-638.jpg?cb=1437831998) Service Details
22. [22.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-22-638.jpg?cb=1437831998) ServiceRegistry • Wrapper on top of etcd persistent store which keeps track of Services • List of Actions that can be performed on this registry – Create Service – Get Service – Delete a Service – Update Service – Update Endpoints for the service – List Services
23. [23.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-23-638.jpg?cb=1437831998) Sequence : List Services
24. [24.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-24-638.jpg?cb=1437831998) Sequence List Services – Server Side
25. [25.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-25-638.jpg?cb=1437831998) Replication Controller • A replication controller ensures that a specified number of pod "replicas" are running at any one time • Relevant for pods with RestartPolicy = Always • Replication Controller uses Pod Templates to create Pods • Replication controller uses Pod Labels to monitor and maintain the number of Pods to the desired level
26. [26.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-26-638.jpg?cb=1437831998) Replication Controller - Sample
27. [27.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-27-638.jpg?cb=1437831998) Volumes • Container’s disks a ephemeral in nature • Everytime container restarts ephemeral disks are restarted • Docker volumes are just a mount point or host dir • Kubernetes Volumes allow lifecycle of a volume to be tied to that of the pod • Different kind of volumes exist : emptyDir, hostPath, iSCSi volume, AWS EBS, GCE Persistent disc
28. [28.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-28-638.jpg?cb=1437831998) Scheduler • Responsible for scheduling POD on a minion • Multiple implementations possible type Scheduler interface { Schedule(api.Pod, MinionLister) (selectedMachine string, err error) }
29. [29.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-29-638.jpg?cb=1437831998) Scheduler Implementations • Random Scheduler • Round robin Scheduler
30. [30.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-30-638.jpg?cb=1437831998) Kubelet • Component which runs on each minion and manages the Pod and Container Lifecycle • There is 1:1 mapping between a Host and a Kubelet • Key Elements of a Kubelet – Docker Client – Root Direcotry – Pod Workers – Etcd client – Cadvisor client
31. [31.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-31-638.jpg?cb=1437831998) Kubelet • Key Elements of a Kubelet – Hostname : Name of the host, – Docker Client: based on github.com/fsouza/go-dockerclient, used for Docker container create, start, stop and delete – Pod Workers : Workers which act on each POD – Etcd client : Interface for the persistent store – Cadvisor client – Health Checker
32. [32.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-32-638.jpg?cb=1437831998) Functions performed by a Kubelet • Run a Action on a Pod using a Worker • Make binding between Volumes and a container. • Make binding between Ports and a container. • Run a single container in a given POD • Kill a Container • Create a Network Container for a POD • Delete all containers in a POD • Sync POD state with the data structure in a Kubelet
33. [33.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-33-638.jpg?cb=1437831998) Functions performed by a Kubelet..cont • Run a Command in a Container • Health Information of the Container • Root and POD info from Cadvisor
34. [34.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-34-638.jpg?cb=1437831998) Run Container : Sequence Diagram
35. [35.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-35-638.jpg?cb=1437831998) Run Container : Sequence Diagram
36. [36.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-36-638.jpg?cb=1437831998) Run Container : Sequence Diagram
37. [37.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-37-638.jpg?cb=1437831998) Run Container : Sequence Diagram
38. [38.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-38-638.jpg?cb=1437831998) Run Container : Sequence Diagram
39. [39.](https://image.slidesharecdn.com/intro-kubernetes-140908081347-phpapp02/95/introduction-to-kubernetes-39-638.jpg?cb=1437831998) Summary • Kubernetes allows you to deploy and manage applications running on multiple hosts using docker • Not tied to a particular cloud implementation but inspired by GCE and Google Infrastructure

<https://www.slideshare.net/rajdeep/introduction-to-kubernetes>

<https://www.slideshare.net/imesh/an-introduction-to-kubernetes>

<https://medium.com/devopslinks/how-to-setup-a-perfect-kubernetes-cluster-using-kops-in-aws-b616bdfae013>

<https://aws-labs.com/install-kubernetes-dns-dashboard-centos/>

<https://dzone.com/articles/easy-step-by-step-local-kubernetes-source-code-chav>

[www.yet.org/2016/06/tectonic/](http://www.yet.org/2016/06/tectonic/)

