DevOps Assessment Task

Task Overview:

Scenario:

Your company has a simple **Node.js Express app** with a single endpoint (`/health`) that returns `{ "status": "OK" }`. The app needs to be **containerized**, tested, and deployed locally using **Docker**, with **automated CI/CD**, **monitoring**, and **logging**.

Your task is to:

1. **Containerize the application** using Docker.
2. **Automate CI/CD** using GitHub Actions or Jenkins.
3. **Deploy the application locally** on a Linux-based VM or a local machine.
4. **Set up monitoring/logging** using Prometheus, Grafana, or any lightweight tool.
5. **Automate the process** using Bash or Ansible.

Task Details:

1. Containerization (Docker)

- Create a **Dockerfile** to build a containerized version of the Node.js app.
- Use **Docker Compose** to manage the application and its dependencies (if needed).
- Ensure the `/health` endpoint is accessible at `http://localhost:3000/health`.

2. CI/CD Pipeline (GitHub Actions or Jenkins)

- Create a **CI/CD pipeline** that:
  - Runs tests (if any).
  - Builds a Docker image.
  - Deploys the container locally using **Docker Compose**.

3. Local Deployment Automation

- Write a **Bash script** or **Ansible playbook** to:
  - Install dependencies (Node.js, Docker, Docker Compose).
  - Build and run the application in a container.
  - Restart the container if it crashes.

4. Monitoring & Logging

- Set up **basic system monitoring** (Prometheus + Grafana, or Netdata).
- Capture logs and store them in a file (`logs/app.log`).
- Configure an alert (e.g., send a notification if CPU usage exceeds 80%).

Deliverables:

1. **GitHub Repo** with:
    - `Dockerfile`
    - `docker-compose.yml`
    - CI/CD configuration (`.github/workflows/main.yml` or `Jenkinsfile`).
    - Deployment script (`deploy.sh` or `ansible-playbook.yml`).
    - A `README.md` explaining:
        - Steps to set up the environment.
        - How to run the project.
2. **Working Application on the Candidate's Local Machine**
    - Run `curl http://localhost:3000/health` → Should return `{ "status": "OK" }`.

Bonus (Not Required but Nice to Have)

- Implement **NGINX as a reverse proxy**.
- Use **Systemd instead of a Bash script** for process management.
- Integrate **Slack/Discord notifications** for deployment status.