Complex CI/CD Pipeline with Dependency Scanning, Container Signing, Validation, and Notifications for DigitalOcean Kubernetes

Grok

June 1, 2025

1 Introduction

This document provides an enhanced GitHub Actions pipeline to automate the build, test, scan, and deployment of a three-service application (React front-end, Node.js/Express backend, MongoDB database) to a DigitalOcean Kubernetes (DOKS) cluster. It builds on the previous complex pipeline, reusing the Helm chart, Terraform configurations, and Dockerfiles. New features include dependency vulnerability scanning, container image signing, post-deployment validation, and Slack notifications. The pipeline supports blue-green deployments with Istio and uses DigitalOcean Spaces for state and artifact storage. Comments explain adaptability.

2 Prerequisites

- DigitalOcean account with PAT stored as $\texttt{DO}_{T}OKEN.DigitalOceanSpacescredentialsstoredas\texttt{SPACES}_{A}$

3 Reused Components

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

- **Terraform Configuration**: Provisions DOKS cluster, VPC, Load Balancer, and Spaces (`terraform/main.tf`, `variables.tf`, `terraform.tfvars`).
- **Helm Chart**: Deploys the application with Istio for blue-green deployments (`my-app/Chart.yaml` `values.yaml`, `templates/*.yaml`).
- **Dockerfiles**: Multi-stage builds for frontend and backend (`frontend/Dockerfile`, `backend/Dockerfile`).
- **Previous Pipeline Features**: Unit/integration tests, Trivy scanning, automated rollbacks, and artifact storage in Spaces.

4 New Features in the CI/CD Pipeline

The pipeline introduces four 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 issues.
- **Benefit**: Catches vulnerable dependencies early.
- **Adaptability**: Can use Snyk or adjust severity levels.

4.2 Container Image Signing

- **Tool**: `cosign` signs and verifies Docker images.
- **Action**: Generates key pair, signs images post-push, and verifies before deployment.
- **Benefit**: Ensures image integrity and authenticity.
- **Adaptability**: Supports other registries or signing tools like Notary.

4.3 Post-Publishing Validation

- **Tool**: `curl` performs HTTP health checks.
- **Action**: Checks `/health` endpoint via LoadBalancer or Istio gateway for 200 status.
- **Benefit**: Confirms deployment success.
- **Adaptability**: Can integrate K6 for load testing.

4.4 Notification Integration

- **Tool**: Slack webhook for team notifications.
- **Action**: Sends messages on build, success, or rollback with commit and environment details.
- **Benefit**: Improves team visibility.
-
- **Adaptability**: Supports Discord, Teams, or email.

5 GitHub Actions Pipeline

Adding new features to the existing pipeline.

