Cloud-Native To-Do API on AWS EKS

0. What This Project Is — and Why It Matters

Business Context (Why Brite Systems Cares)\ Many of Brite's public- and private-sector clients need lightweight, highly available APIs (think internal task trackers, work-order queues, service tickets). While the functionality is simple, the delivery requirements are not: end-to-end automation, zero hard-coded secrets, fine-grained IAM, real-time observability, and rollback in under a minute.

This project is a *reference implementation* of those DevOps best practices in one self-contained microservice:

Aspect	How the Project Demonstrates It
CI/CD	GitHub Actions OIDC → AWS, reusable workflows, Trivy security scans, blue/green rollout
IaC	Terraform modules for VPC, RDS, ECR, EKS (state locked in DynamoDB)
Secrets Management	AWS Secrets Manager + CSI driver → env-mounted creds
Container Orchestration	EKS with managed node group (optionally Fargate)
Observability	Prometheus metrics, Grafana dashboards, CloudWatch Logs, explicit SLO & alert rules
Security & Cost	Least-privilege IAM, dependency scanning, cost-tagging, billing alerts

> **Outcome:** A deploy-ready pattern Brite consultants can clone and adapt for any small REST service, shortening client onboarding from **days to hours**.

0.1 Key Terms & Definitions

Term	Plain-English Meaning
FastAPI	Modern Python web framework for building high-performance REST APIs.
Docker	Tool that packages code + dependencies into portable containers.
Amazon ECR	AWS container registry where built images are stored.
Amazon EKS	Managed Kubernetes service that runs and scales Docker containers.
Terraform	Infrastructure-as-Code tool that declaratively provisions AWS resources.
GitHub Actions	CI/CD platform built into GitHub; workflows run on every PR/merge.
OIDC to AWS	OpenID Connect trust that lets GitHub Actions get short-lived AWS creds (no static keys).

Term	Plain-English Meaning
AWS Secrets Manager	Central vault that stores and rotates sensitive data (DB passwords, tokens).
PostgreSQL (RDS)	Relational database; Amazon RDS manages backups and patching.
Prometheus	Time-series database that scrapes and stores metrics.
Grafana	Dashboard UI that visualizes Prometheus metrics.
CloudWatch Logs	AWS log aggregation service for search and retention.
Blue/Green Rollout	Deploy strategy with two environments; instant rollback by flip of traffic.
SLO / p99 Latency	Service-level objective; 99th-percentile request time must stay below threshold.

Functional Demo Story

1. **API Consumer** (front-end or external service) hits `/todos` to create tasks.
2. Request is routed through an AWS ALB → EKS Service → FastAPI Pod.
3. Pod reads writer credentials from Secrets Manager, persists task to RDS PostgreSQL.
4. Metrics (HTTP 5XX, p99 latency) auto-exposed; logs stream to CloudWatch for audit.
5. On each merge to `main`, GitHub Actions runs tests, builds image, pushes to ECR, invokes Terraform apply → rolling upgrade with 30-second health gate.
6. Grafana and CloudWatch alarms validate success or auto-rollback.

---

1. High-Level Architecture

```
graph TD
  A[Developer: Push code<br>→ GitHub] -->|Pull Request| B[GitHub Actions CI]
  B --> C[Unit + Integration Tests]
  C --> D[Build Docker Image]
  D --> E[ECR: Push Image]
  E --> F[GitHub Actions CD]
  F -->|OIDC federated role| G[Terraform Plan/Apply]
  G --> H[EKS Cluster]
  H --> I[FastAPI Pods]
  H --> J[Prometheus Node Exporter]
  I --> K[(RDS PostgreSQL)]
  K -.->|Secrets ARN| L[AWS Secrets Manager]
  I --> M[CloudWatch Logs]
  J --> N[Prometheus Server]
  N --> O[Grafana Dashboards]
```

---

2. Requirements

2.1 Functional

- Expose RESTful **/todos** CRUD endpoints (FastAPI).
- Persist data in **PostgreSQL** (AWS RDS).
- CI pipeline auto-runs tests on pull request.
- CD pipeline builds, pushes, and deploys on merge → `main`.
- Secrets (DB creds, JWT signing key) resolved at runtime via **AWS Secrets Manager**.
- Metrics (HTTP latency, error rate, DB connections) exported to **Prometheus** and visualized in **Grafana**.
- Logs shipped to **CloudWatch Logs** with JSON structure.

2.2 Non-Functional

- Zero hard-coded credentials (OIDC → temporary AWS tokens).
- Rollback to previous image within 1 minute (Kubernetes Deployment revision).
- p99 latency < 300 ms for GET `/todos` under 50 RPS.
- 99.9 % uptime enforced via SLO + alert.
- Cost ceiling: < \$50 / month (dev environment).

2.3 Tech Stack

Layer	Choice
Language	Python 3.12 + FastAPI
Packaging	Poetry + pytest
Container	Docker, pushed to **Amazon ECR**
Orchestration	**AWS EKS** (Fargate profile optional)
IaC	**Terraform** (remote backend in S3, state-lock DynamoDB)
CI/CD	**GitHub Actions** (reusable workflows)
Observability	Prometheus + Grafana, CloudWatch Logs
Secrets	**AWS Secrets Manager**

---

3. Implementation Plan

Phase 1 – Local Proof of Concept (1 week)

1. Scaffold FastAPI project (`poetry new todo_api`).
2. Implement `/todos` CRUD with in-memory store + pytest coverage.
3. Dockerize app; run locally with compose + Postgres.

4. Write GitHub Actions CI (lint → test).

Phase 2 – AWS Foundation (1 week)

1. Provision VPC, subnets, and **RDS Postgres** via Terraform.
2. Create **ECR** repository and OIDC-enabled IAM role for GitHub Actions.
3. Store DB creds in **Secrets Manager**; attach least-privilege policy.

Phase 3 – EKS & CD (2 weeks)

1. Terraform module for EKS (managed node group or Fargate).
2. Helm chart (or K8s manifests) for FastAPI Deployment + Service.
3. Configure External Secrets Operator or Secrets Store CSI Driver → mount Secrets Manager secrets.
4. Extend GitHub Actions: build → scan (Trivy) → push → deploy (kubectl/helm via Terraform outputs).
5. Implement blue/green rollout using `kubectl rollout` + health checks.

Phase 4 – Observability & SRE (1 week)

1. Install **Prometheus Operator** + kube-state-metrics via Helm.
2. Instrument FastAPI with `prometheus-fastapi-instrumentator`.
3. Create Grafana dashboards (latency, RPS, error rate, DB connections).
4. Ship app logs to CloudWatch using Fluent Bit DaemonSet.
5. Define SLO (99.9 % availability) & alert rules (PagerDuty webhook placeholder).

Phase 5 – Hardening & Cost Controls (1 week)

1. Enable AWS Auto Scaling for node group; set min=1, max=3.
2. Add Terraform cost-tags; enable CloudWatch billing alert.
3. Integrate Snyk or Dependabot for dependency scanning.
4. Write run-book: rollback, restore DB snapshot, rotate secrets.

Deliverables

- Public GitHub repo with README, architecture diagram, and screencast link.
- Terraform `plan` & `apply` logs attached in `docs/`.
- Grafana dashboard JSON exported and committed.
- Cost report (AWS Cost Explorer screenshot).

Time-box: 6 weeks total (part-time). Can be trimmed to 3-4 weeks if prior AWS/EKS modules are reused.