ec2-instance(Elastic Complete Cloud)

Types of ec2 instances:

Micro instance,General Purpose,Compute Optimized,Memory Optimized,Accelerated Computing Optimized,Storage Optimized,Gpu optimized.

Images id is different for each region for same IMI image.

User Data: This is a code to run when ec2 instance is launching.

If restarted ip address will not change. If shut down and start ip address will change.

If terminated cpu and ram will not charged. Only Storage will charge.

Tags will help to inspector to check security reached certain expedition.

Ec2 default storage ebs.

**Based on pricing:**

[Amazon EC2 Spot Instances](https://aws.amazon.com/ec2/spot/),

[Amazon EC2 Reserved Instances](https://aws.amazon.com/ec2/pricing/reserved-instances/),

[Amazon EC2 Dedicated Hosts](https://aws.amazon.com/ec2/dedicated-hosts/),

[Amazon EC2 Dedicated Instances](https://aws.amazon.com/ec2/purchasing-options/dedicated-instances/).

We can use ec2 instance as NAT instance.

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

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

s3:(OBJECT STORAGE)

Visibly it is to be global but it is regional specific.

unique name in entire aws in the world.

Cross account access is work but policy should describe.

complete static website we can host.

We can enable logs in s3 buckets.

We can create events , ex any thing changed in s3 buckets this event will trigger.(SNS(Simple Notification Service),LAMBDA FUNCTION(this is a function when specific event occurs) ,sqsque)

Enabling versioning , We can upload multiple files with same name. After enabling we cannot disable versioning ,we can only suspend it only.

Storage classes: standard storage(3 replicas in), standard-infrequently(3 replicas), Reduced redundancy(2 replicas).

We can create action based or requirements. This is called life cycle policies.

To enable cross region replication must and should we have to enable versioning.

Transfer Acceleration : This will enables us to upload objects to long distance buckets. This will create one endpoint near us . We will upload our objects to near endpoint . From near endpoint it goes throw amazon network. This will provide us secure.

We can access s3 buckets with get,and put requests from browser. We can access from cli by using Access key id, Secret key id .

Once object max size 5tb. In one bucket we can store n number of objects.

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

VPC(Virtual Private Cloud)

When creating AWS account we will get default vpc in our account. Same way default subnet also present.

Default VPC can be deleted, If want again contact amazon customer care.

Dedicated/default: Dedicated the resource to us. Default means sharing the resource with isolated.

VPC is region specific.

DHCP will asign ip address to Resources.

Availability zones to save from disaster.

VPC service is free.

By default one route table is created for every vpc. This route table(default route table) connected with IGW.(internet Gate Way)

VPC-Peering

We can connect multiple VPC with VPC peering.

With VPC-peering we can use one VPC resources(ex : ec2) in another VPC.

This can be possible in same account and different account.

This can be possible in same region with different region.

Transitive peering is not supporting.

Example: If vpc1 is peering with vpc2 , vpc2 is peering with vpc3 .

Here vpc1 resource cannot used by vpc 3.This is called Transient peering, This is not possible.

While peering VPC ,CYDER block range will not the same

Process: Assume Two vpc are there VPC1 and VPC2.

Modify the route table of one VPC by placing another VPC ' s CIDR block. Do this for both VPC.

**Subnets**

5 ip addresses are required for amazon internal use, When creating subnets.

Private Subnet: IGW not attached

Public Subnet: IGW attached.

If new subnet is created that new sub net is associated with default route table.

We can provide internet to private sub net with one way internet with using NAT instance,NAT gate way

Difference between NACL & SUCURITY GROUP.

|  |  |
| --- | --- |
| This is virtual fire wall of the sub net.  This is applicable to subnet level.  Statless ( Inbound rule cannot be remembered)  We can control ip level blocking  By default every subnet has it own NACL.  This will allow all traffic inside and out side.  Only one NACL is allowed for one subnet.  It can control in bound and out bound.  Example: This will accept http trafic inside but it can stop outbound trafic of http.  Note: Rule format should be 100,200,300.Because we can add another rule between tease numbers.  Rule will be applicable from less number to higher number. | This is virtual fire wall for ec2,elb,alb,AutoScalling Groups.  Statefull (In bound rules will remember)  No ip level control.  At least one security group is required for one ENA(Electronic network interface). Fore one ENI max 5 security groups can be attached.  This controls Inbound but cannot control out bounds.  Example: This will accept http trafic inside but it cannot stop outbound trafic of http.  Note: If it accepts http request ,it automatically accept out traffic of http. |

NAT-Gate-Way

This is always in public subnet.

