

Kubernetes Project Assignment: FlaskApp with Database

Objective:

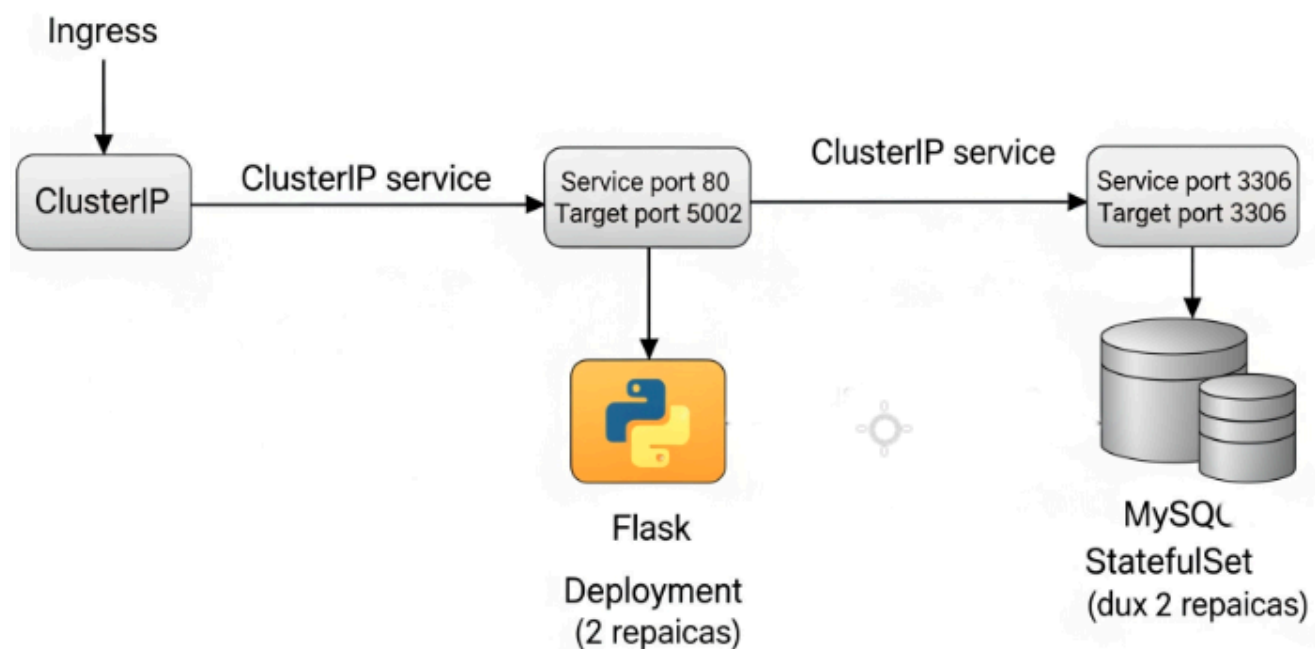
Deploy a web application (FlaskApp) with a database backend on a Kubernetes cluster using core Kubernetes resources.

Prerequisites:

You are provided with two project directories:

- `flaskapp/` – Python Flask application with Dockerfile
 - `database/` – Database service (MySQL) with Dockerfile
-

Simple Architecture :-



Tasks:

1. Build & Push Docker Images

- ✓ Build Docker images for both `flaskapp` and `database` using the provided Dockerfiles.
 - ✓ Push the images to your **DockerHub repository**. "use at least one private repository to use secrets in kubernetes later "
-

2. Kubernetes Cluster Setup

- Set up a Kubernetes cluster using **kubeadm**, **k3s** , "minikube not allowed"

- ✓ Minimum 2 nodes:
 - 1 Master Node
 - 1 Worker Node
-

3. Application Deployment

Use the following Kubernetes resources to deploy your applications:

- ✓ **Deployment** for the Flask application.
 - ✓ **StatefulSet** for the Database to ensure stable persistent identity and volume.
 - ✓ **ConfigMap** to store environment configuration for the FlaskApp.
 - ✓ **ConfigMap** to store environment configuration for the MySQL.
 - ✓ **Secret** to store sensitive data like Docker Registry Credentials .
 - ✓ **PersistentVolume** (hostPath for testing).
 - ✓ **ClusterIP Service** to expose FlaskApp and database internally.
 - ✓ **Ingress** to route HTTP traffic to FlaskApp (use ingress-nginx or similar controller).
 - ✓ **Network policy**
 - ✓ **Readiness , startup , liveness**
 - ✓ **Limits** (you can use **Resource Limit Range** or **Resource Quota**)
 - -----
 - **StorageClass** and **Dynamic Provisioning** (NFS). -- Bonus
 - ✓ **PersistentVolumeClaim** -- Bouns
 - ✓ **NodePort Service** to test external access to FlaskApp. -- Bonus
-

Flask Application to MySQL Database Connection :-

The Flask application connects to a MySQL database using environment variables to configure credentials, database name, and connection host. This setup ensures modularity and ease of deployment in Kubernetes environments.

The following environment variables must be set in the Flask app **Deployment** to allow successful communication with the MySQL **StatefulSet**:

```
MYSQL_DATABASE_USER: "root"
MYSQL_DATABASE_PASSWORD: "root"
MYSQL_DATABASE_DB: "BucketList"
MYSQL_DATABASE_HOST: "db-service"
```

Explanation:

Variable	Description
<code>MYSQL_DATABASE_USER</code>	Username used by the Flask app to access MySQL
<code>MYSQL_DATABASE_PASSWORD</code>	Password for the MySQL user
<code>MYSQL_DATABASE_DB</code>	Name of the database to connect to
<code>MYSQL_DATABASE_HOST</code>	Hostname of the MySQL service (<code>db-service</code>)

2- MySQL Root Password (Important for StatefulSet)

In your MySQL **StatefulSet**, you must configure the `MYSQL_ROOT_PASSWORD` to enable initialization and root-level access:

Bonus: Dynamic Provisioning with NFS :-

- Install and configure an **NFS server** on a node (master node or another new node dedicated for NFS).
- Create a **StorageClass** that points to the NFS server.
- Update PVCs to use the dynamic provisioning from the NFS-backed StorageClass.

Validation & Testing :-

✓ Confirm:

- FlaskApp connects to DB and shows output.
- NodePort works: access via `curl` or browser using `NodeIP:NodePort` and `Cluster IP service`

Python Flask App

Home

Sign In

Sign Up

Bucket List App

Sign up today

Bucket List

Donec id elit non mi porta gravida at eget metus. Maecenas faucibus mollis interdum.

Bucket List

Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Cras mattis consectetur purus sit amet fermentum.

Bucket List

Maecenas sed diam eget risus varius blandit sit amet non magna.

Bucket List

Donec id elit non mi porta gravida at eget metus. Maecenas faucibus mollis interdum.

Bucket List

Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Cras mattis consectetur purus sit amet fermentum.

Bucket List

Maecenas sed diam eget risus varius blandit sit amet non magna.

- Ingress works with domain mapping (e.g., `/flask` path).

- PVCs are bound and pods have persistent volumes.
- Secrets are mounted/used securely.

Submission Requirements

- `README.md` documenting:
 - Steps to deploy
 - How to test each service (curl, browser, ingress)
 - Any issues faced and how you solved them
 - Kubernetes YAML files (`deployments/`, `services/`, `configmap/`, `secrets/`, etc.)
 - screen recording of:
 - Application running
 - `kubectll get all -A`
 - Browser or `curl` result
 - GitHub repo
-