**DevOps Nexus: Automated CI/CD Pipeline for Kubernetes Deployment**

**GitHub repo:** <https://github.com/ShironKurian/Project_Nexus.git>

**1. Overview**

**DevOps Nexus** is an end‑to‑end pipeline that automates building, testing, containerizing, and deploying a Python Flask web application to AWS EKS, with Terraform‑provisioned RDS, ECR for image hosting, Jenkins for CI/CD, and Prometheus & Grafana for monitoring.

**Key Components:**

* **App**: Python Flask backend + HTML templates + PostgreSQL RDS
* **IaC**: Terraform for AWS resources (RDS, VPC, EKS components)
* **Containerization**: Docker + AWS ECR
* **Orchestration**: AWS EKS + kubectl
* **CI/CD**: Jenkins pipeline (checkout, build, test, push, deploy)
* **Monitoring**: Prometheus & Grafana deployed via Helm

**2. Architecture Diagram  
A diagram of a computer process

AI-generated content may be incorrect.**

**3. Infrastructure Provisioning (Terraform)**

1. **RDS Creation**
   * Defined in terraform/main.tf, variables.tf, backend.tf.
   * Backend state stored in an S3 bucket with DynamoDB locking.
2. **VPC & EKS Prerequisites**
   * Created custom VPC, subnets, security groups.
   * Outputs used by Jenkins for aws eks update-kubeconfig.

**4. Application Code**

* **app.py**:
  + REST endpoints for “task-manager” features.
  + Uses psycopg2 to connect to PostgreSQL RDS.
* **templates/**: HTML front‑end templates.
* **static/**: CSS/JS assets.
* **requirements.txt**:

Flask==2.2.5

psycopg2-binary==2.9.6

Werkzeug==2.2.3

**5. Dockerization**

**Dockerfile**:

FROM python:3.8-slim

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 5000

CMD ["python", "app.py"]

* **Build & Test Locally**

docker build -t task-master .

docker run -p 5000:5000 task-master

**6. AWS ECR**

1. **Create ECR Repo**

aws ecr create-repository --repository-name task-master --region us-east-1

1. **Tag & Push Image**

$(aws ecr get-login-password --region us-east-1) | \

docker login --username AWS --password-stdin 643716337997.dkr.ecr.us-east-1.amazonaws.com

docker tag task-master:latest 643716337997.dkr.ecr.us-east-1.amazonaws.com/task-master:latest

docker push 643716337997.dkr.ecr.us-east-1.amazonaws.com/task-master:latest

**7. Kubernetes Deployment (EKS)**

* **deployment.yaml**:

apiVersion: apps/v1

kind: Deployment

metadata:

name: task-manager

spec:

replicas: 2

selector: { matchLabels: { app: task-manager } }

template:

metadata: { labels: { app: task-manager } }

spec:

containers:

- name: task-master

image: 643716337997.dkr.ecr.us-east-1.amazonaws.com/task-master:latest

ports: [ { containerPort: 5000 } ]

---

apiVersion: v1

kind: Service

metadata: { name: task-manager-svc }

spec:

type: LoadBalancer

selector: { app: task-manager }

ports: [ { protocol: TCP, port: 80, targetPort: 5000 } ]

* **Apply**

aws eks update-kubeconfig --region us-east-1 --name task-manager-cluster

kubectl apply -f deployment.yaml

**8. CI/CD with Jenkins**

**Jenkinsfile** (final, fully‑working version):

pipeline {

agent any

environment {

AWS\_REGION = 'us-east-1'

ECR\_REGISTRY = '643716337997.dkr.ecr.us-east-1.amazonaws.com'

ECR\_REPO = 'task-master'

IMAGE\_NAME = "${ECR\_REGISTRY}/${ECR\_REPO}:latest"

}

stages {

stage('Checkout Code') {

steps {

git credentialsId: 'github-credentials',

url: 'https://github.com/ShironKurian/Project\_Nexus.git', branch: 'main'

