Kubernetes ( K8S):

<https://kubernets.io/docs/setup/production-environment/>

<https://kubernets.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

<https://kubernets.io/docs/setup/best-practices/cluster-large>

<https://kubernets.io/docs/reference/kubectl/overview/>[Harikrishna](res://\\G2MResource_en.dll/%3cA%20HREF=%22%3conLeftClick%3eeCMD_SetChatTo%20109%3c/onLeftClick%3e%3conRightClick%3eeCMD_DoAttendeeContextMenu%207143436%3c/onRightClick%3e%22%3e%3c/A%3e)

<https://www.vagrantup.com/downloads.html>

<https://www.virtualbox.org/wiki/Downloads>

gitclone

Kubernetes core Features:

1.PODS ( Production on demand service ): smallest object in Kubernetes cluster. Docker container and few more functionalities all of this encapsulated by a wrapper and called a pod inside a Kubernetes clusters. Add ip address, it generates metadata of container, can do continuous monitor, runs in a healthy environment.

2.Replication Controller: multiplies copies of a pod. It copies containers as and when needed based on need.

3.Storage Management : attach volume to a pod and store data from pod into it.

4.Resource monitoring: cpu, memory, etc. allocates resources and monitors

5.Health checks. Continuous monitor of applications, it consumes resources and checks if the pod state is healthy.

6.Networking: does not sit on existing network. creates its on POD network. ( calico, flannel etc )

7.Secret management: password, keys, access keys, stored in a encrypted db. Etcd database

8.Rolling Updates: new production releases updated without bringing down the environment.

9.Service Discovery: has many services, discovery means in the background , tracking issues, health etc. Different services all continuously trying to discover the different status of the environment.

Docker Swarm and Kubernetes comparison: ( Manager/Worker node ):

Kubernetes Architecture : Master/Worker Node:

Master Node : 1. etcd, 2. api server, 3. Scheduler, 4. controller Manager and Docker

**Etcd** : is a encrypted database which stores important Master and worker node live information.

**API Server**: is the interface between Master and Worker node.

Worker node: 1. Kubelet, 2. proxy, 3. POD1 ( container1, container2 ), POD2. Recommended 1 container per pod.

**Proxy** : it is like a proxy, Apps are hosted in a container any access it is done thru the proxy.

**Installation steps of Kubernetes: Bare metal installation**

To run containers in Kubernetes pods we need container run time. Docker ( or cri-o, containerd etc) is the runtime container. Any docker version above version 17.06 will work for k8s.

( apt-transport-https - to share any or expose any links thru secured web based protocol, ca certificates -- for all components of k8s to communicate with each other and it is ca also the source of truth and creates a secure channel, curl -- downloading, uploading binaries and any web pages, software-properties-common -- other common software properties that are needed)

Master node needs 2 cpus. Worknode 1 cpu is enough.

**Install docker and start docker service first:**

apt-get update && apt-get install -y apt-transport-https ca-certificates curl software-properties-common

apt-get update && apt-get install -y docker.io ( to install ubuntu )

systemctl enable docker.service ( to load and docker start service )

**Next Install Kubernetes key: ( all packages are available here )**

curl -s <https://packages.cloud.google.com/apt/doc/apt-get.gpg> | apt-get add -

( apt-get add – is a signature key provided by the provider ( google in this case ) to validate the packages and download it safely so there is no hacking etc )

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

( create a file Kubernetes.list and add EOF to it for a list of all the packages in it )

deb <https://apt.kubernetes.io/kubernetes-xenial> main

EOF

( the packages are downloaded from here )

apt-get update && apt-get install -y kubelet kubeadm kubectl

systemctl daemon-reload && systemctl restart kubelet

( install the 3 packages and load kubelet)( there are also other steps in Kubernetes which are not shown here and are usually not required )

**Initailiaze kubeadm, pod network and advertise ip address**

**Calico-->** kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=$(hostname -i)

**Flannel->** kubeadm init --apiserver-advertise-address=$(hostname -i) --pod-network-cidr=10.244.0.0/16

kubeadm init

----------------------------------------------------------------

Important : Good Practice copy the output like shown below and save it in a notepad.

Your Kubernetes control-plane has initialized successfully!