Elastic Ip address is required for Nat-Gate Way,

It just forward the traffic from/to the private subnet. It will scale upto 45GBps.

Nat-gate way took private and public ip address.

Bandwidth is maintained by amazon.

Private ip address will assign from available subnet range.

This is best for production.

SSh is not possible because this is not a machine.

Shut down is not possible.

Bill will be calculated based on amount of date transferred.

We cannot configure security groups, But this gate way is in Public subnet so NACL(Subnet level) rule will applicable.

Port forwarding is not possible.

All software is maintained by amazon.

NAT-Instance

Nat instance is a machine(ec2-instance).

This bandwidth is depending on instance type which we selected.

We can change the elastic ip address to this instance.

We can configure security group because this is ec2 instance(machine in cloud).

Best for testing purpose , Because if bandwidth wants to increase we should shut down and start another NAT-instance(ec2) .

**Horizontal scaling** means that you **scale** by adding more machines into your pool of resources.

Vertical Scaling means increasing CPU and RAM to existing machine.

We can stop/start this NAT-instance.

SSH is possible to this machine.

Data coming from cloud is free and data coming from out side world is chargeable.

Port forwarding is possible.

Load Balancers :Application Load Balancers(ALB),Network Load Balancers.(NLB),Elastic Load Balancers.(ELB)

Application Load Balancer: (layer 7)

This will support http,https. This is specific to vpc. Used to micro services.

AlB follows context path ec2 based routing .

Target Group: Logical group of servers associated with your load balancer.

Based on rules request will forwarded to certain target group. In target group request is processed and response will given to load balancers.

Existing machine. We can attach security group to ALB.

LISTNER1

We can enable termination protection to ALB.

EC2

ALB

We can enable access logs. We can enable multiple request types.

EC2

EC22

EC2

LISTNER2

States Of ALB: Provisioning, active, Free

ELASTIC LOAD BALANCER(ELB)

This is specific to vpc. But we can do with multiple regions with the help of route53.

ELB can secured with security group.

All the servers contain same content.

Load will be distributed among all servers.

If will perform health checks then it will send traffic to healthyresource.

Unhealthy Thresh hold: This is a numeric value we have to give in ELB configuration .

example: If value is 2 , If two conjugative health checks fails. This resource will mark as unhealthy and resource will remove from the service.

Healthy Thresh hold: This is a numeric value we have to give in ELB configuration

example: If value is 2 , If two conjugative health checks pass. This resource will mark as unhealthy and resource will remove from the service.

Cross Zone Load Balancing: Load will distributed equally among the zones.

Connection Draining: Time to complete active request to the servers. After completing active requests LB will remove server from the service.

We can access load balancers with end points.

Enable Load Balancer Generated cookie: Request from specific Clint is bound with same server .

All request will goes to same server until the specific time period.

Load Balancers follows round Robin algorithm.

Auto Scalling

launch Configuration: This is a template this contains all configuration for new resource.

1)AMI 2)Security Group 3)volume type 4)IAM role(some time ec2instance will talk to s3 so its required) 5) Product Key

It can be integrated with load balancers

Alarem is required for auto scalling

Increase group size decrease group size required alarm:

Example: If cpu utilization is increase more than 50%es alarm fires , If alarm fires instance will up based on launch configuration.

We can send notifications .

Desired capacity will change based on policies.

Stright options to test increase group size, decrease group size we can manually click the button and check.

We can schedule auto scale capacity.

EBS

Elastic Block Storage:

5types 3 types we can use as boot devise.

1)General Purpose

2)Provisioned IOPS These will work as boot devise.

3) Magnetic

4)Cold HDD These are not used for boot volume.

5)Throw put optimized.

We can increase EBS volume when ec2 instance is running , but we cant decreased the EBS volume.

Multiple EBS can attach with single instance (ec2)

EFS

This is like NFS in lan.

This disk can be used across network, across region,

It can be mounted to a premise server as well (OVER VPN (or) Direct Connect.

Can be mounted to multiple ec2 instance at a time.

Sizening is not required .(It scales Automatically)

terraform:(0.11.3)

Aws Authentication types in terra form.

