|  |  |
| --- | --- |
| *A close up of a logo  Description automatically generated* | *DEPARTMENT OF INFORMATION TECHNOLOGY* |

|  |  |
| --- | --- |
| Semester | T.E. Semester V – Information Technology |
| Subject | Advance DevOps Lab |
| Subject Professor In-charge | Prof. Indu Anoop |
| Laboratory | (Leave blank for now) |

|  |  |  |
| --- | --- | --- |
| Student Name | Aryaman More | |
| Roll Number | 20101A0072 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment | 4 | |
| Problem Statement | To install kubectl and execute kubectl commands to manage the Kubernetes cluster and deploy your first Kubernetes Application. | |
| Resources / Apparatus Required | Hardware: Computer System (Internet Connectivity) | Software: Web Browser |
| Details | **Theory:** Kubernetes led by google is an open-source platform for managing container technologies such as Docker.  Docker lets you create containers for a pre-configured image and application. *Kubernetes [ Greek for “Pilot”] provides the next step, allowing you to balance loads between containers and run multiple containers across multiple systems.*    **Container:** Provides an isolated context in which an app together with it’s environment (supporting structure eg: web server) can run.  **Pods:** Represents a runnable unit usually consisting of a single container. [May contain more containers if containers are tightly coupled] Kubernetes connects the pod to the n/w and rest of the Kubernetes eco-system. | |
| Code | Prerequisite:  2 AWS instance (virtual servers-ubuntu 20) one acting as Master Node and Other as Worker Node. Docker and Kubernetes installation done on both nodes.  <https://mobaxterm.mobatek.net/download.html>  Now that your cluster is verified successfully, let’s schedule an example Nginx application on the cluster.  **SECTION D: Running An Application on the Cluster**  You can now deploy any containerized application to your cluster. To keep things familiar, let’s deploy Nginx using Deployments and Services to see how this application can be deployed to the cluster. You can use the commands below for other containerized applications as well, provided  you change the Docker image name and any relevant flags (such as ports and volumes).  **Step 1: Create deployment named nginx [on master]**  Still within the master node, execute the following command to create a deployment named nginx:  kubectl create deployment nginx --image=nginx  A deployment is a type of Kubernetes object that ensures there’s always a specified number of pods running based on a defined template, even if the pod crashes during the cluster’s lifetime.  The above deployment will create a pod with one container from the Docker registry’s Nginx Docker Image.  Next, run the following command to create a service named nginx that will expose the app publicly. It will do so through a NodePort, a scheme that will make the pod accessible through an arbitrary port opened on each node of the cluster:  kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort  Services are another type of Kubernetes object that expose cluster internal services to clients, both internal and external. They are also capable of load balancing requests to multiple pods, and are an integral component in Kubernetes, frequently interacting with other components.  Run the following command:  kubectl get services  This will output text like the following:  Output  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 1d  nginx NodePort 10.109.228.209 <none> 80:**nginx\_port**/TCP 40m  From the third line of the above output, you can retrieve the port that Nginx is running on. Kubernetes will assign a random port that is **greater than 30000** automatically, while ensuring that the port is not already bound by another service.    **Note: if you're running your setup on ec2 ensure the nginx\_port is open under the inbound rules in the security groups.**  To test that everything is working, visit  http://worker\_1\_ip:nginx\_port  or  http://worker\_2\_ip:nginx\_port  through a browser on your local machine. You will see Nginx’s familiar welcome page.  To see the deployed container on worker node switch to worker01  docker ps  Output: you will see the container for nginx image running.  **SECTION E: Scale up replicas for a deployment**  If you want to scale up the replicas for a deployment (nginx in our case) the use the following command:  kubectl scale --current-replicas=1 --replicas=2 deployment/nginx  kubectl get pods  Output: you will see 2/2 as output in nginx deployment.  kubectl describe deployment/nginx  Output: give details about the service deployed  If you would like to remove the Nginx application, first delete the nginx service from the master node:  kubectl delete service nginx  Run the following to ensure that the service has been deleted:  kubectl get services  You will see the following output:  Output  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 1d  Then delete the deployment:  kubectl delete deployment nginx  Run the following to confirm that this worked:  kubectl get deployments  Output  No resources found. | |
| Output | On master node:    Access of worker node ip via browser to see successfully deployed application:    Replication of Pods:    Deletion of application: | |
| Conclusion | Executed kubectl commands to manage the Kubernetes cluster and deploy a nginx Application. | |