Config file is stored in /var/lib/kubelet/config,

Every component of k8s is binded by a cert. when one component talk to another it has to validate with cert layer ( security ) first. Cert dir is /etc/Kubernetes/pki

( certs generated : etcd/ca, etcd/server, etcd/healthcheck etc for each componenet )

Kubeconfig folder is /etc/Kubernetes

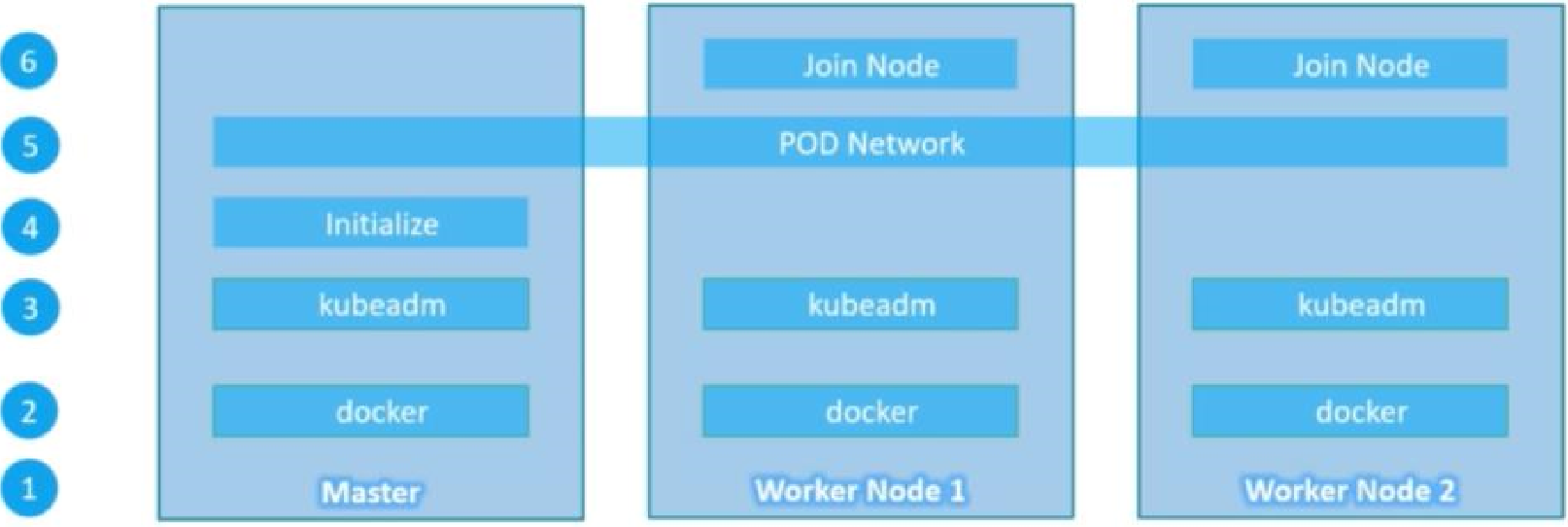
Order of creation etcd, apiserver, controller manager, scheduler

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BARE METAL Installation steps:

AWS setup.

Master Node setup:



\*\*\*\*\*\*\*\*\*\*\* Step 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Click on running instances/ Launch instance/ and select. ( this is not free but costs a few cents )**
2. **Ubuntu Server 18.04 LTS (HVM), 2 cpu SSD Volume Type - ami-07c1207a9d40bc3bd (64-bit x86) ( 2 cpu )**
3. **Configure port by adding 6443 and /0.0.0.0/0 . This default k8s port. ( docker default port is 2347 )**
4. **Select and click review and launch**
5. **Name node as Master Node.**
6. **Use your key pair from local driver to as key**
7. **Once instance is running – click on Connect button and copy ssh command string.**

**Worker Node:**

1. **Click on running instances/ Launch instance/ free tier/and select.**
2. **Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-07c1207a9d40bc3bd (64-bit x86) ( 2 cpu )**
3. **Select 2 instances as we need to worker nodes.**
4. **Configure port by adding 6443 and /0.0.0.0/0 . This default k8s port. ( docker default port is 2347 )**
5. **Select and click review and launch**
6. **Name nodes Worker1 and worker2 nodes**
7. **Use your key pair from local driver to as key**
8. **Once instance is running – click on Connect button and copy ssh command string.**

**Do the following steps for three terminals for all three nodes .**

1. **Now run Mobaxterm: and execute the following commands**
2. **/home/mobaxterm cd desktop/key**
3. **desktop/key chmod 400 MD.pem**
4. **desktop/key ssh -i "MD.pem"** [**ubuntu@ec2-13-59-158-14.us-east-2.compute.amazonaws.com**](mailto:ubuntu@ec2-13-59-158-14.us-east-2.compute.amazonaws.com)
5. **sudo su -**

# Next step 2 and 3 ( 10 commands ) for all three nodes:

**\*\*\*\*\*\*\*\*\*Step 2\*\*\*\*\*\*Docker - 4 commands\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**apt-get update && apt-get install -y apt-transport-https ca-certificates curl software-properties-common**

