Simple CI/CD Pipeline with Dependency Scanning, Validation, and Email Notifications for DigitalOcean Kubernetes

Grok

June 1, 2025

1 Introduction

This document provides an enhanced GitHub Actions pipeline to automate the build, scan, and deployment of a three-service application (React front-end, Node.js/Express backend, MongoDB database) to a DigitalOcean Kubernetes (DOKS) cluster. It reuses existing Kubernetes manifests, Terraform configurations, and Dockerfiles from the previous simple setup. New features include dependency vulnerability scanning, post-deployment validation, and email notifications. The pipeline is lightweight, suitable for development or small-scale deployments. Comments explain adaptability.

2 Prerequisites

- DigitalOcean account with PAT stored as `DO`$_TOKEN.DockerHubcredentialsstoredas$`DOCKERHUB`$_USER$

3 Reused Components

The following components are reused from the previous simple setup and not modified:

- **Terraform Configuration**: Provisions DOKS cluster (`terraform/main.tf`, `variables.tf`, `terraform.tfvars`).
- **Kubernetes Manifests**: Deploys the application (`k8s/secret.yaml`, `frontend-deployment.ya` `backend-deployment.yaml`, `database-deployment.yaml`).
- **Dockerfiles**: Builds frontend and backend images (`frontend/Dockerfile`, `backend/Dockerfile`
- **Previous Pipeline Features**: Trivy scanning for container images and basic rollback on deployment failure.

4 New Features in the CI/CD Pipeline

The pipeline introduces three new features:

4.1 Dependency Vulnerability Scanning

- **Tool**: `npm audit` scans Node.js dependencies for vulnerabilities.
- **Action**: Runs `npm audit -production` in frontend and backend directories, failing on critical or high-severity issues.
- **Benefit**: Prevents deployment of applications with vulnerable dependencies.
- **Adaptability**: Can use Dependabot or adjust severity levels (e.g., include medium vulnerabilities).

4.2 Post-Deployment Validation

- **Tool**: `curl` performs HTTP health checks.
- **Action**: Sends request to `/health` endpoint via LoadBalancer, failing if not 200 status.
- **Benefit**: Ensures the deployed application is functional.
- **Adaptability**: Can check additional endpoints or add basic load testing with `ab`.

4.3 Email Notifications

- **Tool**: SMTP server via `dawidd6/action-send-mail`.
- **Action**: Sends emails with commit SHA, status, and environment after build, deployment, or rollback.
- **Benefit**: Keeps team informed without external platforms.
- **Adaptability**: Supports different SMTP providers (e.g., Gmail, SendGrid) or additional recipients.

5 GitHub Actions Pipeline

Adding new features to the existing pipeline.

