# KUBERNETES TASKS

DEVOPS MASTER ARCHITECT WORKSHOP

# TASK 1:HOW TO CREATE KUBERNETES CLUSTER – MINIKUBE WAY

- Activity: Install Kubernetes cluster in minikube

- Definition of Done:
  - Minikube installed
  - Create a GitHub repository for Kubernetes task. Create a subfolder and upload the relevant screenshot.(Terminal screenshot of version of the Minikube Kubernetes installed)

| # | Steps | Commands |
|---|---|---|
| 1 | Clone the Kubernetes repository | git clone https://github.com/AnjuMeleth/DevOpsMasterKubernetesTasks.git |
| 2 | Run the installation script in 1_installation sub folder | sh minikube_install.sh |
| 3 | Get the kubectl version | sudo kubectl version --client=true |
| 4 | Get the minikube version | sudo minikube version |

# TASK 2:HOW TO START A MINIKUBE CLUSTER

- Activity: Start a minikube cluster on Ubuntu 18.04 system
- Definition of Done:
  - Create a subfolder and upload the relevant screenshot.(Minikube started)

| # | Steps | Commands |
|---|---|---|
| 1 | Start a Minikube cluster | sudo minikube start |
| 2 | Throws an error “Unable to start VM as this computer doesn't have VT-X/AMD-v enabled” | |
| 3 | Start Minikube with no driver | sudo minikube start --vm-driver=none |
| 4 | Throws an error “"docker": executable file not found in $PATH”. Install docker | sudo apt-get install docker.io |
| 5 | Start the minikube once again | sudo minikube start --vm-driver=none |

# TASK 3:HOW TO DEPLOY AN APPLICATION ON MINIKUBE CLUSTER USING PODS

- Activity: Deploy a Nginx application on to Minikube cluster as a pod
- Definition of Done:
  - Pod created .
  - Create a subfolder and upload the relevant screenshot.(List of pods running)

| # | Steps | Commands |
|---|---|---|
| 1 | Run a pod on the minikube cluster | sudo kubectl run nginx --image=nginx --port=80 |
| 2 | View the pods | sudo kubectl get pods |

# TASK 4:HOW TO ACCESS AN APPLICATION RUNNING IN A POD FROM OUTSIDE

- Activity: Create a NodePort service that helps to connect to pod created in the last task.

- Definition of Done:
  - Web server accessible at a port
  - Create a subfolder and upload the relevant screenshot.( Nginx service accessible at the NodePort)

| # | Steps | Commands |
|---|---|---|
| 1 | Create a NodePort service | sudo kubectl expose pod nginx --type=NodePort |
| 2 | List the services | sudo kubectl get services |
| 3 | Access the server from the browser | http://<Ip address of the system>:port |
| 4 | Delete the service | sudo kubectl delete svc nginx |
| 5 | Stop the Minikube cluster | sudo minikube stop |
| 6 | Delete the Minikube cluster | sudo minikube delete |
| 7 | Delete the config file | sudo rm -rf ~/.kube/config |
| 8 | Delete the kubernetes | rm -rf /etc/kubernetes |

# TASK 5:HOW TO ESTABLISH PASSWORD LESS CONNECTIVITY ACROSS INSTANCES GOING TO BE USED AS MANAGER AND WORKER NODES FOR COMMUNICATION

- Activity: Establish passwordless connectivity by using ssh agent

- Definition of Done:
  - Able to connect to worker nodes from the manager without password
  - No screenshot to be uploaded

| # | Steps | Commands |
|---|---|---|
| 1 | Transfer the .pem file to Kubernetes manager node using Filezilla | |
| 2 | Stop all firewall | sudo service ufw stop |
| 3 | Start ssh | sudo service ssh start |
| 4 | Add the ssh details | eval `ssh-agent -s`<br>sudo chmod 400 JenkinsMumbai.pem<br>ssh-add JenkinsMumbai.pem |
| 5 | Access the node system from master | ssh ubuntu@<IP address> |
| 6 | Stop all firewall | sudo service ufw stop |

# TASK 6 :HOW TO BRING UP A KUBERNETES MASTER IN A PRODUCTION CLUSTER USING KUBEADM

- Activity: Create a Kubernetes master using Kubeadm installation

- Definition of Done:
  - Kubernetes master up and running.
  - Create a subfolder and upload the relevant screenshot.(No screenshots needed)

| # | Steps | Commands |
|---|---|---|
| 1 | Modify the hostname in Kubernetes manager node | sudo vi /etc/hostname |
| 2 | Enter the Kubernetes master and node details in /etc/hosts file of the master | sudo vi /etc/hosts |
| 3 | Run the installation script | sh kubeadm_install.sh |
| 4 | Edit the file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf with the content | sudo nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf<br><br>Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs" |
| 5 | Initialise the Kubernetes cluster by advertising the master IP address. Make a note of the output details | sudo kubeadm init --pod-network-cidr=10.244.0.0/16 |

