Final Project

1. Create Instances

A. Create 3 Ubuntu (1Master, 2Worker) instances with the following resources: 2CPU, 8GB Ram, Disk Size 20GB (t2.large). **0.5Point - Screen**

B. Create 1 Ubuntu management instance with the following resources: 1CPU, 1RAM, Disk Size 20GB (t2.small) **0.5Point - Screen**

C. Add Inbound rules for All traffic 0.0.0.0/0 subnet. **0.5Point - Screen**

D. Create Public key on management machine and then insert this key on Master and worker instances **0.5Point - Screen**

2. Install Docker On Master, Worker and management Instances And Add Docker Group And user - 1Point - Screen

3. Install Kubectl on Management Instance: - 0.5Point - Screen

4. **Install Helm on Management Instance, we need for install rancher API: - 0.5Point - Screen**

5. Install And Configure RKE on Management Instance. 0.5Point - Screen

6. Create Cluster Config with 1 master and 2 worker node - **1.5Point - Screen**

7. Export KubeConfig on Management Cluster - 0.5Point - Screen

8. **Generate Certs and install rancher api - 2Point - Screen**

9. Install and Configure Grafana Prometheus From Rancher UI: - 1Point - Screen UI

10. **A project that you are working on has a requirement for persistent data to be available. 4Point**

To facilitate this, perform the following tasks:

1. Create a file on node sk8s-node-0 at /opt/KDSP00101/data/index.html with the content Acct=Finance **0.5Point**
2. Create a PersistentVolume named task-pv-volume using hostPath and allocate 1Gi to it, specifying that the volume is at /opt/KDSP00101/data on the cluster's node. The configuration should specify the access mode of ReadWriteOnce . It should define the StorageClass name exam for the PersistentVolume , which will be used to bind PersistentVolumeClaim requests to this PersistenetVolume. **1Poin**
3. Create a PefsissentVolumeClaim named task-pv-claim that requests a volume of at least 100Mi and specifies an access mode of ReadWriteOnce **1Point**
4. Create a pod that uses the PersistentVolmeClaim as a volume with a label app: my-storage-app mounting the resulting volume to a mountPath /usr/share/nginx/html inside the pod **1.5Point**

Screen: cat /opt/KDSP00101/data/index.html

Screen: k get pv,pvc

Screen: k get pods

11. **Requests and Limits, ServiceAccount 1,5Point**

1. Team Neptune needs 3 Pods of image httpd:2.4-alpine, create a Deployment named neptune-10ab for this. The containers should be named neptune-pod-10ab. Each container should have a memory request of 20Mi

Screen: k get pods -n neptune

Screen: k get sa -n neptune

12. Task: - 2Point

1. Create a secret named app-secret in the default namespace containing the following single key-value pair: Key3: value1 - **0.5Point**
2. Create a Pod named ngnix secret in the default namespace.Specify a single container using the nginx:stable image. **1Point**
3. Add an environment variable named BEST_VARIABLE consuming the value of the secret key3. **0.5Point**

Screen: k get secret

Screen: k get pods

Screen: k get pod nginx -o yaml

13. Task: - 1.5Point

1. Create a Pod named nginx-resources in the existing pod default namespace, Specify a single container using nginx:stable image.
2. Specify a resource request of 300m cpus and 1G1 of memory for the Pod's container.

Screen: k get pods

Screen: k get pods nginx-resources -o yaml

14. Create a new deployment for running.nginx with the following parameters And Publish this with ingress: 6Point

1. Create btu-final namespace **0.5Point**
2. Run the deployment in the btu-final namespace. namespace has already been created, Name the deployment nginx-deployment and configure with 2 replicas **1Point**
3. Configure the pod with a container image of nginx:1.14.2 containerPort 80 **1Point**
4. Set an environment variable of NGINX__PORT=8080 and also expose that port for the container above **0.5Point**
5. Create Service nginx-deployment **0.5Point**
   a. kubectl expose deploy nginx-deployment --type=ClusterIP -n btu-final
6. Create Ingress with the following parameters **1.5Point**
   a. Name of ingress: nginx-deployment
   b. Namespace: btu-final
   c. Host: nginx.final.eu
   d. Backend service nginx-deployment port 80

Screen: k get namespace

Screen: k get deployment -n btu-final

Screen: k get deployment nginx-deployment -o yaml -n btu-final

15. You are tasked to create a ConfigMap in btu-final and consume the ConfigMap in a pod using a volume mount. 2Point

1. Create a ConfigMap named another-config containing the key/value pair: key4/value3 - **1Point**
2. start a pod named nginx-configmap containing a single container using the nginx image, and mount the key you just created into the pod under directory /also/a/path **1Point**

Screen: k get cm -n btu-final

Screen: k get pods nginx-configmap -n btu-final -oyaml

16. Create a new file /root/Dockerfile to build a container image from. It should: - 1Point

1. use bash as base image and run ping killercoda.com **-0.5Point**
   a. Build the image and tag it as pinger and tag 3.0 .
2. Using the tool of your choice export the built container image in OC-format and store it at /root/pinger3.0.tar **-0.5Point**

Screen: docker images

Screen: ls /root/

17. You are required to create a pod in the btu-final namespace that requests a certain amount of CPU and memory, so it gets scheduled to-a node that has those resources available. 1Point

1. Create a pod named nginx-resources in the btu-final namespace that requests a minimum of 200m CPU and 1Gi memory for its container
2. The pod should use the nginx image

Screen: k get pods -n btu-final

Screen: k get pods nginx-resources -o yaml -n btu-final

18. Team Neptune needs 3 Pods of image httpd:2.4-alpine, create a Deployment named neptune-10ab for this. The containers should be named neptune-pod-10ab. Each container should have a memory request of 20Mi and a memory limit of 50Mi, Team Neptune has its own ServiceAccount neptune-sa-v2 under which the Pods should run. The Deployment should be in Namespace btu-final. 2Point

Screen: k get pods -n btu-final

Screen: k get sa -n btu-final

Final Score: 30