Listing 1: .github/workflows/cicd.yml

```yaml
name: Complex CI/CD Pipeline
on:
  push:
    branches:
      - main
      - staging
jobs:
  build-and-test:
    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
        working-directory: ./frontend
        run: npm audit --production
      - name: Install frontend dependencies
        run: npm install
        working-directory: ./frontend
      - name: Run frontend unit tests
        run: npm test
        working-directory: ./frontend
      - name: Scan backend dependencies
        run: npm audit --production
        working-directory: ./backend
      - name: Install backend dependencies
        run: npm install
        working-directory: ./backend
      - name: Run backend unit tests
        run: npm test
        working-directory: ./backend
      - name: Run backend integration tests
        run: npm run test:integration
        working-directory: ./backend
      - name: Set up Docker Buildx
        uses: docker/setup-buildx@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
```

```
51          tags: ${{ secrets.DOCKERHUB_USERNAME }}/frontend:${{ github.sha }}
52          outputs: type=docker,dest=/tmp/frontend.tar
53        - name: Scan frontend image
54          uses: aquasecurity/trivy-action@0.24.0
55          with:
56            image-ref: /tmp/frontend.tar
57            format: table
58            exit-code: 1
59            severity: CRITICAL,HIGH
60        - name: Push frontend image
61          uses: docker/build-push-action@v5
62          with:
63              context: ./frontend
64              image: ./frontend/Dockerfile
65              push: true
66              tags: ${{ secrets.DOCKERHUB_USERNAME }}/frontend:${{ github.sha }}
67        - name: Sign frontend image
68          uses: sigstore/cosign-installer@v3.1.0
69          run: |
70              echo "${{ secrets.COSIGN_PRIVATE_KEY }}" > cosign.key
71              cosign sign --key cosign.key -y ${{ secrets.DOCKERHUB_USERNAME }}/
                    frontend:${{ github.sha }}
72        - name: Build backend
73          uses: docker/build-push-action@v5
74          with:
75              context: ./backend/
76              file: ./backend/Dockerfile
77              push: false
78              tags: ${{ secrets.DOCKERHUB_USERNAME }}/backend:${{ github.sha }}
79              outputs: type=docker,dest=/tmp/backend.tar
80        - name: Scan backend image
81          uses: aquasecurity/trivy-action@0.24.0
82          with:
83            image-ref: /tmp/backend.tar
84            format: table
85            exit-code: 1
86            severity: CRITICAL,HIGH
87        - name: Push backend
88          uses: docker/build-push-action@v5
89          with:
90              context: ./backend/
91              file: ./backend/Dockerfile
92              push: true
93              tags: ${{ secrets.DOCKERHUB_USERNAME }}/backend:${{ github.sha }}
94        - name: Sign backend image
95          run: cosign
96          uses: sigstore/cosign-installer@v3.1.0
97            with:
98              cosign-release: 'v2.4.0'
99            run: |
100             echo "${{ secrets.COSIGN_PRIVATE_KEY }}" > cosign.key
```

```
          cosign sign --key cosign.key -y ${{ secrets.DOCKERHUB_USERNAME }}/backend:${{ github.sha }}
      - name: Upload artifacts to Spaces
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.SPACES_ACCESS_KEY }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.SPACES_SECRET_KEY }}
        run: |
          aws s3 cp /tmp/frontend.tar s3://app-artifacts/frontend-${{ github.sha }}.tar --endpoint-url https://nyc3.digitaloceanspaces.com
          aws s3 cp /tmp/backend.tar s3://app-artifacts/backend-${{ github.sha }}.tar --endpoint-url https://nyc3.digitaloceanspaces.com
      - name: Notify Slack on build
        uses: slackapi/slack-github-action@v1.23.0
        with:
          slack-bot-token: ${{ secrets.SLACK_WEBHOOK_URL }}
          channel-id: 'deployments'
          text: 'Build completed for commit ${{ github.sha }} on ${{ github.ref_name }}'
  deploy:
    needs: build-and-test
    runs-on: ubuntu-latest
    strategy:
      matrix:
        environment: [staging, production]
    steps:
      - uses: actions/checkout@v4
      - name: Set up Helm
        uses: azure/setup-helm@v4
        with:
          version: 'v3.13.0'
      - name: Set up kubectl
        uses: azure/setup-kubectl@v3
        with:
          version: 'v1.28.0'
      - name: Configure kubeconfig
        run: |
          echo "${{ secrets.DO_KUBECONFIG }}" > kubeconfig.yaml
          export KUBECONFIG=./kubeconfig.yaml
      - name: Verify image signatures
        run: |
          echo "${{ secrets.COSIGN_PUBLIC_KEY }}" > cosign.pub
          cosign verify --key cosign.pub ${{ secrets.DOCKERHUB_USERNAME }}/frontend:${{ github.sha }}
          cosign verify --key cosign.pub ${{ secrets.DOCKERHUB_USERNAME }}/backend:${{ github.sha }}
      - name: Install Istio