# Task 6 :how to bring up a Kubernetes master in a production cluster using kubeadm Contd…

| 6 | **Run the noted commands** | **mkdir -p $HOME/.kube<br>sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config<br>sudo chown $(id -u):$(id -<br>$HOME/.kube/config** |
|---|---|---|
| 7 | List the pods | kubectl get pods -o wide --all-namespaces |
| 8 | Deploy a pod network | kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/2140ac876ef134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml |
| 9 | List the pods | kubectl get pods -o wide --all-namespaces |

# TASK 7:HOW TO ADD A WORKER NODE TO KUBERNETES CLUSTER

- Activity: Add a worker node Ubuntu 18.04 to Kubernetes cluster

- Definition of Done:
  - Node added to cluster
  - Create a subfolder and upload the relevant screenshot.(List of nodes added to the cluster)

| # | Steps | Commands |
|---|---|---|
| 1 | Make sure you have the right hostname details in /etc/hosts file and /etc/hostname files | sudo vi /etc/hostname<br>sudo vi /etc/hosts |
| 2 | Run the installation script | sh kubeadm_install.sh |
| 3 | Edit the file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf with the content | sudo nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf<br><br>Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs" |
| 4 | Run the join token command received during the kubeadm init command | sudo kubeadm join <>--token <token> |
| 5 | Get the list of nodes in the Kubernetes master | kubectl get nodes |

# TASK 8: HOW TO DEPLOY AN APPLICATION POD ON KUBERNETES CLUSTER AND ACCESS IT

- Activity: Apply a pod of nginx application to the Kubernetes cluster and access it by a NodePort service

- Definition of Done:
  - Nginx up , running and accessible at a NodePort
  - Create a subfolder and upload the relevant screenshot.(Nginx web page on the worker node )

| # | Steps | Commands |
|---|---|---|
| 1 | Run a pod on the cluster | sudo kubectl run nginx --image=nginx --port=80 |
| 2 | View the pods | sudo kubectl get pods |
| 3 | Create a NodePort service | sudo kubectl expose pod nginx --type=NodePort |
| 4 | List the services | sudo kubectl get svc |
| 5 | Access the server from the browser | http://< Correct Ip address of the system>:port |
| 6 | Get more details about the pods | kubectl describe pods nginx |
| 7 | Delete the service | sudo kubectl delete svc nginx |
| 8 | Delete pods | kubectl delete pods nginx |

# TASK 9 : HOW TO DEPLOY A POD ON KUBERNETES CLUSTER AND ACCESS IT USING KUBECTL

- Activity: Deploy apache application on the Kubernetes cluster

- Definition of Done:
  - Apache up , running and accessible at NodePort.
  - Create a subfolder and upload the relevant screenshot.(Apache web page on the worker node)

| # | Steps | Commands |
|---|---|---|
| 1 | Run a pod on the cluster | sudo kubectl run apache --image=httpd --port=80 |
| 2 | View the pods | sudo kubectl get pods |
| 3 | Create a NodePort service | sudo kubectl expose pod apache --type=NodePort |
| 4 | List the services | sudo kubectl get svc |
| 5 | Access the server from the browser | http://< Correct Ip address of the system>:port |
| 6 | Get more details about the pods | kubectl describe pods apache |

# TASK 10 : HOW TO DEPLOY A POD ON KUBERNETES CLUSTER USING YML

- Activity: Deploy nginx on the Kubernetes cluster using yml

- Definition of Done:
  - Nginx up , running and accessible at NodePort.
  - Create a subfolder and upload the relevant screenshot.(No screenshot to be uploaded)

| # | Steps | Commands |
|---|---|---|
| 1 | Create ngin_pod.yml | vi nginx_pod.yml |
| 2 | Apply the yml | kubectl apply -f nginx_pod.yml |
| 3 | Get the list of pods running | kubectl get pods |
| 4 | Delete the pods | kubectl delete pods mypod |

# TASK 11:HOW TO CREATE A REPLICATION CONTROLLER

- Activity: Create a replication controller for Nginx application so that it maintains the number of instances of the same pod running