<https://kubernetes.io/docs/tutorials/hello-minikube/>

<https://github.com/red-gate/ks>

<https://kubernetes.io/docs/setup/scratch/>

<https://www.edureka.co/blog/install-kubernetes-on-ubuntu>

<https://icicimov.github.io/blog/kubernetes/Kubernetes-cluster-step-by-step/>

<https://www.digitalocean.com/community/tutorials/how-to-create-a-kubernetes-1-10-cluster-using-kubeadm-on-centos-7>

<https://www.assistanz.com/steps-to-install-kubernetes-cluster-manually-using-centos-7/>

<https://mherman.org/blog/setting-up-a-kubernetes-cluster-on-ubuntu/>

<https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/>

<https://www.admintome.com/blog/install-kubernetes-ubuntu/>

**mongo sharding**

<https://www.howtoforge.com/tutorial/deploying-mongodb-sharded-cluster-on-centos-7/>

<https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/>

<http://www.treselle.com/blog/mongodb-shard-setup-part-ii/>

<https://www.guru99.com/mongodb-sharding-implementation.html>

<https://sanaulla.info/2015/02/02/setting-up-sharded-mongodb-cluster-in-localhost/>

<https://computingforgeeks.com/how-to-setup-3-node-kubernetes-cluster-on-ubuntu-18-04-with-weave-net-cni/>

kubeadm join 172.31.43.228:6443 --token swhezj.ogb6g3xcsb640a5j --discovery-token-ca-cert-hash sha256:0b67a2a33dd6c3389f9ba87fa3c03acecea444ce4181ae63afed057a4d89a20f

In this guide, I’ll take you through the steps to install and set up a working 3 node Kubernetes Cluster on Ubuntu 18.04 Bionic Beaver Linux.  Kubernetes is an open-source container-orchestration system used for automating deployment, management, and scaling of containerized applications.

## Kubernetes on Ubuntu 18.04 – System Diagram

Let’s configure system hostnames before we can proceed to next steps:

**On Master Node:**

Set hostname like below:

$ sudo hostnamectl set-hostname k8s-master

**On Worker Node 01:**

Set the hostname using hostamectl command line tool.

$ sudo hostnamectl set-hostname k8s-node-01

**On Worker Node 02:**

Also set hostname for Kubernetes worker node 02.

$ sudo hostnamectl set-hostname k8s-node-02

Once correct hostname has been configured on each host, populate on each node with the values configured.

$ cat /etc/hosts

192.168.2.2 k8s-master

192.168.2.3 k8s-node-01

192.168.2.4 k8s-node-02

## Setup Kubernetes on Ubuntu 18.04 – Prerequisites (Run on all nodes)

Before doing any Kubernetes specific configurations, let’s ensure all deps are satisfied. Here we will do a system update and create Kubernetes user.

Update system packages to the latest release on all nodes:

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install linux-image-extra-virtual

sudo reboot

Add user to manage Kubernetes cluster:

sudo useradd -s /bin/bash -m k8s-admin

sudo passwd k8s-admin

sudo usermod -aG sudo k8s-admin

echo "k8s-admin ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/k8s-admin

If you prefer entering sudo password when running sudo commands as k8s-admin user, then you can ignore the last line. You can test if no password prompt for sudo:

$ su - k8s-admin

k8s-admin@k8s-master:~$ sudo su -

root@k8s-master:~#

All looks good, let’s proceed to install Docker engine.

## Setup Kubernetes on Ubuntu 18.04 – Install Docker Engine

Kubernetes requires docker to run containers used for hosting applications and other Kubernetes services. We have a comprehensive Docker installation guide:

[How to install Docker CE on Ubuntu / Debian / Fedora / Arch / CentOS](https://computingforgeeks.com/installing-docker-ce-ubuntu-debian-fedora-arch-centos/)

If you need a quick installation guide, then use the following commands to install Docker Engine on Ubuntu 18.04. Ensure any old version of Docker engine is uninstalled on your system:

sudo apt-get remove docker docker-engine docker.i

Install dependencies:

$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

Import Docker repository GPG key:

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

$ sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

Install docker:

sudo apt-get update

sudo apt-get install docker-ce

sudo usermod -aG docker k8s-admin

When docker has been installed, you can continue to configure the Kubernetes master node.

