Outlook

[Draft] (No subject)

From vispatster@live.com
Draft saved Mon 11/17/2025 6:43 AM

72.29.241.222 vispatster@live.com

For this assignment, you will work on productionizing a simplified Wikipedia-like API service which allows users to create posts. You can download the source code for the API service here.

The current version of the API service uses FastAPI for the business logic, stores data in SQLite, and exposes two Prometheus metrics through the `/metrics` endpoint, `users_created_total` for the total number of users created and `posts_created_total` for the total number of posts created.

This assignment contains 2 parts. Part 2 builds on top of part 1. Completing part 1 will give you 50% credit and completing part 2 will give you full credit — please attempt to complete part 1 even if you aren't able to complete part 2.

You are allowed to use coding assistants and the Internet to complete the assignment. Please **do not** consult with others, and complete the assignment by yourself — we will match your submission against other existing submissions to detect cheating.

Please submit assignment here:
https://docs.google.com/forms/d/e/1FAIpQLSfcaHKwh1B4KIT5iGcO5LWJ60g_1IwvbSV71n5aUSxRnogdJA/viewform

Part 1: Package environment as Helm chart

We want to productionize the service with the following components:

1. FastAPI: contains the business logic layer, packaged as a Docker container
2. PostgreSQL: stores data (you will need to configure FastAPI to replace SQLite with this)
3. Prometheus: collects data from `/metrics` endpoint of the FastAPI server
4. Grafana: visualizes the data, specifically the rate of users and posts creation over time in a single chart in a dashboard at `/d/creation-dashboard-678/creation`.

The cluster's ingress should expose the following endpoints:

- `/users/*`, `/posts/*`: routes the requests to FastAPI server
- `/grafana/d/creation-dashboard-678/creation`: routes the requests to the Grafana dashboard that displays the rate of users and posts creation. Please create this under username ***admin*** and password ***admin***.

You should package the FastAPI as Dockerfile, and the entire system as a Helm chart that installs all the required components in a Kubernetes cluster. The entire Kubernetes cluster should require at most 2 CPUs, 4GB RAM, 5 GB of disk of resources.

Deliverables**:** You must provide a single zip file that contains the following directory structure (**please follow this exact format so that our automatic evaluation doesn't fail your solution**):

```
/
|_wiki-service/
|   |_Dockerfile
|   |_ …
|_wiki-chart
```

1. `wiki-service/:` A directory that contains a Dockerfile for the FastAPI service along with all the files necessary to build it. Running `docker build .` should build the Wikipedia service in a container.
2. `wiki-chart/:` A Helm chart that packages all the system components. Please allow changing the name of the FastAPI service's Docker image name by setting `fastapi.image_name` in values.yaml.

Part 2: Containerizing the cluster with Docker-in-Docker and k3d

This section requires you to run the entire cluster in a Docker container. You will use k3d to run the cluster within the Docker container, and expose all the required endpoints from Part 1 (e.g. `/users/*`, `/posts/*`, `/grafana/d/creation-dashboard-678/creation`) through port 8080 of the container. You can use Docker-in-Docker to run Docker containers within your container — we will use --privileged (**note:** we will not mount Docker socket) when running your Docker image.

Deliverable**:** You must provide a single zip file that contains the following directory structure (**please follow this exact format so that our automatic evaluation doesn't fail your solution**):

```
/
|_wiki-service/
|   |_Dockerfile
|   |_ …
|_wiki-chart
|_Dockerfile
|_<any_helper_files>
```

1. `wiki-service/:` same as part 1.
2. `wiki-chart/:` same as part 1.
3. `Dockerfile`: A Dockerfile that builds and runs the whole cluster within a Docker container. We will build the image with `docker build .` and run the cluster with the proper permission.
4. Also include any other files necessary for `docker build .` to run successfully.