- Definition of Done:
  - Replication controller created.
  - Create a subfolder and upload the relevant screenshot.(Terminal screenshot of list of replication controllers created)

| # | Steps | Commands |
|---|---|---|
| 1 | Edit the nginx_rc.yml file in 11_rc folder | vi nginx_rc.yml |
| 2 | Apply the yml file | kubectl apply -f nginx_rc.yml |
| 3 | List the replication controller | kubectl get rc |
| 4 | Get the list of pods | kubectl get pods |
| 5 | Get the details about the replication controller created | kubectl describe replicationcontrollers/nginx |
| 6 | To list only the pods under the replication controller | pods=$(kubectl get pods --selector=app=nginx --output=jsonpath={.items..metadata.name})<br><br>echo $pods |

# TASK 12:HOW TO SCALE AN APPLICATION IN A KUBERNETES CLUSTER

- Activity: Scale the Nginx application from the previous example to run 5 instances to cater to a greater demand

| # | Steps | Commands |
|---|---|---|
| 1 | Scale the application to run 5 instances | kubectl scale replicationcontrollers/nginx --replicas=5 |
| 2 | List the rc | kubectl get rc |
| 3 | List the pods | kubectl get pods |

- Definition of Done:
  - Create a subfolder and upload the relevant screenshot.(List of 5 pods running)

# TASK 13 : HOW TO ACCESS AN APPLICATION FROM OUTSIDE

- Activity: Create a service for the Nginx application running in the replication controller in previous tasks

| # | Steps | Commands |
|---|---|---|
| 1 | Edit the nginx_service.yml in 13_service folder | vi nginx_service.yml |
| 2 | Apply the yml | kubectl apply -f nginx_service.yml |
| 3 | Get the list of services | kubectl get svc |
| 4 | Access the Nginx application in port 8080 | http://<Ip address>:port |

- Definition of Done:
  - Create a subfolder and upload the relevant screenshot.()

# TASK 14 : HOW DOES SELF HEALING WORK IN REPLICATION CONTROLLER

- Activity: Delete all pods in the replication controller in previous example

- Definition of Done:
  - New pods gets created. No screenshots to be uploaded.

| # | Steps | Commands |
|---|---|---|
| 1 | Get the list of rc | kubectl get rc |
| 2 | Get the list of services | kubectl get svc |
| 3 | Get the list of the pods running | kubectl get pods |
| 4 | Delete all pods | kubectl delete po --all |
| 5 | Get the list of the pods running | kubectl get pods |

# TASK 15 : HOW DOES THE CLIENT PODS GET TO KNOW THE IP ADDRESS AND PORT OF SERVICES(SERVICE DISCOVERY)

- Activity: All the services in the Kubernetes cluster has an Ip assigned and this could be seen in the pods created . If the service is created before the pods recreate the pods once again.

- Definition of Done:
  - Environment variables could be seen in the pod.
  - Create a subfolder and upload the relevant screenshot.(Terminal screenshot of environment variables in a pod)

| # | Steps | Commands |
|---|---|---|
| 1 | Get the list of rc | kubectl get rc |
| 2 | Get the list of services | kubectl get svc |
| 3 | Get the list of the pods running | kubectl get pods |
| 4 | View the environment variables in any one pod | kubectl exec <pod name> env |

# TASK 16: HOW TO CREATE A DEPLOYMENT IN KUBERNETES CLUSTER AND ACCESS IT FROM OUTSIDE

- Activity: Create a deployment that has 5 instances of nginx application running. Then access the nginx service from a NodePort

- Definition of Done:
  - Deployment and service created.
  - Create a subfolder and upload the relevant screenshot.(Terminal screenshot of the list of deployments in the Kubernetes cluster)

| # | Steps | Commands |
|---|---|---|
| 1 | Edit the nginx_deployment.yml file in 16_deployment folder | vi nginx_deployment.yml |
| 2 | Apply the yml file | kubectl apply -f nginx_deployment.yml |
| 3 | List the replication controller | kubectl get deployment |
| 4 | Get the list of pods | kubectl get pods |
| 5 | Get more details about the deployment | kubectl describe deployment/nginx-deployment |
| 6 | To list only the pods under the replication controller | pods=$(kubectl get pods --selector=app=nginx1 --output=jsonpath={.items..metadata.name})<br><br>echo $pods |
| 7 | Create a service | vi nginx_service.yml |

# TASK 16 : CONTD…

| 8 | **Apply the service yml file** | **kubectl apply -f nginx_service.yml** |
|---|---|---|
| 9 | List the services | kubectl get svc |
| 10 | Access the Nginx application | http://<IpAddress>:port |