**apt-get update && apt-get install -y docker.io**

**systemctl enable docker.service**

**docker –version**

**\*\*\*\*\*\*\*\*Step 3\*\*\*\*\*Kubeadm – 6 commands\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

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

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

**deb https://apt.kubernetes.io/ kubernetes-xenial main**

**EOF**

**apt-get update && apt-get install -y kubelet kubeadm kubectl**

systemctl daemon-reload && systemctl restart kubelet

\*\*\*\*\*\*\*\*\*\*\*\*Step 4- Initialize – 1 command – Master Node Only\*\*\*\*

Next Initialize just Master node only : Next 5 steps Master Node Only:

**kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=$(hostname -i)**

**\*\*\*\*\*\*\*\*\*\*\*\*Step 5--POD Network - 4 commands—All nodes\*\*\*\*\*\*\*\***

**mkdir -p $HOME/.kube**

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

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

**kubectl apply -f** [**https://docs.projectcalico.org/v3.8/manifests/calico.yaml**](https://docs.projectcalico.org/v3.8/manifests/calico.yaml)

**( deploy a pod network to the cluster ( calico or flannel command )**

**\*\*\*\*\*\*\*\*\*Step 6\*\*\*\*\*\*\*\*\*Join Node – 1 command—Worker Node\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Worker node command to join the cluster :**

**Then you can join any number of worker nodes by running the following on each as root:**

**kubeadm join 172.31.64.60:6443 --token lk3wg1.0cywm02z7xauovqs \**

**--discovery-token-ca-cert-hash sha256:f30553444716c075d2ee08f904a408cd210507fba7134659bc7b75a777071762**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Master Node commands to create PODS:**

**Kubectl run my-pod --image nginx**

**Kubectl run demo --image redis**

**NOTES from example :**

**Master Node ip: 172.31.67.103**

**Worker node Node ip: 172.31.48.195**

**Pod IP: 10.244.1.2**

**Container ID: my-container-1(name) or docker://** a5c8**53444716c075d2ee08f904a408cd210507fba7134659bc7b75a777071762**

**my-container-2(name) or docker://** b4ba751**22815n2eddde08f904a4d08cd210507fba7134659bc7b75a745746233**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Master Node commands to delete a pod:**

**Kubectl delete pod demo**

**Kubectl delete my-pod**

**Kubectl get pods ( returns nothing now )**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**KUBECTL is a master node command- only works in master node**

**kubectl cluster-info dump ( investigate for error )**

**Kubectl cluster-info**

**Kubectl version**

**Kubectl version --short**

**Kubectl get node(s)**

**Kubectl get pod(s)**

**Kubectl get pod -o wide (for pod and node details )**

**Kubectl get pod --output wide**

**Kubectl get service(s) (master node created this service of type cluster ip)**

**Kubectl describe pod my-pod**

**Kubectl describe service <service-name>**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Automation of POD creation in Kubernetes:**

**There are few kinds of k8s resources ( objects ) – important**

**These are the 4 apiVersion options in YAML file**

**-POD :v1**

**-ReplicaSet :apps/v1 ( scale up )**

**-Deployment :apps/v1**

**-Services :v1**

**Types of Services:**

**--ClusterIp ( default )**

**--NodePort**

**--Load Balancer**

**--ExternalName**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Automation is done thru YAML file ( yet another markup language , which is a key-value pair )**

**4 properties of Kubernetes YAML file :**

**( data of data is metadata, who is creating the pod, name of the pod, other artifacts like label ( frontend ), category (db) etc )**

**Whatever is the label name selector uses that , in the below example my label is type.**

**Always recommended to have only 1 container per pod (1 to 1 mapping ). As it is easy to manage and do load balancing for Kubernetes.**

