OESON CAPSTONE - End-to-End CI/CD and Infrastructure for a Microservice on Kubernetes

🌐 Cloud Provider:

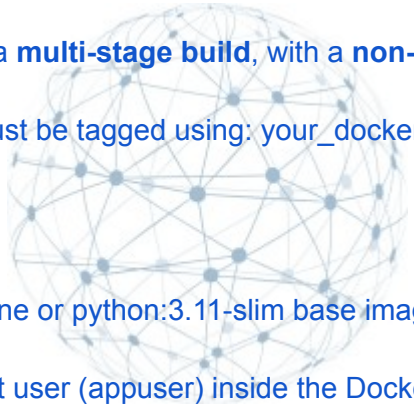
Use **AWS** (can be replaced with GCP or Azure if needed, but all tasks below assume AWS for clarity).

Task 1: Dockerize a Microservice (Secure Non-Root Build)

Goal: Containerize a Python Flask or Node.js app securely and efficiently.

Deliverables:

- Dockerfile using a **multi-stage build**, with a **non-root user**, minimal base image.
- Docker image must be tagged using: your_dockerhub_username/app-name:version.



Requirements:

- Use node:18-alpine or python:3.11-slim base image.
- Create a non-root user (appuser) inside the Dockerfile.
- Run the app on port 8080.
- Do not use root or run as UID 0.

Naming Convention:

- Dockerfile path: ./docker/Dockerfile
- App repo:

Task 2: Kubernetes Manifests (Resource Quotas & Security Context)

Goal: Deploy the Dockerized microservice to Kubernetes using manifests.

Deliverables:

- Deployment, Service, ConfigMap, Secret YAMLs.

- All YAMLs in k8s/ directory.
- Include livenessProbe, readinessProbe, and **resource limits**.

Requirements:

- Namespace: microservices
- Pod resource limits:
    - cpu: "250m"
    - memory: "512Mi"
- Use runAsNonRoot: true and readOnlyRootFilesystem: true.

Task 3: Infrastructure Provisioning with Terraform (AWS EC2 + VPC)

Goal: Provision a secure and minimal cloud infrastructure for the deployment.

Deliverables:

- Terraform code to create:
    - 1 VPC (devops-vpc)
    - 2 public subnets (subnet-a, subnet-b)
    - 1 EC2 instance (t2.micro) for Jenkins (jenkins-host)
    - 2 EC2 instances (t2.micro) for K8s (or EKS cluster)
    - Security groups for SSH (22), HTTP (80), NodePort (30000-32767)

Terraform output:

- Jenkins Public IP
- Kubernetes node IPs
- kubeconfig if EKS used

Naming Convention:

- Files in infra/terraform/

- Variables file: variables.tf
- Use Terraform Cloud or local backend.

Task 4: Jenkins Pipeline for CI/CD (Full Lifecycle)

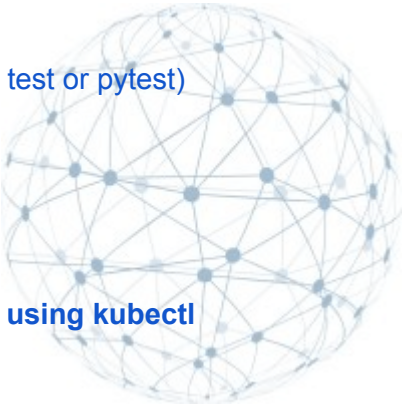
Goal: Write a declarative Jenkins pipeline to build, test, push, and deploy the microservice.

Deliverables:

- Jenkinsfile at root of the Git repo.
- Jenkins job auto-triggered on Git push.

Pipeline Stages:

1. **Checkout**
2. **Test** (unit tests via npm test or pytest)
3. **Build Docker Image**
4. **Push to DockerHub**
5. **Deploy to Kubernetes using kubectl**



Requirements:

- Store DockerHub and K8s credentials as Jenkins secrets.
- Use kubectl apply -f k8s/ for deployment.

Task 5: Monitoring Stack Setup (Prometheus + Grafana via Helm)

Goal: Monitor your Kubernetes cluster and microservice.

Deliverables:

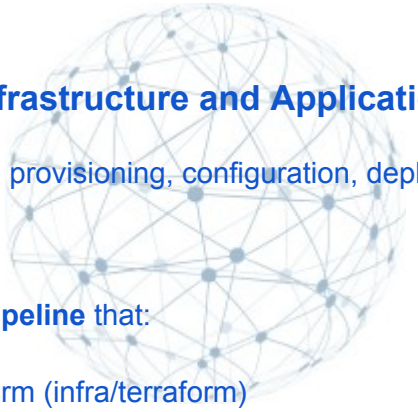
- Install Prometheus & Grafana.
- Configure ServiceMonitors to scrape your app.
- Export sample metrics using Prometheus client library.

Requirements:

- Dashboards pre-configured for:
    - CPU/Memory usage per pod.
    - HTTP request rate (if instrumented).
- Run Grafana on port 3000, with admin/admin login.

Naming:

- names: monitoring-prometheus, monitoring-grafana
- Config path: monitoring/



Task 6: End-to-End Infrastructure and Application Delivery Pipeline

Goal: Automate everything: provisioning, configuration, deployment, and monitoring.

Deliverables:

- A **single Jenkins pipeline** that:
    1. Runs Terraform (infra/terraform)
    2. Uses Ansible to install Jenkins, Docker, K8s, Helm
    3. Deploys the app using kubectl
    4. Sets up monitoring
- Use ansible-pull or remote Ansible execution.

Naming:

- Jenkinsfile: cd-pipeline/Jenkinsfile
- Terraform plan: infra/terraform/plan.tf
- Ansible playbooks: ansible/playbooks/*.yml

```
devops-capstone/
├── README.md
├── .gitignore
├── docker/
│   └── Dockerfile
│   └── entrypoint.sh
├── k8s/
│   ├── namespace.yaml
│   ├── deployment.yaml
│   ├── service.yaml
│   ├── configmap.yaml
│   └── secret.yaml
├── infra/
│   └── terraform/
│       ├── main.tf
│       ├── variables.tf
│       ├── outputs.tf
│       ├── providers.tf
│       └── backend.tf            # (optional for remote state)
├── ansible/
│   └── inventory.ini
│   └── playbooks/
│       ├── install-docker.yml
│       ├── install-k8s.yml
│       ├── install-jenkins.yml
│       ├── configure-pipeline.yml
│       ├── install-monitoring.yml
│       └── setup-app.yml
├── cd-pipeline/
│   └── Jenkinsfile
│   └── job-seed.groovy           # (optional: auto-create jobs
├── monitoring/
│   ├── namespace.yaml
│   ├── prometheus/
│   │   ├── config-map.yaml
│   │   ├── deployment.yaml
│   │   └── service.yaml
│   ├── grafana/
│   │   ├── config-map.yaml
│   │   ├── deployment.yaml
│   │   ├── service.yaml
│   │   └── dashboards/
│   │       └── microservice-dashboard.json
├── scripts/
│   ├── init-kubeconfig.sh
│   └── check-cluster.sh
└── app/
    ├── src/
    │   ├── index.js              # or app.py for Python
    │   └── server.js
    ├── tests/
    │   └── test_basic.js
    ├── package.json              # or requirements.txt for Pyt
    └── .env                      # local envi ment file (not
```