**Kubernetes Course Content**

Kubernetes Basic Idea

Kubernetes Architecture

Kubernetes Installation and Creation of PODS , Nodes

Deployments and Services

Creation of Elastic Search

Creation of Application to Welcome the Kubernetes

Nginx and Ingress controllers

Creation and Configuring the Nginx, Services

Creation of Load Balancer

kubeadm: the command to bootstrap the cluster.

Kubeadm is a tool used **to build Kubernetes (K8s) clusters**. Kubeadm performs the actions necessary to get a minimum viable cluster up and running quickly. By design, it cares only about bootstrapping, not about provisioning machines (underlying worker and master nodes)

kubelet: the component that runs on all of the machines in your cluster and does things like starting PODs and containers.

kubectl: the command line until to talk to your cluster.

**A 3-Node Cluster Setup Using kubeadm, kubelet, and kubectl**

**Prerequisites**

* 3 virtual machines (VMs) up and running in the same subnet / VPC. We are using EC2 instances on the AWS Cloud for this how-to guide. One of the VMs need to have 2 vCPUs and 4 GB RAM, which means at least a t2.medium instance type. Rest of the two VMs can be t2.micro instances.
* **Ubuntu 20.04 LTS** OS on all machines.
* Familiarity with the [Kubernetes Components](https://kubernetes.io/docs/concepts/overview/components/).

**Part A - Controller and Worker Nodes**

Run these commands on all the VMs. The t2.medium VM is designated as the controller (master) node and the t2.micro VMs are designated as the worker nodes.

1. $ sudo apt-get update
2. $ sudo apt-get install docker.io
3. $ sudo apt-get update
4. $ sudo apt-get install -y apt-transport-https ca-certificates curl
5. $ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
6. $ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
7. $ sudo apt-get update
8. $ sudo apt-get install -y kubelet kubeadm kubectl
9. $ sudo apt-mark hold kubelet kubeadm kubectl

**Part B - Controller Node ONLY**

Run these commands only on the VM designated as the controller (master) node.

1. (RUN AS ROOT) Initiate API server:
2. $ sudo -s
3. # kubeadm init --apiserver-advertise-address=<ControllerVM-PrivateIP> --pod-network-cidr=192.168.0.0/16
4. # exit
5. (RUN AS NORMAL USER) Add a user for kube config:
6. $ mkdir -p $HOME/.kube
7. $ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
8. $ sudo chown $(id -u):$(id -g) $HOME/.kube/config
9. (RUN AS NORMAL USER) Deploy Calico network:
10. $ curl https://docs.projectcalico.org/manifests/calico-typha.yaml -o calico.yaml
11. $ kubectl apply -f calico.yaml
12. (RUN AS ROOT) Create cluster join command:
13. $ sudo -s
14. # kubeadm token create --print-join-command
15. (SAMPLE OUTPUT - DO NOT USE)
16. kubeadm join 172.31.11.170:6443 --token bv79oe.4z4yb8w0cxcfiv23 --discovery-token-ca-cert-hash sha256:153bab87f30a3f84264b6455b07ec01e038f7a8e7fed9055b21a634d8e1d5699

**Part C - Worker Nodes ONLY**

Copy the output of the cluster join command from the previous step and use on the VMs designated as the worker nodes.

(EXAMPLE COMMAND - DO NOT USE):

$ kubeadm join 172.31.11.170:6443 --token bv79oe.4z4yb8w0cxcfiv23 --discovery-token-ca-cert-hash sha256:153bab87f30a3f84264b6455b07ec01e038f7a8e7fed9055b21a634d8e1d5699

The Kubernetes cluster is now configured. Repeat **Part C** for additional worker nodes.