## Setup Kubernetes on Ubuntu 18.04 – Install and Configure Kubernetes Master

All commands that will be executed on this section are meant to be run on the master node. Don’t execute any of the commands on Kubernetes worker nodes. Kubernetes Master components provide the cluster’s control plane – API Server, Scheduler, Controller Manager. They make global decisions about the cluster e.g scheduling and detecting and responding to cluster events.

#### Add Kubernetes repository

As of this writing, there is no official repository for Ubuntu 18.04, we will add a repository for Ubuntu 16.04. I tested it. All packages and dependencies should install fine. I’ll update this article when a repo for Ubuntu 18.04 is available.

# cat <<EOF > /etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

Then import GPG key:

curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

Update apt package index:

sudo apt update

#### Install Kubernetes Master Components

Install kubectl, kubelet, kubernetes-cni and kubeadm Kubernetes master components:

sudo apt install kubectl kubelet kubeadm kubernetes-cni

Confirm that all package binaries are present on the file system.

$ which kubelet

/usr/bin/kubelet

$ which kubeadm

/usr/bin/kubeadm

If swap is on, turn it off.

sudo swapoff -a

#### Initialize Kubernetes Cluster:

When all Kubernetes packages have been installed, you’re ready to initialize the cluster using kubeadm command line tool.

Export required variables (**Optional)**

export API\_ADDR=`ifconfig eth0 | grep 'inet'| cut -d':' -f2 | awk '{print $1}'`

export DNS\_DOMAIN="k8s.local"

export POD\_NET="10.4.0.0/16"

export SRV\_NET="10.5.0.0/16"

Then initialize the Kubernetes cluster using variables defined above:

kubeadm init --pod-network-cidr ${POD\_NET} --service-cidr ${SRV\_NET} \

--service-dns-domain "${DNS\_DOMAIN}" --apiserver-advertise-address ${API\_ADDR}

If all goes well, you should get a success message with the instructions of what to do next:

---

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node

as root:

kubeadm join 192.168.2.2:6443 --token 9y4vc8.h7jdjle1xdovrd0z --discovery-token-ca-cert-hash sha256:cff9d1444a56b24b4a8839ff3330ab7177065c90753ef3e4e614566695db273c

#### Configure Access for k8s-admin user on the Master server

Switch to k8s-adminand copy Kubernetes configuration file with cluster information.

su - k8s-admin

mkdir -p $HOME/.k8s

sudo cp -i /etc/kubernetes/admin.conf $HOME/.k8s/config

sudo chown $(id -u):$(id -g) $HOME/.k8s/config

export KUBECONFIG=$HOME/.k8s/config

echo "export KUBECONFIG=$HOME/.k8s/config" | tee -a ~/.bashrc

### Deploy Weave Net POD Network to the Cluster ( Run as normal user)

Weave Net creates a virtual network that connects Docker containers across multiple hosts and enables their automatic discovery. Services provided by application containers on the Weave network can be exposed to the outside world, regardless of where they are running.

Weave Net can be installed onto your CNI-enabled Kubernetes cluster with a single command:

# su - k8s-admin

$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

serviceaccount/weave-net created

clusterrole.rbac.authorization.k8s.io/weave-net created

clusterrolebinding.rbac.authorization.k8s.io/weave-net created

role.rbac.authorization.k8s.io/weave-net created

rolebinding.rbac.authorization.k8s.io/weave-net created

daemonset.extensions/weave-net created

After a few seconds, a Weave Net pod should be running on each Node and any further pods you create will be automatically attached to the Weave network.

k8s-admin@k8s-master:~$ kubectl get pod -n kube-system | grep weav

weave-net-d9v5v 2/2 Running 0 11h

weave-net-mhp46 2/2 Running 0 11h

weave-net-vmksr 2/2 Running 0 11h

## Setup Kubernetes Worker Nodes

When Kubernetes cluster has been initialized and the master node is online, start Worker Nodes configuration. A node is a worker machine in Kubernetes, it may be a VM or physical machine.  Each node is managed by the master and has the services necessary to run pods – docker, kubelet, kube-proxy