1) static credentials : These are mention in the terraform file itself(hard coded).

2)Environment Variable: These credentials are exported in current machine.

3) shared credentials : These are written in another files and refer in this main terraform file.

4)Ec2 roles.

Infrastructure as a code.

support planning from execution.

Terraform state files: Every run of terrafom, state of the terraform will store in statefiles

1).tfstate

2).tfstat backup

To Access the attribute of the terraform we us below syntax.

Syntax: $(resource\_type.logicalname.parameter\_name)

example:

Terraform will identify what resource is created in earlier execution.

Terraform will not change the ec2instance key ,it will remove and recreate entire ec2 instance.

terraform config: To provide accesskeyid and secretaccess key to terraform engine.

terraform plan : This will display what changes to b apply on environment.

terraform init : To update all plug-in to connect providers.

terraform apply : To apply the terraform to executing in Infrastructure .

terraform fmt : To check the code properly alligned or not.

terraform console: To check interpolation(replacing the values) out put in a console.

terraform import : To get resource into terraform control from outside

Terraform support different types of datatypes.

string : "sample\_string"

List : ["sanjay" ,"Thanvik", "vaishnavi", " vhinnu"]

Map : {

key1=value

key2=value

}

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

Variable Declaration in terraform :

String:

variable "region" {

type="string"

default="south\_india"

}

List:

variable "fruits" {

type="List"

default=["mango" ,"apple","pinapple"]

} Map:

varaible "imageid" {

type="Map"

default={

"andhrapradesh"="amaravathi"

"telengana" = "hyderabad"

}

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Input variables of a terraform is :

varaible "imageid" {}: This will expect input from user or machine.

output variable of a terraform is:

output "ami\_name"{

value="sanjaykumarreddy"

}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

To using variable values : ${vars.imageid} ,${vars. fruits},${vars.region}

To loop the code we can use : count variable.

length(list): This method will take list and return length of the list.

datasource: This will useful when dynamic values required at the time of running.

lookupmethod(map,key): This method will return the value from map with specific key .

**templates**: Templates is structured file for reusing by reusing in terraform we can replace the values with our values.

Modules: Collection of terraform templates .

To import the modules:

modules "samole\_module"{

source="path\_of\_the\_module(folder)"

}

After importing we can use all variables in that modules by using below syntax.

module.modulename.variable\_name

# How to Install Kubernetes (k8s) 1.7 on CentOS 7 / RHEL 7

by [Pradeep Kumar](https://www.linuxtechi.com/author/pradeep/" \o "Posts by Pradeep Kumar) · Published September 4, 2017 · Updated December 12, 2017

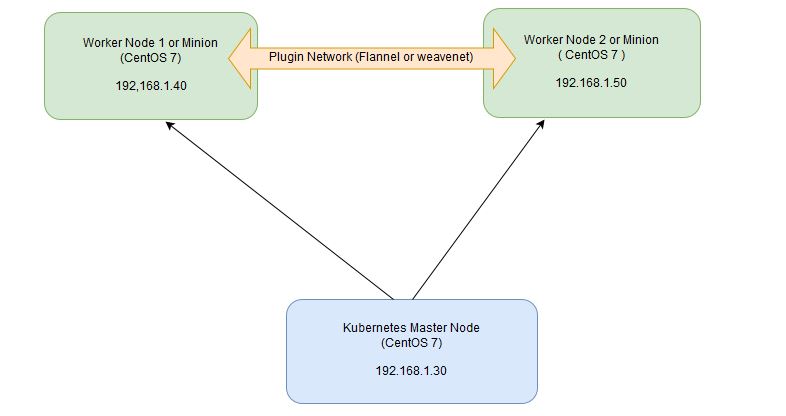
Kubernetes is a **cluster** and **orchestration** engine for docker containers. In other words Kubernetes is  an open source software or tool which is used to orchestrate and manage docker containers in cluster environment. Kubernetes is also known as k8s and it was developed by Google and donated to “Cloud Native Computing foundation”

In Kubernetes setup we have one master node and multiple nodes. Cluster nodes is known as worker node or Minion. From the master node we manage the cluster and its nodes using ‘**kubeadm**‘ and ‘**kubectl**‘  command.