**Translating the following Kubernetes run command into an YAML file**

**$kubectl run my-pod –image nginx**

**Create a YAML file in master node with the following commands in it**

**$vi pod.definition.yaml ( or pod.manifest.yaml std naming conventions )**

**---**

**apiVersion: v1**

**kind: pod**

**metadata:**

**annotations:**

**on-call: "“Madhu Duggi-2488941002”"**

**support: “support@tw.com”**

**labels:**

**app: myfirst-pod**

**type: front-end**

**name: myfirst-pod**

**spec:**

**containers:**

**-**

**image: nginx**

**name: my-container-1**

**-**

**image: redis**

**name: my-container-2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pod** | **ReplicaSet** | **Development** | **Services** |
| ***apiVersion: v1***  **kind: pod**  **metadata:**  **name: myfirst-pod**  **labels:**  **app: myfirst-pod**  **type: front-end**  **annotations:**  **on-call: "“MD-2488941002”"**  **support: “support@tw.com”**  **spec:**  **containers:**  **-**  **image: nginx**  **name: my-container-1**  **-**  **image: redis**  **name: my-container-2** | **apiVersion: apps/v1**  **kind: RelicaSet**  **metadata:**  **name: my-rs**  **labels:**  **app: my-rs**  **type: front-end**  **spec:**  **replicas: 3**  **template:**  **metadata:**  **name: myfirst-pod**  **labels:**  **app: myfirst-pod**  **type: front-end**  **annotations:**  **on-call: "prod@support.com"**  **dependencies: "Python 3.6"**  **spec:**  **containers:**  **- name: my-container-1**  **image: nginx**  **selector:**  **matchLabels:**  **type: front-end** | **apiVersion: apps/v1**  **kind: RelicaSet**  **metadata:**  **name: my-rs**  **labels:**  **app: my-rs**  **type: front-end**  **spec:**  **replicas: 3**  **template:**  **metadata:**  **name: myfirst-pod**  **labels:**  **app: myfirst-pod**  **type: front-end**  **annotations:**  **on-call: "prod@support.com"**  **dependencies: "Python 3.6"**  **spec:**  **containers:**  **- name: my-container-1**  **image: nginx**  **selector:**  **matchLabels:**  **type: front-end** |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **apiVersion:** | **v1** | **Apps/v1** | **Apps/v1** | **V1** |  |  |  |  |  |
| **kind:** | **Pod** | **ReplicaSet** | **Development** | **Services** |  |  |  |  |  |
| **metatdata:** |  |  |  |  |  |  |  |  |  |
|  | **name: my-rs** |  |  |  |  |  |  |  |  |
|  | **labels:** |  |  |  |  |  |  |  |  |
|  |  | **app: my-rs** |  |  |  |  |  |  |  |
|  |  | **type: front-end** |  |  |  |  |  |  |  |
|  | **annotations:** |  |  |  |  |  |  |  |  |
|  |  | **on-call: “1002”** |  |  |  |  |  |  |  |
|  |  | **support: “MD@TW”** |  |  |  |  |  |  |  |
| **spec:** |  |  |  |  |  |  |  |  |  |
|  | **~~containers:~~** |  |  |  |  |  |  |  |  |
|  | **replicas: 3** |  |  |  |  |  |  |  |  |
|  | **template:** |  |  |  |  |  |  |  |  |
|  |  | **metatdata:** |  |  |  |  |  |  |  |
|  |  |  | **name: myfirst-pod** |  |  |  |  |  |  |
|  |  |  | **labels:** |  |  |  |  |  |  |
|  |  |  |  | **app: myfirst-pod** |  |  |  |  |  |
|  |  |  |  | **type: front-end** |  |  |  |  |  |
|  |  |  | **annotations:** |  |  |  |  |  |  |
|  |  |  |  | **On-call: “MD”** |  |  |  |  |  |
|  |  |  |  | **Dependencies: “Java”** |  |  |  |  |  |
|  |  | **spec:** |  |  |  |  |  |  |  |
|  |  |  | **containers:** |  |  |  |  |  |  |
|  |  |  | **-** | **name: my-container-1** |  |  |  |  |  |
|  |  |  |  | **image: nginx** |  |  |  |  |  |
|  | **selector:** |  |  |  |  |  |  |  |  |
|  |  | **matchLabels:** |  |  |  |  |  |  |  |
|  |  |  | **type: front-end** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**To create the pod(s) now use thee following command:**

**Kubectl create -f pod.definition.yaml**

**Kubectl apply -f pod.definition.yaml ( new command instead of create )**

**Kubectl get pod**

**Kubectl describe pod myfirst-pod**

**Kubectl get pods –watch ( Live status )**

**Kubectl get nodes -w ( live status )**

**Kubectl get pods –show-labels**

**Kubectl get pods -l app=myfirst-pod ( -l is label )**

**Kubectl get pods -l type-front-end**

**kubectl logs -p myfirst-pod -c my-container-1**

**You can delete with the same YAML file with following command:**

**Kubectl delete -f pod.definition.yaml**

ReplicaSet :

ReplicaSet YAML file :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

apiVersion: apps/v1

kind: RelicaSet

metadata:

name: my-rs

labels:

app: my-rs

type: front-end

spec:

replicas: 3

template:

metadata:

name: myfirst-pod

labels:

app: myfirst-pod

type: front-end

annotations:

on-call: "prod@support.com"

dependencies: "Python 3.6, Harbor, NodeJS"

spec:

containers:

- name: my-container-1

image: nginx

selector:

matchLabels:

type: front-end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Kubectl scale --replicas=8 replicaset my-rs

Kubectl get pods -o wide

Kubectl delete pod my-rs-p828p

Kubectl get replicaset

Kubectl delete replicaset my-rs