vLLM OpenAI-Compatible Inference API with Docker and Kubernetes

Abrar Zahin

Overview

This project demonstrates how to build a scalable, OpenAI-compatible language model inference API using **vLLM**, containerize it with **Docker**, and deploy it using **Kubernetes**. Although it simulates a large model setup with `facebook/opt-125m`, the architecture supports deployment of very large models (e.g., 70B+ parameter LLMs).

Key Technologies

- **vLLM**: A high-performance LLM inference engine designed to serve large language models efficiently.
- **Docker**: A containerization platform to bundle the API and dependencies into a portable unit.
- **Kubernetes (K8s)**: An orchestration system to deploy and manage containerized applications in a scalable way.
- **Hugging Face Transformers**: Source of the pretrained model.

Project Structure

- `Dockerfile` - Defines how to containerize the vLLM inference API.
- `deployment.yaml` - Describes the Kubernetes Deployment object.
- `service.yaml` - Exposes the deployment externally using a LoadBalancer.
- `app.py` - (Optional) A simplified FastAPI version of the summarization app.
- `README.md` - Instructions for local and K8s setup.

What Does the API Do?

The vLLM server provides an OpenAI-compatible endpoint: `/v1/completions`. Clients can send POST requests with a prompt, and receive model-generated text completions in response.

How to Use Locally with Docker

1. Install Docker from docker.com.
2. Build the image:

   ```
   docker build -t vllm-api .
   ```

3. Run the container:

```
docker run -p 8000:8000 vllm-api
```

4. Test it using curl or Python:

```
curl -X POST http://localhost:8000/v1/completions \
  -H "Content-Type: application/json" \
  -d '{"model": "facebook/opt-125m", "prompt": "Hello AI,", "max_tokens": 50}'
```

How to Deploy with Kubernetes

1. Make sure you have `kubectl` and a K8s cluster set up (e.g., Minikube, GKE).
2. Apply the deployment and service manifests:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

3. Retrieve the external IP:

```
kubectl get service vllm-api-service
```

4. Send requests to the IP just like the local version.

Scaling and GPU Support

- The Kubernetes deployment requests a GPU using `nvidia.com/gpu: 1`.
- You can scale replicas by editing `replicas: 1` in `deployment.yaml`.
- Add autoscaling via a HorizontalPodAutoscaler (HPA) for CPU-based scaling.

Conclusion

This project is a complete template for serving LLMs with industry-grade tools like Docker and Kubernetes. It simulates large-scale inference setups and is ready to scale with real hardware and models in a production environment.