        run: |
          helm repo add istio https://istio-release.storage.googleapis.com/charts
          helm repo update
          kubectl create namespace istio-system || true
```

```
145           helm install istio-base istio/base -n istio-system --wait
146           helm install istiod istio/istiod -n istio-system --wait --set profile
                  =demo
147           helm install istio-ingress istio/gateway -n istio-system --wait
148           kubectl label namespace default istio-injection=enabled --overwrite
149       - name: Install Prometheus and Grafana
150         run: |
151           helm repo add prometheus-community https://prometheus-community.
                  github.io/helm-charts
152           helm repo update
153           helm install prometheus prometheus-community/prometheus -n istio-
                  system --wait
154           helm install grafana grafana/grafana -n istio-system --wait
155       - name: Deploy with Helm (Blue-Green)
156         run: |
157           RELEASE_VERSION=blue
158           if helm history my-app -n default | grep -q "blue"; then
159             RELEASE_VERSION=green
160           fi
161           helm upgrade --install my-app ./my-app -n default \
162             --set releaseVersion=$RELEASE_VERSION \
163             --set frontend.tag=${{ github.sha }} \
164             --set backend.tag=${{ github.sha }} \
165             --set mongodb.username=${{ secrets.MONGO_USER }} \
166             --set mongodb.password=${{ secrets.MONGO_PASSWORD }} \
167             --set istio.gateway.host=${{ matrix.environment == 'staging' && .
                    Values.istio.stagingHost || .Values.istio.gateway.host }}
168           kubectl rollout status deployment/frontend-$RELEASE_VERSION --timeout
                  =300s
169           kubectl rollout status deployment/backend-$RELEASE_VERSION --timeout
                  =300s
170         continue-on-error: true
171       - name: Post-deployment validation
172         run: |
173           LB_IP=$(kubectl get svc istio-ingress -n istio-system -o jsonpath='{.
                  status.loadBalancer.ingress[0].ip}')
174           curl -f http://$LB_IP/health || exit 1
175       - name: Notify Slack on success
176         if: success()
177         uses: slackapi/slack-github-action@v1.23.0
178         with:
179           slack-bot-token: ${{ secrets.SLACK_WEBHOOK_URL }}
180           channel-id: 'deployments'
181           text: 'Deployment succeeded for commit ${{ github.sha }} to ${{
                  matrix.environment }}'
182       - name: Rollback on failure
183         if: failure()
184         run: |
185           PREV_VERSION=blue
186           if [ "$RELEASE_VERSION" = "blue" ]; then
187             PREV_VERSION=green
```

```
        fi
        helm upgrade my-app ./my-app -n default \
          --set releaseVersion=$PREV_VERSION \
          --set frontend.tag=stable \
          --set backend.tag=stable \
          --set istio.gateway.host=${{ matrix.environment == 'staging' && .
              Values.istio.stagingHost || .Values.istio.gateway.host }}
        kubectl rollout status deployment/frontend-$PREV_VERSION --timeout
            =300s
        kubectl rollout status deployment/backend-$PREV_VERSION --timeout=300
            s
    - name: Notify Slack on rollback
      if: failure()
      uses: slackapi/slack-github-action@v1.23.0
      with:
        slack-bot-token: ${{ secrets.SLACK_WEBHOOK_URL }}
        channel-id: 'deployments'
        text: 'Deployment failed for commit ${{ github.sha }} to ${{ matrix.
            environment }}. 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. Generate `cosign` key pair: `cosign generate-key-pair`, store `cosign.key` as $\texttt{COSIGN}_{P}RIVATE_{K}$
4. `frontend/`: Code, Dockerfile, tests.
5. `backend/`: Code, Dockerfile, tests.
6. `my-app/`: Helm chart.
7. `terraform/`: Terraform files.
8. `.github/workflows/cicd.yml`: Pipeline.

Push to `staging` or `main` to trigger the pipeline.

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

Access: `terraform output loadbalancer`$_{i}pordomains($`staging.app.example.com, app.example.co`
`kubectl port-forward svc/grafana -n istio-system 3000:3000`, $visit$`http://localhost:3000`.

`Destroy: terraform destroy; helm uninstall my-app -n default; helm uninstall istio-base istiod istio-ingress prometheus grafana -n istio-system`.

7 Adapting to Requirement Changes

- **Change Region/Node Size**: Update `main.tf`.
- **Change Frameworks**: Update `Dockerfile`, `values.yaml`, `templates`.

- **Change Database**: Update `values.yaml`, `database-deployment.yaml`.
- **Scaling**: Adjust `main.tf` or `values.yaml`.
- **Adjust Scanning**: Modify `npm audit` or Trivy severity.
- **Change Signing**: Use different tools or registries.
- **Change Notifications**: Switch to Discord, Teams, or email.
- **Extend Validation**: Add performance or load tests.