ADL EXP-6
COMMANDS

Steps to install Kubectl and execute Kubectl commands to manage the kubernetes cluster and deploy your First Kubernetes Application.

1. **Install Kubernetes tools**

Commands:
Both Master and Slave

1. **sudo apt-get install kubeadm kubelet kubectl –y**
2. **sudo apt-mark hold kubeadm kubelet kubectl**

2. **Verify the installation**

Both Master and Slave
Commands: kubeadm version

3. **Begin Kubernetes deployment**

Both Master and Slave
Commands: swapoff --a

4. **Assign Unique Hostname for Each Server Node Decide which server to set as the master node. Next, set a worker node hostname.**

On Master
Command: sudo hostnamectl set-hostname master-node
On Worker
Command: sudo hostnamectl set-hostname worker01

5. **Initialize Kubernetes on Master Node**

On Master
Command:

1. **sudo su**
2. **sudo kubeadm init --pod-network-cidr=10.244.0.0/16**

If you are trying to run this on EC2 you'll get an error message saying less cpu and memory to override the error run the above command with --ignore-preflight-errors=all
On Master
Command: sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

Once this command finishes, it will display a kubeadm join message at the end. Make a note of the whole entry. This will be used to join the worker nodes to the cluster.
Next, create a directory for the cluster.
On Master

Command:

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

6. **Deploy Pod Network to Cluster.**

On Master

Command: sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Allow the process to complete.

Verify that everything is running and communicating.

On Master

Command: kubectl get pods --all-namespaces

7. **Join Worker Node to Cluster. you can enter the kubeadm join command on each worker node to connect it to the cluster. Switch to the worker01 system.**

On Worker

Command:

1. **sudo su**
2. **kubeadm join 172.31.44.201:6443 --token iv4qnb.7aczfwllmfs4o7sc --discovery-token-ca-cert-hash sha256:33faf5ca90d079f20f6d1d48ca1bd225bc858462dbe0881418e7eeae4c4db075**

Switch to Master server.

The system should display the worker nodes that you joined to the cluster.

On Master.

Command: kubectl get nodes

8. **Running An Application on the Cluster. Deploy any containerized application to your cluster.**

On Master.

Command: kubectl create deployment nginx –image=nginx

9. **Create a service named nginx that will expose the app publicly.**

On Master.

Command: kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort

10. **Services are another type of Kubernetes object that expose cluster internal services to clients, both internal and external.**

On Master.

Command: kubectl get services

From the output you can retrieve the port that Nginx is running on.
To test that everything is working, visit http://worker_1_ip:nginx_port

11. Check the deployed container on the worker node.
On Worker.
Command: docker ps

12. To scale up the replicas for a deployment (nginx in our case)
On Master.
Command:

1. **kubectl scale --current-replicas=1 --replicas=2**
2. **kubectl get pods**
3. **kubectl describe deployment/nginx**

13. Remove the Nginx application. So first delete the nginx service from the master node. Then check that the service has been deleted.
On Master.
Command:

1. **kubectl delete service nginx**
2. **kubectl get services**

14. Delete the deployment. Then check if it has been deleted.
On Master.
Command:

1. **kubectl delete deployment nginx**
2. **kubectl get deployments**

15. Delete the node by Finding it, Draining it and then finally deleting it.
On Master.
Command:

1. **kubectl get nodes**
2. **kubectl drain nodetoberemoved**
3. **kubectl delete node nodetoberemoved**

16. Remove join/init setting from Worker node
On Worker.
Command:

1. **kubeadm reset**
2. **docker ps**

On Master.
Command: kubectl get nodes