Kubernetes can be installed and deployed using following methods:

* Minikube ( It is a single node kubernetes cluster)
* Kops ( Multi node kubernetes setup into AWS )
* Kubeadm ( Multi Node Cluster in our own premises)

In this article we will install latest version of Kubernetes 1.7 on CentOS 7 / RHEL 7 with kubeadm utility. In my setup I am taking three CentOS 7 servers with minimal installation. One server will acts master node and rest two servers will be minion or worker nodes.

[](https://www.linuxtechi.com/wp-content/uploads/2017/09/Kubernetes-settup-Diagram.jpg)

#### On the Master Node following components will be installed

* **API Server**  – It provides kubernetes API using Jason / Yaml over http, states of API objects are stored in etcd
* **Scheduler**– It is a program on master node which performs the scheduling tasks like launching containers in worker nodes based on resource availability
* **Controller Manager** – Main Job of Controller manager is to monitor replication controllers and create pods to maintain desired state.
* **etcd** – It is a Key value pair data base. It stores configuration data of cluster and cluster state.
* **Kubectl utility** – It is a command line utility which connects to API Server on port 6443. It is used by administrators to create pods, services etc.

#### On Worker Nodes following components will be installed

* **Kubelet** – It is an agent which runs on every worker node, it connects to docker  and takes care of creating, starting, deleting containers.
* **Kube-Proxy** – It routes the traffic to appropriate containers based on ip address and port number of the incoming request. In other words we can say it is used for port translation.
* **Pod** – Pod can be defined as a multi-tier or group of containers that are deployed on a single worker node or docker host.

### Installations Steps of Kubernetes 1.7 on CentOS 7 / RHEL 7

**Perform the following steps on Master Node**

#### Step 1: Disable SELinux & setup firewall rules

Login to your kubernetes master node and set the hostname and disable selinux using following commands

~]# hostnamectl set-hostname 'k8s-master'

~]# exec bash

~]# setenforce 0

~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux

Set the following firewall rules.

[root@k8s-master ~]# firewall-cmd --permanent --add-port=6443/tcp

[root@k8s-master ~]# firewall-cmd --permanent --add-port=2379-2380/tcp

[root@k8s-master ~]# firewall-cmd --permanent --add-port=10250/tcp

[root@k8s-master ~]# firewall-cmd --permanent --add-port=10251/tcp

[root@k8s-master ~]# firewall-cmd --permanent --add-port=10252/tcp

[root@k8s-master ~]# firewall-cmd --permanent --add-port=10255/tcp

[root@k8s-master ~]# firewall-cmd --reload

[root@k8s-master ~]# modprobe br\_netfilter

[root@k8s-master ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

**Note:** In case you don’t have your own dns server then update /etc/hosts file on master and worker nodes

192.168.1.30 k8s-master

192.168.1.40 worker-node1

192.168.1.50 worker-node2

#### Step 2: Configure Kubernetes Repository

Kubernetes packages are not available in the default CentOS 7 & RHEL 7 repositories, Use below command to configure its package repositories.

[root@k8s-master ~]# cat <<EOF > /etc/yum.repos.d/kubernetes.repo

> [kubernetes]

> name=Kubernetes

> baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

> enabled=1

> gpgcheck=1

> repo\_gpgcheck=1

> gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg

>         https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

> EOF [root@k8s-master ~]#

#### Step 3: Install Kubeadm and Docker

Once the package repositories are configured, run the beneath command to install kubeadm and docker packages.

