Assignment 3

Deadline: 24/11/2025

Task 1 – Create a Cluster

1. Create private repository named as devops in github if not already created in your account. Create folder assignment3 and inside that folder create 6 folders task1, task2, task4, task5 and task6 for each task in this assignment. Push all code in those folders when completed.

2. Add collaborator muhammadimranfarooqi in your repository if not done in previous assignment.

3. Add README.md in markdown format in each task folder to explain tasks in detail.

4. Write report also in pdf/word file with screenshots and explanation. Also add your github repository link in first page.

Task 1:

1 Install Minikube to create a single-node cluster on your local machine. Choose the option that uses Docker.

2 Deploy a single-container app: The goal of this task is to create a pod that consists of one container. Use an image of your preference and create a deployment via Kubernetes. Explain the commands you used to view your deployment and how you are going to check your cluster's runtime.

3 Configure Kubernetes Dashboard (GUI) (https://minikube.sigs.k8s.io/docs/handbook/dashboard/)

4 Apart from the kubectl, use the Kubernetes Dashboard too for deployment and provide YAML templates and screenshot.

Attention! For the next exercises make sure that a Minikube cluster is up and running.

Task 2 -Converting the Docker-compose flow to Kubernetes

In Docker lab, with docker compose, you were able to build multi-container apps. Unfortunately, you cannot use the same configuration file format to run multi-container pods into Kubernetes. However, there are tools that allow us to convert a compose file into a Kubernetes.

1. Install Kompose (https://kompose.io/getting-started). This tool will allow us to convert the

compose file to Kubernetes.
2. Use the source code from the first Docker tutorial (https://docs.docker.com/compose/gettingstarted) to make some adjustments to the compose file. You should make the following changes:

(a) For the web service, you cannot use any more build command that points to the local Dockerfile. You should publish your image to your repository as username-toregistry/image-name. Then change the build declaration to image and insert the name of the registry you just created.

(b) Add to the web service an "always" restart policy.

(c) You should define the ports for the **redis** service.

3. Convert the compose file to Kubernetes, and create a deployment to your cluster.

(HINT; Follow the documentation from Kompose.)

4. Find the services using the appropriate **kubectl** command, and get its output with the command: minikube service <service-name> This will automatically open your browser to the node's proxy address.
HINT; If everything was implemented right, you will see the same output as the one you saw when you used the compose.

Task 3 – Scale up your App

1. Delete the previous Minikube container and start a new Minikube cluster with two nodes, using the following command: **minikube start --nodes 2**. You can view the available nodes using the following command: **kubectl get nodes** Wait until all nodes are ready. It may take some time...
2. Inside folder qu3, there is a Kubernetes configuration file, called "hello–deployment.yaml". Describe the deployment and the following fields:

a. kind

b. replicas

c. replicas.strategy

d. spec.template.spec.affinity

e. spec.template.spec.containers

3. Make a new deployment to Kubernetes using the aforementioned configuration file. Use the command: **kubectl get pods -o wide**. You will see that the two pods are running on two different nodes. We want to change that and assign the two pods to one specific node. To do so, perform the following steps:
a. Delete the previous deployment and make sure no other pod is running.

b. You should associate the preferred node with a label. You can do so, by using the command: **kubectl label nodes <your-node-name> disktype=<new-label>**

c. View the new label using the command: **kubectl get nodes –-show-labels**

d. Create a new configuration file with the name "hello–deployment_updated.yaml". Copy inside it the code from "hello–deployment.yaml", and make adjustments to associate these two pods with the node that has been received the <new-label>.
HINT: There is a configuration inside the hello–deployment.yaml that ensures the pods will land on separate hosts. Find this configuration and delete it.

Task 4 – Create Phpmyadmin and mysql apps

1. Create deployment and expose them using ClusterIPs.
2. Create a **Persistent Volume (PV)** and a **Persistent Volume Claim (PVC)** using YAML.
3. Mount the PVC to the mysql deployment.
4. Verify that apps work

Task 5 – Managing Configurations and Secrets

1. Create a **ConfigMap** for storing application specific variables.
2. Create a **Secret** containing sensitive data
3. Modify a deployment to use:
    - ConfigMap values as environment variables.
    - Secrets securely using valueFrom references.
4. Verify the environment variables inside the running pod:

Task 4 – Expose Services via NGINX Ingress Controller (New Task)

Configure nginx ingress controller in minikube. Create and expose apps of phpmyadmin and mysql using Cluster IPs. Then add ingress to expose service via nginx.