#### Step 1: Ensure Docker is installed (covered)

Ensure docker engine is installed on all Worker nodes. Refer to docker installation section

#### Step 2: Add Kubernetes repository ( covered)

Ensure that repository for Kubenetes packages is added to the system. Refer ^^

#### Step 3: Install Kubenetes components

Once you’ve added Kubernetes repository, install components using:

sudo apt install kubelet kubeadm kubectl kubernetes-cni

#### Step 4: Join the Node to the Cluster:

Use the join command given after initializing Kubernetes cluster. E.g

kubeadm token create --print-join-command

kubeadm join 192.168.2.2:6443 --token 9y4vc8.h7jdjle1xdovrd0z \

--discovery-token-ca-cert-hash sha256:cff9d1444a56b24b4a8839ff3330ab7177065c90753ef3e4e614566695db273c

---

[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...

[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "k8s-node-02" as an annotation

This node has joined the cluster:

\* Certificate signing request was sent to master and a response

was received.

\* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.

When done,  Check nodes status on the master:

k8s-admin@k8s-master:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master Ready master 35m v1.11.0

k8s-node-01 Ready <none> 2m v1.11.0

k8s-node-02 Ready <none> 1m v1.11.0

On the two nodes, Weave Net should be configured.

root@k8s-node-01:~# ip ad | grep weave

6: weave: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1376 qdisc noqueue state UP group default qlen 1000

inet 10.44.0.0/12 brd 10.47.255.255 scope global weave

9: vethwe-bridge@vethwe-datapath: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1376 qdisc noqueue master weave state UP group default

root@k8s-node-02:~# ip ad | grep weave

6: weave: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1376 qdisc noqueue state UP group default qlen 1000

inet 10.47.0.0/12 brd 10.47.255.255 scope global weave

9: vethwe-bridge@vethwe-datapath: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1376 qdisc noqueue master weave state UP group default

### Test Kubernetes Deployment

Let us create test pod to confirm that our cluster is running as expected.

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: http-app

spec:

replicas: 3

template:

metadata:

labels:

app: http-app

spec:

containers:

- name: http-app

image: katacoda/docker-http-server:latest

ports:

- containerPort: 80

#### Deploy it to the cluster

Create a test namespace:

$ kubectl create namespace test-namespace

namespace/test-namespace created

After namespace is created, create a pod using deployment object defined earlier. **-n** is used to specify the namespace. We expect three pods to be created since our replicas value is 3.

$ kubectl create -n test-namespace -f http-app-deployment.yml

deployment.extensions/http-app created

Confirm:

$ kubectl -n test-namespace get deployments

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

http-app 3 3 3 3 1m

$ kubectl -n test-namespace get pods

NAME READY STATUS RESTARTS AGE

http-app-97f76fcd8-68pxg 1/1 Running 0 1m

http-app-97f76fcd8-f9bdk 1/1 Running 0 1m

http-app-97f76fcd8-vgmq7 1/1 Running 0 1m

You can see we have http-app deployment live.

With the deployment created, we can use kubectl to create a service which exposes the Pods on a particular port. An alternative method is defining a Service object with YAML. Below is our service definition.

$ cat http-app-service.yml

apiVersion: v1

kind: Service

metadata:

name: http-app-svc

labels:

app: http-app

spec:

type: NodePort

ports:

- port: 80

nodePort: 30080

selector:

app: http-app

Create a service using kubectl command:

$ kubectl -n test-namespace create -f http-app-service.yml

service/http-app-svc created

This service will be available on Cluster IP and port **30080.**To get cluster IP, use:

$ kubectl -n test-namespace get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

http-app-svc NodePort 10.5.45.208 <none> 80:30080/TCP 1m

### Kubernetes on Ubuntu 18.04 – Post-Installation

Enable shell autocompletion for kubectl commands.  kubectl includes autocompletion support, which can save a lot of typing!. To enable shell completion in your current session, run:

source <(kubectl completion bash)

To add kubectl autocompletion to your profile, so it is automatically loaded in future shells run:

echo "source <(kubectl completion bash)" >> ~/.bashrc

If you are using zsh edit the **~/.zshrc** file and add the following code to enable kubectl autocompletion:

if [ $commands[kubectl] ]; then

source <(kubectl completion zsh)

fi

Or when using Oh-My-Zsh, edit the ~/.zshrc file and update the plugins= line to include the kubectl plugin.

source <(kubectl completion zsh)

## Conclusion

We have successfully deployed a 3 node Kubernetes cluster on Ubuntu 18.04 LTS servers. Our next guides will cover Kubernetes HA, Kubernetes Monitoring, How to configure external storage and more cool stuff. Stay tuned!.