[root@k8s-master ~]# yum install kubeadm docker -y

Start and enable kubectl and docker service

[root@k8s-master ~]# systemctl restart docker && systemctl enable docker

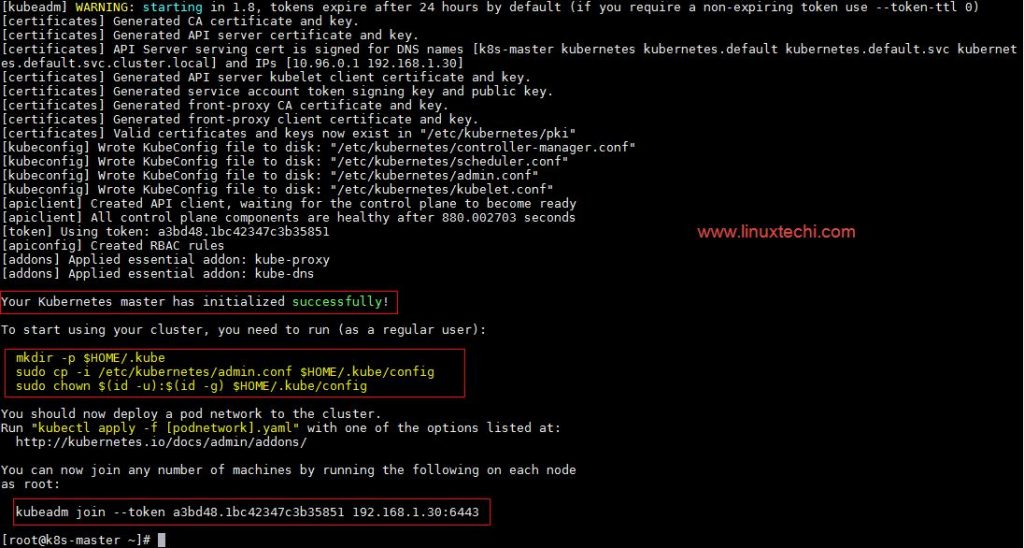
[root@k8s-master ~]# systemctl  restart kubelet && systemctl enable kubelet

#### Step 4: Initialize Kubernetes Master with ‘kubeadm init’

Run the beneath command to  initialize and setup kubernetes master.

[root@k8s-master ~]# kubeadm init

Output of above command would be something like below

[](https://www.linuxtechi.com/wp-content/uploads/2017/09/kubeadm-init-output.jpg)

As we can see in the output that kubernetes master has been initialized successfully. Execute the beneath commands to use the cluster as root user.

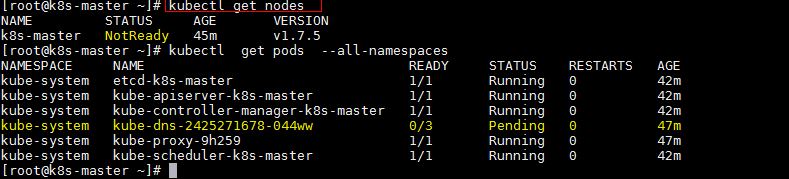
[root@k8s-master ~]# mkdir -p $HOME/.kube

[root@k8s-master ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@k8s-master ~]# chown $(id -u):$(id -g) $HOME/.kube/config

#### Step 5: Deploy pod network to the cluster

Try to run below commands to get status of cluster and pods.

[](https://www.linuxtechi.com/wp-content/uploads/2017/09/kubectl-get-nodes.jpg)

To make the cluster status ready and kube-dns status running, deploy the pod network so that containers of different host communicated each other.  POD network is the overlay network between the worker nodes.

Run the beneath command to deploy network.

[root@k8s-master ~]# export kubever=$(kubectl version | base64 | tr -d '\n')

[root@k8s-master ~]# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"

serviceaccount "weave-net" created

clusterrole "weave-net" created

clusterrolebinding "weave-net" created

daemonset "weave-net" created

[root@k8s-master ~]#

Now run the following commands to verify the status

[root@k8s-master ~]# kubectl get nodes

NAME         STATUS    AGE       VERSION

k8s-master   Ready     1h        v1.7.5

[root@k8s-master ~]# kubectl  get pods  --all-namespaces

NAMESPACE     NAME                                 READY     STATUS    RESTARTS   AGE

kube-system   etcd-k8s-master                      1/1       Running   0          57m

kube-system   kube-apiserver-k8s-master            1/1       Running   0          57m

kube-system   kube-controller-manager-k8s-master   1/1       Running   0          57m

kube-system   kube-dns-2425271678-044ww            3/3       Running   0          1h

kube-system   kube-proxy-9h259                     1/1       Running   0          1h

kube-system   kube-scheduler-k8s-master            1/1       Running   0          57m

kube-system   weave-net-hdjzd                      2/2       Running   0          7m

[root@k8s-master ~]#

Now let’s add worker nodes to the Kubernetes master nodes.

