Session_1

Initialise a Cluster

Become root

on all the machines, get admin privileges by running the below command:
sudo su -

Bootstrap the cluster

Om Master machine:
```
kubeadm init --node-name control-plane
```

output:

```
root@ip-172-31-11-237:~# kubeadm init --node-name control-plane
I1028 12:57:06.643391   10355 version.go:254] remote version is much newer: v1.22.3;
falling back to: stable-1.20
[init] Using Kubernetes version: v1.20.12
[preflight] Running pre-flight checks
	[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver.
The recommended driver is "systemd". Please follow the guide at
https://kubernetes.io/docs/setup/cri/
	[WARNING SystemVerification]: this Docker version is not on the list of validated
versions: 20.10.7. Latest validated version: 19.03
	[WARNING Hostname]: hostname "master" could not be reached
	[WARNING Hostname]: hostname "master": lookup master on 127.0.0.53:53: server
misbehaving
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet
connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default
kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs [10.96.0.1
172.31.11.237]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
```

```
[certs] etcd/server serving cert is signed for DNS names [localhost master] and IPs
[172.31.11.237 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs
[172.31.11.237 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from
directory "/etc/kubernetes/manifests". This can take up to 4m0s
[kubelet-check] Initial timeout of 40s passed.
[apiclient] All control plane components are healthy after 59.002483 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the
"kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.20" in namespace kube-system with the
configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master as control-plane by adding the labels
"node-role.kubernetes.io/master=''" and "node-role.kubernetes.io/control-plane='' (deprecated)"
[mark-control-plane] Marking the node master as control-plane by adding the taints
[node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: qafdgh.zj8seb6bky5m2zqa
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in
order for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically
approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client
certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client
certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy
```

```
Your Kubernetes control-plane 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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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/

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

kubeadm join 172.31.11.237:6443 --token qafdgh.zj8seb6bky5m2zqa \
    --discovery-token-ca-cert-hash
sha256:93409d19a371a0ef1a1f6f4fa7bc95fcade298a28bc36a7fed83357c7c68c130
```

Add kube config file

On Master

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Install CNI

On Master

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

Optional (auto completion)

On Master

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

Add Nodes

On All Nodes, execute below command:

```
kubeadm join <IP Address>:6443 --token <token> \
    --discovery-token-ca-cert-hash <some hash> --node-name <Desired Nodename>
```

Verify the cluster setup:

```
kubectl get nodes
```

Output

```
root@ip-172-31-19-105:~# kubectl get nodes
NAME                STATUS   ROLES                   AGE     VERSION
ip-172-31-19-129    Ready    <none>                  3m21s   v1.20.5
ip-172-31-20-26     Ready    <none>                  3m28s   v1.20.5
control-plane       Ready    control-plane,master    10m     v1.20.5
```

Pods

Create Pod Imperatively

```
kubectl run myfirstpod --image=httpd --port 80
kubectl get pods -o wide
```

Create a Pod Declaratively

Create pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
 name: mypod
spec:
 containers:
 - name: mycontainer
   image: httpd
   ports:
   - containerPort: 80
```

Apply the yaml

```
kubectl apply -f pod.yaml
```

Generate YAML template

To generate YAML template from imperative command

```
kubectl run podname --image nginx --port 80 --dry-run -o yaml > pod.yaml
```