Commands:

kubectl describe nodes

kubectl get nodes

Dashboard:

mkdir $HOME/certs

cd $HOME/certs

openssl genrsa -out dashboard.key 2048

openssl rsa -in dashboard.key -out dashboard.key

openssl req -sha256 -new -key dashboard.key -out dashboard.csr -subj '/CN=localhost'

openssl x509 -req -sha256 -days 365 -in dashboard.csr -signkey dashboard.key -out dashboard.crt

kubectl -n kube-system create secret generic kubernetes-dashboard-certs --from-file=$HOME/certs

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml>

sudo kubectl get pods --all-namespaces

kubectl -n kube-system edit service kubernetes-dashboard

change 🡪 type: NodeIP to type: NodePort

to get dashboard port

kubectl -n kube-system get service kubernetes-dashboard

<https://www.vikki.in/kubernetes-on-ubuntu-18-04-with-dashbaoard>

<https://kubernetes.io/docs/tasks/configure-pod-container/translate-compose-kubernetes/>

<https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-volumes-example-nfs-persistent-volume.html>

<https://tachingchen.com/blog/kubernetes-assigning-pod-to-nodes/>

<https://tachingchen.com/blog/kubernetes-rolling-update-with-deployment/>

<https://medium.com/@shalithasuranga/creating-simple-shared-persistent-storage-for-micro-services-in-kubernetes-8a7af29f67de>

Running pods in particular node

<https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/>

<https://kubernetes.io/docs/concepts/overview/working-with-objects/common-labels/>

**#kubectl get nodes**

Chose one of your nodes, and add a label to it:

#**kubectl label nodes <your-node-name> disktype=ssd**

Verify that your chosen node has a **disktype=ssd** label

#**kubectl get nodes --show-labels**

This pod configuration file describes a pod that has a node selector, **disktype: ssd**. This means that the pod will get scheduled on a node that has a **disktype=ssd** label.

**Ex:**

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: nginx**

**labels:**

**env: test**

**spec:**

**containers:**

**- name: nginx**

**image: nginx**

**imagePullPolicy: IfNotPresent**

**nodeSelector:**

**disktype: ssd**

Kubernets Hostpath with local drive

<https://raw.githubusercontent.com/IBM/Kubernetes-container-service-GitLab-sample/master/kubernetes/local-volumes.yaml>

Kubernets – NFS

<https://github.com/kubernetes-incubator/external-storage/tree/master/nfs>

<https://opensource.ncsa.illinois.edu/confluence/display/~lambert8/NFS+in+Kubernetes>

<https://github.com/kubernetes-retired/nfs-provisioner>

<https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/#deploy-wordpress>

#install aws-nfs util

<https://github.com/aws/efs-utils>

AWS EFS

<https://banzaicloud.com/blog/aws_provision_efs/>

<https://github.com/banzaicloud/banzai-charts/blob/master/efs-provisioner/efs-provisioner.yaml>

Make sure your Amazon images contain the \*nfs-common\* package. If not, ssh to all the nodes and install `nfs-common` with

#sudo apt-get install nfs-common```

#wget <https://raw.githubusercontent.com/banzaicloud/banzai-charts/master/efs-provisioner/efs-provisioner.yaml>```

Edit \*efs-provisioner.yaml\* and replace the values in brackets: {FILE\_SYSTEM\_ID}, {AWS\_REGION}, {AWS\_ACCESS\_KEY\_ID}, {AWS\_SECRET\_ACCESS\_KEY} with yours.

Alternatively, instead of specifying ASW credentials, you can set up instance profile roles that allow EFS access. Apply with kubectl.