### Perform the following steps on each worker node

#### Step 1: Disable SELinux & configure firewall rules on both the nodes

Before disabling SELinux set the hostname on the both nodes as ‘worker-node1’ and ‘worker-node2’ respectively

~]# setenforce 0

~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux

~]# firewall-cmd --permanent --add-port=10250/tcp

~]# firewall-cmd --permanent --add-port=10255/tcp

~]# firewall-cmd --permanent --add-port=30000-32767/tcp

~]# firewall-cmd --permanent --add-port=6783/tcp

~]# firewall-cmd  --reload

~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

#### Step 2: Configure Kubernetes Repositories on both worker nodes

~]# cat <<EOF > /etc/yum.repos.d/kubernetes.repo

> [kubernetes]

> name=Kubernetes

> baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

> enabled=1

> gpgcheck=1

> repo\_gpgcheck=1

> gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg

>         https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

> EOF

#### Step 3: Install kubeadm and docker package on both nodes

[root@worker-node1 ~]# yum  install kubeadm docker -y

[root@worker-node2 ~]# yum  install kubeadm docker -y

Start and enable docker service

[root@worker-node1 ~]# systemctl restart docker && systemctl enable docker

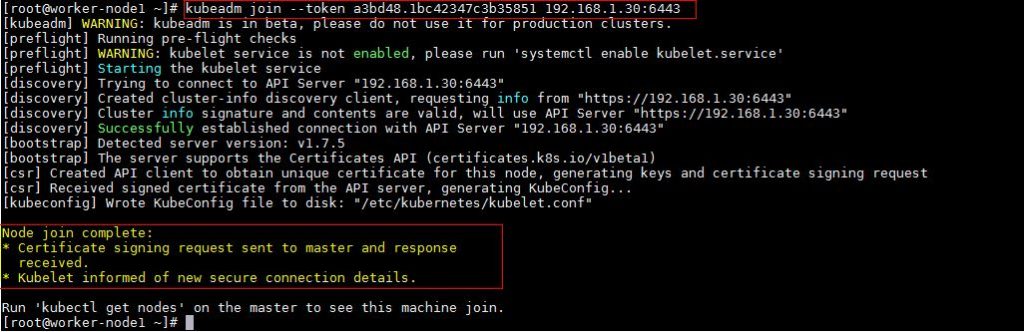
[root@worker-node2 ~]# systemctl restart docker && systemctl enable docker

#### Step 4: Now Join worker nodes to master node

To join worker nodes to Master node, a token is required. Whenever kubernetes master initialized , then in the output we get command and token.  Copy that command and run on both nodes.

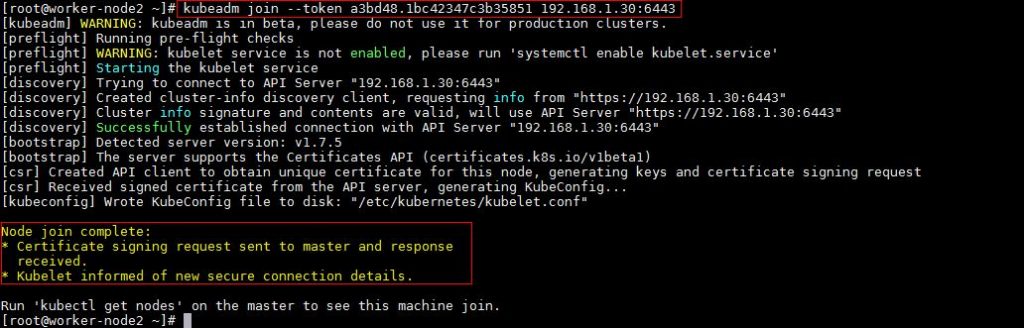
[root@worker-node1 ~]# kubeadm join --token a3bd48.1bc42347c3b35851 192.168.1.30:6443

Output of above command would be something like below

[](https://www.linuxtechi.com/wp-content/uploads/2017/09/kubeadm-node1.jpg)

[root@worker-node2 ~]# kubeadm join --token a3bd48.1bc42347c3b35851 192.168.1.30:6443

Output would be something like below

[](https://www.linuxtechi.com/wp-content/uploads/2017/09/kubeadm-join-node2.jpg)

Now verify Nodes status from master node using kubectl command

[root@k8s-master ~]# kubectl get nodes

NAME           STATUS    AGE       VERSION

k8s-master     Ready     2h        v1.7.5

worker-node1   Ready     20m       v1.7.5

worker-node2   Ready     18m       v1.7.5

[root@k8s-master ~]#

As we can see master and worker nodes are in ready status. This concludes that kubernetes 1.7 has been installed successfully and also we have successfully joined two worker nodes.  Now we can create pods and services.