# TASK 17 : HOW TO DO ROLLING UPDATE IN KUBERNETES

- Activity: Create a deployment that has 3 instance of a Node js(version1) application running. We need to update it with a newer version version 2

- Definition of Done:
  - Pods run with newer version
  - Create a subfolder and upload the relevant screenshot.( Terminal screenshot showing the version of the image used in pods of replication controller)

| # | Steps | Commands |
|---|---|---|
| 1 | Edit the file kubia.yml in 17_rolling update folder | vi kubia.yml |
| 2 | Apply | kubectl apply -f kubia.yml |
| 3 | Get the pods | kubectl get pods -l app=kubia |
| 4 | Modify the image of the pods | kubectl set image deployment kubia nodejs=luksa/kubia:v2 |
| 5 | Get the list of pods | kubectl get pods -l app=kubia |
| 6 | Get the details of the deployment | kubectl describe deployment kubia |

# TASK 18 :HOW TO DO A ROLL BACK IN KUBERNETES

- Activity: Update the deployment created in the previous task with an error image. Then perform a roll back

- Definition of Done:
  - All the pods uses the previous version of the image.
  - Create a subfolder and upload the relevant screenshot.(Terminal screenshot showing the rollback status)

| # | Steps | Commands |
|---|---|---|
| 1 | Update the image used in the deployment kubia to a new version | kubectl set image deployment kubia nodejs=luksa/kubia:v3 |
| 2 | Access the server | http://<Ipaddress>:port |
| 3 | Refresh the webserver for few times so that we get error message | |
| 4 | Roll back to previous version | kubectl rollout undo deployment kubia |

# TASK 19 : HOW TO DO CANARY DEPLOYMENTS IN KUBERNETES

- Activity: Update the deployment created with a new version. Pause the update . Resume it

- Definition of Done:
  - Canary deployment is performed
  - Create a subfolder and upload the relevant screenshot.( Terminal screenshot showing the new pod with new image )

| # | Steps | Commands |
|---|---|---|
| 1 | Update the image of the deployment with Version 4 | kubectl set image deployment kubia nodejs=luksa/kubia:v4 |
| 2 | Pause the deployment | kubectl rollout pause deployment kubia |
| 3 | Resume the deployment | kubectl rollout resume deployment kubia |

| # | Steps | Commands |
|---|---|---|
| 1 | Goto the folder 19_canary and apply the yml configuration to the cluster | kubectl apply -f newkubia.yml |
| 2 | Get the services from the nodeport address | http://<ip address>:Nodeport |

# TASK 20 :HOW TO INSTALL KUBERNETES DASHBOARD

- Activity: Install Kubernetes GUI
- Definition of Done:
  - Kubernetes dashboard is visible
  - Create a subfolder and upload the relevant screenshot.(screenshot of Kubernetes dashboard)

| # | Steps | Commands |
|---|---|---|
| 1 | Apply dashboard yml available | kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta4/aio/deploy/recommended.yaml |
| 2 | Edit the services and put the type as NodePort | kubectl edit services kubernetes-dashboard -n kubernetes-dashboard |
| 3 | Run kubectl proxy and open another terminal | kubectl proxy & |
| 4 | Get the services list and note down the NodePort | kubectl get svc --all-namespaces |
| 5 | Open any browser other than chrome. Go to Advanced and accept risk and continue | https://<ipaddress>:port |
| 6 | Create a service account | kubectl create serviceaccount dashboard -n default |
| 7 | Create cluster role binding | kubectl create clusterrolebinding dashboard-admin -n default --clusterrole=cluster-admin --serviceaccount=default:dashboard |

# TASK 20:CONTD…

| # | Steps | Commands |
|---|---|---|
| 8 | Get the secret token | kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" \| base64 --decode |
| 9 | Copy and paste the token in the dashboard | |

# TASK 21 :HOW TO DEPLOY SPRING PET CLINIC APPLICATION

- Activity: Deploy the spring pet clinic application on to the Kubernetes cluster

- Definition of Done:
  - Spring pet clinic application is accessibl at the NodePort
  - Create a subfolder and upload the relevant screenshot.(screenshot of spring pet clinic application)

| # | Steps | Commands |
|---|---|---|
| 1 | Apply the yml in 21_springpetclinic folder | kubectl apply -f petclinic.yml |
| 2 | Get the Nodeport port from the service | kubectl get svc |
| 2 | Access the application | http://<ipaddress>:port |