DevSecOps Assessment

Step 1: Kubernetes Setup (Manual)

Tasks:

- Set up a **single-node Kubernetes cluster** on your local VM using `kubeadm`.
- Use **containerd** as the runtime and **Calico** as the CNI.
- Install **metrics-server** to support autoscaling.
- Enable **RBAC** and create a **non-admin service account** for deployments.
- Verify **cluster readiness**.

Security Additions:

- Generate and store kubeconfig for the service account separately (do not use the admin kubeconfig for app deployments).
- Ensure API server is only accessible from localhost or a specific IP range.

Step 2: CI/CD Pipeline

Preferred Tools: GitHub Actions **or** GitLab CI

Required Pipeline Steps:

1. **Security Scanning Stage** (before build):
    - Run **SAST** (Static Application Security Testing) for Python.
    - Run **dependency scanning** using a tool of your choice.

2. **Test Stage:**
    - Run Python unit tests using `pytest`.

3. **Build Stage:**
    - Build a **multi-stage Docker image** for the Flask app.
    - Include a **Dockerfile linting** step to catch insecure patterns.

4. **Image Security Stage:**
    - Scan Docker image before pushing.

5. **Push Stage:**
    - Push the image to DockerHub (or a local/private registry).
    - Tag Docker images with short Git SHA (e.g., `myapp:<sha>`).

6. **Deploy Stage:**
    - Deploy to Kubernetes using `helm upgrade --install`.

CI/CD Best Practices:

- Use **workflow secrets** for all credentials (no hardcoding).
- Separate stages: **security-scan**, **test**, **build**, **deploy**.
- Fail the pipeline if any **high severity vulnerabilities** are found.

Step 3: Deploy Flask App via Helm

Helm Chart Requirements:

- Externalize config using `values.yaml`.
- Add:
  - Readiness probe on `/healthz`
  - Liveness probe on `/failcheck`
- Set CPU & memory **requests and limits**.
- Add an **HPA** with a CPU target of 50%.
- Use a **Kubernetes secret** for any sensitive env vars.
- Ensure **network policies** only allow incoming traffic from specific namespaces or pods.

Step 4: Observability with EFK Stack

Tasks:

- Deploy **Elasticsearch, Fluent Bit, and Kibana** on the same cluster.
- Fluent Bit must:
  - Collect logs from your app pods.
  - Extract HTTP method, path, and status.
- Build a Kibana dashboard showing:
  - **Errors over time** (4xx, 5xx).
  - **Log volume trends**.

Security Additions:

- Restrict Kibana access via **basic authentication**.
- Ensure Elasticsearch is **TLS-enabled** with self-signed or provided certs.

Step 5: Documentation & Submission

Push to GitHub in this structure:

- /app/ # Flask app, Dockerfile, tests
- /chart/ # Helm chart
- /.github/ # GitHub Actions workflows (or /.gitlab-ci.yml)
- /docs/ # Kibana dashboard export + screenshots + README

README must include:

- How to set up the cluster.
- How to run the pipeline.
- Any security measures implemented.
- How to access the Kibana dashboard.