Listing 1: .github/workflows/cicd.yml

```
name: Simple CI/CD Pipeline
on:
  push:
    branches:
      - main
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'
      - name: Scan frontend dependencies
        run: npm audit --production
        working-directory: ./frontend
      - name: Scan backend dependencies
        run: npm audit --production
        working-directory: ./backend
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      - name: Log in to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      - name: Build frontend
        uses: docker/build-push-action@v5
        with:
          context: ./frontend
          file: ./frontend/Dockerfile
          push: false
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/frontend:${{ github.sha }}
          outputs: type=docker,dest=/tmp/frontend.tar
      - name: Scan frontend image
        uses: aquasecurity/trivy-action@0.24.0
        with:
          image-ref: /tmp/frontend.tar
          format: table
          exit-code: 1
          severity: CRITICAL,HIGH
      - name: Push frontend
        uses: docker/build-push-action@v5
        with:
          context: ./frontend
          file: ./frontend/Dockerfile
          push: true
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/frontend:${{ github.sha }},$
              {{ secrets.DOCKERHUB_USERNAME }}/frontend:latest
      - name: Build backend
        uses: docker/build-push-action@v5
        with:
          context: ./backend
          file: ./backend/Dockerfile
          push: false
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/backend:${{ github.sha }}
          outputs: type=docker,dest=/tmp/backend.tar
      - name: Scan backend image
        uses: aquasecurity/trivy-action@0.24.0
        with:
          image-ref: /tmp/backend.tar
          format: table
          exit-code: 1
          severity: CRITICAL,HIGH
      - name: Push backend
        uses: docker/build-push-action@v5
        with:
          context: ./backend
          file: ./backend/Dockerfile
          push: true
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/backend:${{ github.sha }},${{ secrets.DOCKERHUB_USERNAME }}/backend:latest
      - name: Notify email on build
        uses: dawidd6/action-send-mail@v3
        with:
          server_address: smtp.example.com
          server_port: 587
          username: ${{ secrets.SMTP_USERNAME }}
          password: ${{ secrets.SMTP_PASSWORD }}
          subject: ’Build Completed: Commit ${{ github.sha }}’
          to: team@example.com
          from: CI/CD Pipeline <noreply@example.com>
          body: ’Build completed for commit ${{ github.sha }} on main branch.’
      - name: Set up kubectl
        uses: azure/setup-kubectl@v3
        with:
          version: ’v1.28.0’
      - name: Configure kubeconfig
        run: |
          echo "${{ secrets.DO_KUBECONFIG }}" > kubeconfig.yaml
          export KUBECONFIG=./kubeconfig.yaml
      - name: Get current images
        run: |
          kubectl get deployment frontend -o jsonpath=’{.spec.template.spec.containers[0].image}’ > frontend_image.txt
          kubectl get deployment backend -o jsonpath=’{.spec.template.spec.containers[0].image}’ > backend_image.txt
      - name: Deploy to DOKS
        run: |
          kubectl apply -f k8s/secret.yaml
          kubectl apply -f k8s/frontend-deployment.yaml
          kubectl apply -f k8s/backend-deployment.yaml
          kubectl apply -f k8s/database-deployment.yaml
          kubectl rollout status deployment/frontend --timeout=300s
          kubectl rollout status deployment/backend --timeout=300s
        continue-on-error: true
      - name: Post-deployment validation
        run: |
          LB_IP=$(kubectl get svc frontend-service -o jsonpath=’{.status.loadBalancer.ingress[0].ip}’)
          curl -f http://$LB_IP/health || exit 1
      - name: Notify email on success
        if: success()
        uses: dawidd6/action-send-mail@v3
        with:
          server_address: smtp.example.com
          server_port: 587
          username: ${{ secrets.SMTP_USERNAME }}
          password: ${{ secrets.SMTP_PASSWORD }}
          subject: 'Deployment Succeeded: Commit ${{ github.sha }}'
          to: team@example.com
          from: CI/CD Pipeline <noreply@example.com>
          body: 'Deployment succeeded for commit ${{ github.sha }} on main branch.'
      - name: Rollback on failure
        if: failure()
        run: |
          PREV_FRONTEND=$(cat frontend_image.txt)
          PREV_BACKEND=$(cat backend_image.txt)
          kubectl set image deployment/frontend frontend=$PREV_FRONTEND
          kubectl set image deployment/backend backend=$PREV_BACKEND
          kubectl rollout status deployment/frontend --timeout=300s
          kubectl rollout status deployment/backend --timeout=300s
      - name: Notify email on rollback
        if: failure()
        uses: dawidd6/action-send-mail@v3
        with:
          server_address: smtp.example.com
          server_port: 587
          username: ${{ secrets.SMTP_USERNAME }}
          password: ${{ secrets.SMTP_PASSWORD }}
          subject: 'Deployment Failed: Commit ${{ github.sha }}'
          to: team@example.com
          from: CI/CD Pipeline <noreply@example.com>
          body: 'Deployment failed for commit ${{ github.sha }} on main branch. Rolled back to previous version.'
```

6 How to Run

1. Apply Terraform: `terraform init; terraform apply` in `terraform/`.
2. Save kubeconfig: `terraform output -raw kubeconfig > kubeconfig.yaml`.
3. Store secrets in GitHub: DO$_{T}OKEN,$ DO$_{K}UBECONFIG,$ DOCKERHUB$_{U}SERNAME,$ DOCKERHUB$_{T}$

4. `4. frontend/: Code and Dockerfile.`

   `5. backend/: Code and Dockerfile.`

   `6. k8s/: Manifests.`

   `7. terraform/: Terraform files.`

   `8. .github/workflows/cicd.yml: Pipeline.`

Push to `main` to trigger the pipeline.

Verify: `kubectl get pods -kubeconfig kubeconfig.yaml`.

Access: `kubectl get svc frontend-service`.

Destroy: `terraform destroy`.

7 Adapting to Requirement Changes

- **Change Region/Node Size**: Update `main.tf`.
- **Change Frameworks**: Update `Dockerfile`, manifests.
- **Change Database**: Update `database-deployment.yaml`.
- **Scaling**: Adjust `main.tf` or manifest replicas.
- **Adjust Scanning**: Modify `npm audit` severity.
- **Change Notifications**: Use different SMTP provider or recipients.
- **Extend Validation**: Add more endpoints or load tests.