Ensure the project has two folders:

* frontend (Flask app)
* backend (Express app)

**2. Write Dockerfiles for Both Services**

In frontend/:

dockerfile

CopyEdit

FROM python:3.9-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

EXPOSE 5000

CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]

In backend/:

dockerfile

CopyEdit

FROM node:18-alpine

WORKDIR /app

COPY . .

RUN npm install

EXPOSE 3000

CMD ["npm", "start"]

**3. Build Docker Images & Load into Minikube**

bash

CopyEdit

minikube start

# For frontend

docker build -t ares-frontend:latest frontend/

minikube image load ares-frontend:latest

# For backend

docker build -t ares-backend:latest backend/

minikube image load ares-backend:latest

Loading images avoids pushing to DockerHub and speeds up deployment [Reddit+1Reddit+1](https://www.reddit.com/r/flask/comments/ncta2v?utm_source=chatgpt.com)[Reddit+14Towards Data Science+14CodingEasyPeasy+14](https://towardsdatascience.com/deploy-any-python-project-to-kubernetes-2c6ad4d41f14/?utm_source=chatgpt.com)[Medium](https://medium.com/%40yaa.meir/deploy-flask-app-in-kubernetes-24980e6e9378?utm_source=chatgpt.com).

**4. Create Kubernetes Manifests**

Under a k8s/ directory:

**a) backend-deployment.yaml**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata: { name: ares-backend }

spec:

replicas: 1

selector: { matchLabels: { app: ares-backend } }

template:

metadata: { labels: { app: ares-backend } }

spec:

containers:

- name: backend

image: ares-backend:latest

ports: [{ containerPort: 3000 }]

**b) backend-service.yaml**

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata: { name: ares-backend-svc }

spec:

selector: { app: ares-backend }

ports: [{ port: 3000, targetPort: 3000 }]

type: ClusterIP

**c) frontend-deployment.yaml**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata: { name: ares-frontend }

spec:

replicas: 1

selector: { matchLabels: { app: ares-frontend } }

template:

metadata: { labels: { app: ares-frontend } }

spec:

containers:

- name: frontend

image: ares-frontend:latest

env:

- name: BACKEND\_URL

value: "http://ares-backend-svc:3000"

ports: [{ containerPort: 5000 }]

**d) frontend-service.yaml**

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata: { name: ares-frontend-svc }

spec:

selector: { app: ares-frontend }

ports: [{ port: 80, targetPort: 5000 }]

type: NodePort

**5. Deploy to Minikube**

bash

CopyEdit

kubectl apply -f k8s/backend-deployment.yaml

kubectl apply -f k8s/backend-service.yaml

kubectl apply -f k8s/frontend-deployment.yaml

kubectl apply -f k8s/frontend-service.yaml

Verify:

bash

CopyEdit

kubectl get pods,svc

Expect to see both services and pods running.

**6. Access the Application**

To get the frontend URL:

bash

CopyEdit

minikube service ares-frontend-svc --url

Copy the returned URL (e.g. http://192.168.49.2:xxxxx) into your browser—it should load your Flask frontend, which communicates with the Express backend via the service name.

**7. (Optional) Add Ingress or Autoscaling**

To use an Ingress (e.g. host localhost) or enable HPA, you can adapt these techniques from advanced guides [Medium+2Towards Data Science+2CodingEasyPeasy+2](https://towardsdatascience.com/deploy-any-python-project-to-kubernetes-2c6ad4d41f14/?utm_source=chatgpt.com)[Reddit](https://www.reddit.com/r/kubernetes/comments/1iu92sy?utm_source=chatgpt.com).

**📝 Summary**

1. Clone your repo and confirm structure matches front-/backend split.
2. Dockerize both services.
3. Start Minikube and load your images.
4. Prepare Kubernetes manifests (Deployment & Services).
5. Apply the manifests to your cluster.
6. Use minikube service ... --url to browse the frontend.