Steps to Develop, Build & Deploy a Simple Flask App to a Minikube cluster

Following is a recipe for a **"Hello World" style web app** (Python/Flask backend + HTML frontend), Docker file and Minikube/Kubernetes deployment.

---

1. Prerequisites

- Install Docker
- Install Minikube
- Install kubectl

---

2. Create the Flask App

Project Structure

```
flask-app/
├── main.py          # Flask backend
├── templates/
│   └── index.html  # Frontend
└── Dockerfile      # Container setup
```

A. Backend (`main.py`)

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

B. Frontend (`templates/index.html`)

```html
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
    <style>
        body, html {
            margin: 0;
            padding: 0;
```

```
            height: 100%;
            display: flex;
            justify-content: center;
            align-items: center;
            text-align: center;
            font-family: Arial, sans-serif;
        }
        h1 {
            font-size: clamp(2rem, 5vw, 3rem);
        }
    </style>
</head>
<body>
    <h1>Hello World! 🎉</h1>
</body>
</html>
```

---

3. Containerize the App with Docker

A. Create a `Dockerfile`

```
# Use Python 3.10 base image
FROM python:3.10-slim

# Set working directory
WORKDIR /app

# Copy requirements first (for caching)
COPY requirements.txt .
RUN pip install -r requirements.txt

# Copy the rest of the app
COPY . .

# Expose port 5000 (Flask default)
EXPOSE 5000

# Run the app
CMD ["python", "main.py"]
```

B. Build the Docker Image

```
docker build -t flask-app .
```

---

4. Deploy to Minikube

A. Start Minikube & Use Its Docker Daemon

```
minikube start  # Start Minikube cluster
eval $(minikube docker-env)  # Use Minikube's Docker
```

For Windows i.e. PowerShell use:

```
& minikube -p minikube docker-env --shell powershell | Invoke-Expression
```

B. Rebuild the Image Inside Minikube

```
docker build -t flask-app .
```

C. Create a Kubernetes Deployment (`deployment-local.yaml`)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: flask-hello-world
        imagePullPolicy: Never  # Use local image
        ports:
        - containerPort: 5000
```

D. Expose the App with a Service (`service.yaml`)

```
apiVersion: v1
kind: Service
metadata:
  name: flask-service
spec:
```

```
  selector:
    app: flask-app
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
```

If working in a local context you cannot use a load balancer but if working in a cloud context you can add a `type: LoadBalancer` entry to the service.

E. Apply the Configs

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

---

5. Access the App

A. Get the Minikube Service URL

```
minikube service flask-service
```

This will open the app in your browser at `http://<minikube-ip>:<port>`.