Project: Kubernetes-Based IP Viewer Web App

Objective:

To build a simple web application that displays:

- The **public IP address** of the pod (as seen from outside).
- The **internal Pod IP address** (assigned by Kubernetes).

  Also demonstrates:

- **Replica scaling** to run multiple instances (pods) of the app.
- How each pod receives a unique Pod IP.

Tech Stack:

- **Frontend**: HTML + CSS + JavaScript (served using Flask)
- **Backend**: Python Flask
- **Containerization**: Docker
- **Orchestration**: Kubernetes

How It Works:

1. Frontend

- A minimal UI is built using HTML and CSS.
- It fetches the IP information via JavaScript using `/api/info`.

2. Backend (Flask)

- The `/` route serves the HTML page.
- The `/api/info` route returns a JSON object with:
  - **Public IP** → Fetched using ipify API.
  - **Pod IP** → Fetched using `socket.gethostbyname(socket.gethostname())`.

3. Dockerization

- A Dockerfile is used to:
  - Install dependencies (`Flask`, `requests`).
  - Copy the app code.

- Run the Flask app on port `80`.

4. Kubernetes Deployment

- The Docker image is deployed in a Kubernetes pod.
- When the pod runs:
  - Kubernetes assigns it an internal **Pod IP**.
  - The app inside the pod uses Flask to detect and display this Pod IP.
- A Kubernetes **Service** exposes the app externally, allowing access from a browser.

Replica Scaling Concept:

- In Kubernetes, when we scale the deployment to **multiple replicas**, each replica (pod):
  - Runs a separate instance of the application.
  - Is assigned a **unique Pod IP** by the Kubernetes network.
- All pods share the same **Service IP** (if exposed via ClusterIP or NodePort), but internally each pod has a different identity.

Verification (How to Test):

- Run `kubectl scale deployment ip-viewer --replicas=3`
- Use `kubectl get pods -o wide` to verify Pod IPs.
- Port-forward to each pod individually to see different Pod IPs:

  ```
  kubectl port-forward pod/<pod-name> 8080:80
  ```

- Refresh the browser for each to see the IP change.

Use Cases:

- Educational: Understanding IP concepts in Kubernetes.
- Debugging: Useful for developers to verify networking.
- DevOps: To quickly validate cluster and pod networking setup.

Output (Displayed in Browser):

```
Public IP:   203.0.113.45
Pod IP:      10.244.2.15
```

Benefits:

- Simple and visual.
- Demonstrates interaction between frontend, backend, container, and Kubernetes networking
- Great for learning about scaling, networking, and container behavior in Kubernetes..