

*System Admin*

*Training Assignments*

|  |  |
| --- | --- |
| **Program Code** |  |
| **Issue/Revision** | **x/y** |
| **Effective date** | **04/Aug /2023** |

**Assignment Day 15: Installing and Configuring Kubernetes**

**Part 1: Installation**

1. Choose a Linux-based system for the installation. You can use Ubuntu, CentOS, or any other distribution of your choice.
2. Install Docker: Kubernetes relies on Docker for containerization. Install Docker on your system. You can follow the official Docker installation documentation for your chosen Linux distribution.
3. Install **kubectl**: **kubectl** is the Kubernetes command-line tool used for interacting with the Kubernetes cluster. Install **kubectl** by following the instructions provided in the official Kubernetes documentation.
4. Install **kubeadm**, **kubelet**, and kubectl: These are essential components for setting up a Kubernetes cluster. You can install them by following the instructions for your specific Linux distribution on the Kubernetes website.

**Part 2: Cluster Initialization**

1. Initialize the Kubernetes cluster: Use kubeadm to initialize the cluster on your master node. This command will generate a unique token to be used by worker nodes to join the cluster.
2. Join worker nodes: On each worker node, use the token generated in the previous step to join the cluster. This will establish communication between the master and worker nodes.

**Part 3: Configuration**

1. Configure kubectl: Configure kubectl to connect to your Kubernetes cluster. You'll need to provide the IP address or hostname of the master node. Test the configuration by running a simple kubectl command to retrieve cluster information.
2. Deploy a sample application: Using kubectl, deploy a sample application (e.g., Nginx web server) on your Kubernetes cluster. Ensure that the application runs successfully.

**Part 4: Verification**

1. Verify the cluster status: Use kubectl to check the status of your Kubernetes nodes and pods. Ensure that all nodes are in the "Ready" state, and the application pods are running.

**Part 5: Cleanup (Optional)**

1. If desired, you can remove the cluster and clean up the nodes using kubeadm.

**Solution**

**Part 1: Installation**

* Follow the official Docker installation guide for your Linux distribution. Or use the old Docker VM
* Install ‘kubectl’:
* *sudo apt-get update && sudo apt-get install -y apt-transport-https*
* *curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -*
* *echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list*
* *sudo apt-get update*
* *sudo apt-get install -y kubectl*
* Install **kubeadm**, **kubelet**, and **kubectl**:
* *sudo apt-get update && sudo apt-get install -y kubeadm kubelet kubectl*

Note: Do this step on both Master node and Worker node

**Part 2: Cluster Initialization**

* Initialize the Kubernetes cluster on the master node:
* *sudo kubeadm init*
* On each worker node, join the cluster using the token generated during initialization:
  + *sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash sha256:<hash>*

**Part 3: Configuration**

* Configure kubectl on your local machine:
* *mkdir -p $HOME/.kube*
* *sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config*
* *sudo chown $(id -u):$(id -g) $HOME/.kube/config*
* Deploy a sample Nginx application:
* *kubectl create deployment nginx --image=nginx*

**Part 4: Verification**

* Verify the cluster status:
* *kubectl get nodes*
* *kubectl get pods*

**Part 5: Cleanup (Optional)**

* To clean up and remove the cluster, use the following commands on the master node:
* *kubectl drain <node-name> --delete-local-data --force --ignore-daemonsets kubectl delete node <node-name>*
* *sudo kubeadm reset*