#kubectl apply -f efs-provisioner.yaml```

Run Docker composer on kubernetes

<https://kubernetes.io/docs/tasks/configure-pod-container/translate-compose-kubernetes/>

**#curl -L https://github.com/kubernetes/kompose/releases/download/v1.16.0/kompose-linux-amd64 -o kompose**

**#chmod +x kompose**

**#sudo mv ./kompose /usr/local/bin/kompose**

**Directly run docker-compose file**

**#kompose up**

**# kompose --file ./examples/docker-guestbook.yml up**

**Convert docoker-composefile file**

**#kompose --file docker-voting.yml convert**

Jenkins on kubernetes

<https://www.blazemeter.com/blog/how-to-setup-scalable-jenkins-on-top-of-a-kubernetes-cluster>

<https://rancher.com/blog/2018/2018-11-27-scaling-jenkins/>

Git on kubernetes

<https://github.com/IBM/Kubernetes-container-service-GitLab-sample/tree/master/kubernetes>

<https://blog.sonatype.com/using-nexus-3-as-your-repository-part-3-docker-images>

Nexus

<https://blog.sonatype.com/kubernetes-recipe-sonatype-nexus-3-as-a-private-docker-registry>

UI- <http://ec2-52-42-219-183.us-west-2.compute.amazonaws.com:30091/>

Login sudo docker login -u admin -p admin123 172.31.43.228:30092

Tag - docker tag vinwipro/keyclock 172.31.43.228:30092/repository/vin

Push - sudo docker push 172.31.43.228:30092/repository/vin

Pull - sudo docker pull 172.31.43.228:30092/repository/vin

UI- <http://ec2-52-42-219-183.us-west-2.compute.amazonaws.com:30091/>

sudo docker login -u admin -p admin123 54.189.195.210:30092

sudo docker push ec2-52-42-219-183.us-west-2.compute.amazonaws.com30092/repository/vin:test

sudo docker puLL ec2-52-42-219-183.us-west-2.compute.amazonaws.com:30092/repository/vin

Jenkins

<https://rancher.com/blog/2018/2018-11-27-scaling-jenkins/>

<https://rancher.com/blog/2018/2018-11-27-scaling-jenkins/>

NFS:

1st create normal claim and you will get PVC name and folder in shard drive

2nd create new PV using above pvc folder ex: path: /export/pvc-c2ac56c1-359a-11e9-ae51-06c2fb30f90e

3rd create new PVC using above PV

4th use new PVC in pod

Kubernetes on RHEL and CentOs

<https://www.linuxtechi.com/install-kubernetes-1-7-centos7-rhel7/>

<https://myopswork.com/how-to-install-kubernetes-k8-in-rhel-or-centos-in-just-7-steps-2b78331174a5>

<https://www.devtech101.com/2018/08/13/installing-a-kubernetes-1-11-cluster-on-centos-7-51804-the-manual-way-installing-3-master-nodes-etcd-part-1/>

sudo yum install -y <http://mirror.centos.org/centos/7/extras/x86_64/Packages/container-selinux-2.74-1.el7.noarch.rpm>

Deployment :

<https://betsol.com/2018/11/devops-using-jenkins-docker-and-kubernetes/>

<https://akomljen.com/set-up-a-jenkins-ci-cd-pipeline-with-kubernetes/>

<https://kublr.com/blog/using-jenkins-and-kubernetes-for-continuous-integration-and-delivery/>

<https://kumorilabs.com/blog/k8s-6-integrating-jenkins-kubernetes/>

Create kubconfig

<https://gist.github.com/innovia/fbba8259042f71db98ea8d4ad19bd708>

swap disable on redhat

nano /etc/waagent.conf

MongoDB replication and mongoDB docker imgaes

<https://github.com/bitnami/bitnami-docker-mongodb>

Kafka

<https://dzone.com/articles/kafka-setup>

<https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_kafka-component-guide/content/prepare-kafka-env.html>

<https://github.com/simplesteph/kafka-stack-docker-compose>

yml files that will produce the existing cluster

<https://github.com/kubernetes/kubernetes/issues/24873>

Services

<https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-ports-targetport-nodeport-service.html>