}

}

stage('Build Docker Image') {

steps { sh "docker build -t ${ECR\_REPO} ." }

}

stage('Run Unit Tests') {

steps {

sh '''

python3 -m ensurepip --upgrade

python3 -m pip install --user --upgrade pip

python3 -m pip install --user -r requirements.txt

python3 -m pip install --user Werkzeug==2.2.3 pytest

export PATH=$HOME/.local/bin:$PATH

export PYTHONPATH=.

pytest tests/ --maxfail=1 --disable-warnings -q

'''

}

}

stage('Login to AWS ECR') {

steps {

withCredentials([usernamePassword(

credentialsId: 'aws-credentials',

usernameVariable: 'AWS\_ACCESS\_KEY\_ID',

passwordVariable: 'AWS\_SECRET\_ACCESS\_KEY'

)]) {

sh '''

aws configure set aws\_access\_key\_id $AWS\_ACCESS\_KEY\_ID

aws configure set aws\_secret\_access\_key $AWS\_SECRET\_ACCESS\_KEY

aws configure set default.region $AWS\_REGION

aws ecr get-login-password --region $AWS\_REGION |

docker login --username AWS --password-stdin $ECR\_REGISTRY

'''

}

}

}

stage('Push to ECR') {

steps {

sh '''

docker tag ${ECR\_REPO}:latest ${IMAGE\_NAME}

docker push ${IMAGE\_NAME}

'''

}

}

stage('Deploy to EKS') {

steps {

sh '''

aws eks update-kubeconfig --region $AWS\_REGION --name task-manager-cluster

kubectl apply -f deployment.yaml

'''

}

}

}

post {

success { echo '✅ Deployment Successful!' }

failure { echo '❌ Pipeline failed. Check logs.' }

}

}

**9. Monitoring with Prometheus & Grafana**

**Helm installation** (on your EKS bastion/Dev machine with helm installed):

# Add repos

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

# Create a namespace

kubectl create ns monitoring

# Install kube-prometheus-stack (Prometheus + Grafana)

helm upgrade --install monitoring prometheus-community/kube-prometheus-stack \

--namespace monitoring \

--set grafana.service.type=LoadBalancer

* **Verify services**:

kubectl get svc -n monitoring

* **Access URLs**
  + **Prometheus UI**: http://<prometheus-service-ip>:9090/
  + **Grafana UI**: http://<grafana-loadbalancer-dns>:80/

**10. Future Enhancements**

* **Blue/Green or Canary deployments** with Kubernetes
* **Automated security scans** (e.g. Trivy) in Jenkins
* **Infrastructure cost-optimization** dashboards in Grafana
* **Autoscaling** via Kubernetes HPA

**11. Screenshots Checklist :**https://github.com/ShironKurian/Project\_Nexus.git

A screenshot of a computer

AI-generated content may be incorrect.

1. **CI/CD Pipeline (Jenkins)**A screenshot of a computer

   AI-generated content may be incorrect.

**Pipeline Job configured:**A screenshot of a computer

AI-generated content may be incorrect. **Console Output:**A screenshot of a computer

AI-generated content may be incorrect.

**building docker image:**

A screenshot of a computer

AI-generated content may be incorrect. **Testing ://pytests**A screenshot of a computer

AI-generated content may be incorrect. **ECR Login and Push**A screenshot of a computer

AI-generated content may be incorrect.

**Triggering deployment.yaml**A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Docker & ECR**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Kubernetes (EKS)**A screen shot of a computer

   AI-generated content may be incorrect.A screen shot of a computer

   AI-generated content may be incorrect.A screenshot of a computer

   AI-generated content may be incorrect.

**deployment.yaml**

A screenshot of a computer program

AI-generated content may be incorrect.

1. **Prometheus and Grafana (Monitoring)**A screenshot of a computer

   AI-generated content may be incorrect.A screenshot of a computer

   AI-generated content may be incorrect. **Prometheus**A screenshot of a computer

   AI-generated content may be incorrect.
2. **Application**A screenshot of a computer

   AI-generated content may be incorrect.

**EKS**A screenshot of a computer

AI-generated content may be incorrect.

**Jenkins EC2 instance and worker node instances**A screenshot of a computer

AI-generated content may be incorrect.