fSmart Campus Event Management System - Implementation Plan

Project Timeline: 5 Weeks

Start Date: [To be determined]
End Date: [Start Date + 5 weeks]
Team Size: 5-6 members
Methodology: Agile with weekly sprints

Table of Contents

Team Structure and Roles

Recommended Team Configuration

```None
Project Lead / DevOps Engineer (1 person)
├── Overall project coordination
├── CI/CD pipeline design and implementation
├── Infrastructure automation
└── Team synchronization

Infrastructure Engineer (1 person)
├── EKS cluster provisioning
├── Terraform/IaC development
├── Network architecture
└── AWS service integration

Backend Developer (1-2 persons)
├── Events API development
├── Notification service development
├── Database schema design
└── API documentation

Frontend Developer (1 person)
├── React application development
├── UI/UX implementation
├── API integration
└── Frontend containerization

Monitoring & Security Lead (1 person)
├── Observability stack setup
├── Security implementation
├── Compliance documentation
└── Alerting configuration

Documentation Lead (Can be shared role)
├── Architecture diagrams
├── README and runbooks
├── API documentation
└── Deployment guides
```

RACI Matrix

Task	**Project Lead**	**Infra Engineer**	**Backend Dev**	**Frontend Dev**	**Monitor/Sec Lead**
Project Planning	R	C	C	C	C
Infrastructure Design	A	R	I	I	C
EKS Cluster Setup	A	R	I	I	C
Backend API	A	C	R	I	C
Frontend App	A	C	C	R	C

Task	**Project Lead**	**Infra Engineer**	**Backend Dev**	**Frontend Dev**	**Monitor/Sec Lead**
Database Schema	C	C	R	I	I
CI/CD Pipeline	R	C	C	C	C
Security Config	A	C	C	C	R
Monitoring Setup	A	C	C	C	R
Documentation	A	C	C	C	C

R=Responsible, A=Accountable, C=Consulted, I=Informed

Week 1: Planning & Design

Goals

- Complete system architecture design
- Set up development environment
- Initialize Git repository
- Create Infrastructure as Code foundation
- Define API contracts

Day 1-2: Project Kickoff and Architecture

Project Lead Tasks

- [ ] Organize kickoff meeting
- [ ] Review project requirements
- [ ] Set up project management board (GitHub Projects/Jira)
- [ ] Create communication channels (Slack, Discord, Teams)
- [ ] Define sprint schedule and standup times

Infrastructure Engineer Tasks

- [ ] Review AWS account access and permissions
- [ ] Set up AWS Organization (if needed)
- [ ] Configure AWS CLI and credentials
- [ ] Plan VPC architecture (3 AZs, CIDR blocks)
- [ ] Design network topology diagram

All Team Members

- [ ] Read project requirements
- [ ] Review architecture best practices
- [ ] Set up local development environment
- [ ] Install required tools (see checklist below)

Required Tools Checklist:

```shell
# Development Tools
- Git (v2.40+)
- Docker Desktop (v24.0+)
- kubectl (v1.31+)
- Terraform (v1.9+)
- AWS CLI (v2.15+)
- Helm (v3.14+)
- eksctl (v0.175+)
- Node.js (v20 LTS)
- Python (v3.12+)
- Code editor (VSCode recommended)

# Optional but Recommended
- k9s (Kubernetes CLI)
- stern (log viewing)
- kubectx/kubens
- terraform-docs
- pre-commit
```

Day 3-4: Repository Setup and IaC Foundation

Project Lead Tasks

- [ ] Create GitHub organization/repository
- [ ] Set up branch protection rules

- [ ] Create PR and issue templates
- [ ] Configure GitHub Actions secrets
- [ ] Set up project board with milestones

Repository Structure:

```None
campus-events-eks/
├── .github/
│   ├── workflows/           # GitHub Actions
│   ├── PULL_REQUEST_TEMPLATE.md
│   └── ISSUE_TEMPLATE/
├── terraform/
│   ├── environments/
│   │   ├── dev/
│   │   ├── staging/
│   │   └── prod/
│   ├── modules/
│   │   ├── eks/
│   │   ├── vpc/
│   │   ├── rds/
│   │   └── security/
│   └── README.md
├── kubernetes/
│   ├── base/                # Kustomize base
│   ├── overlays/
│   │   ├── dev/
│   │   ├── staging/
│   │   └── prod/
│   └── helm-charts/
├── applications/
│   ├── frontend/
│   ├── events-api/
│   └── notification-service/
├── scripts/
│   ├── setup.sh
│   ├── deploy.sh
│   └── teardown.sh
├── docs/
│   ├── ARCHITECTURE.md
│   ├── IMPLEMENTATION_PLAN.md
│   ├── RUNBOOK.md
│   └── API.md
├── .gitignore
├── .pre-commit-config.yaml
├── README.md
└── CONTRIBUTING.md
```

Infrastructure Engineer Tasks

- [ ] Create Terraform module structure
- [ ] Write VPC module
- [ ] Write EKS module (using EKS Blueprints)
- [ ] Write RDS module
- [ ] Configure remote state backend (S3 + DynamoDB)

VPC Module (terraform/modules/vpc/main.tf):

```None
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "~> 5.0"

  name = var.cluster_name
  cidr = var.vpc_cidr

  azs = var.availability_zones

  private_subnets  = var.private_subnet_cidrs
  public_subnets   = var.public_subnet_cidrs
  database_subnets = var.database_subnet_cidrs

  enable_nat_gateway   = true
  single_nat_gateway   = false
  enable_dns_hostnames = true
  enable_dns_support   = true

  enable_flow_log                      = true
  create_flow_log_cloudwatch_iam_role  = true
  create_flow_log_cloudwatch_log_group = true

  public_subnet_tags = {
    "kubernetes.io/role/elb" = "1"
  }

  private_subnet_tags = {
    "kubernetes.io/role/internal-elb" = "1"
    "karpenter.sh/discovery"          = var.cluster_name
  }

  tags = var.tags
}
```

Backend Developers Tasks

- [ ] Design database schema (PostgreSQL)
- [ ] Create ER diagrams
- [ ] Define API endpoints (OpenAPI spec)
- [ ] Write data models
- [ ] Create sample seed data

API Specification (OpenAPI 3.0):

```
None
openapi: 3.0.0
info:
  title: Campus Events API
  version: 1.0.0
  description: API for managing campus events

paths:
  /api/v1/events:
    get:
      summary: List all events
      parameters:
        - name: page
          in: query
          schema:
            type: integer
        - name: limit
          in: query
          schema:
            type: integer
        - name: category
          in: query
          schema:
            type: string
      responses:
        200:
          description: List of events

    post:
      summary: Create a new event
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/EventInput'
```

```yaml
      responses:
        201:
          description: Event created

  /api/v1/events/{eventId}:
    get:
      summary: Get event by ID
      parameters:
        - name: eventId
          in: path
          required: true
          schema:
            type: string
            format: uuid
      responses:
        200:
          description: Event details
        404:
          description: Event not found

    put:
      summary: Update event
      parameters:
        - name: eventId
          in: path
          required: true
          schema:
            type: string
            format: uuid
      responses:
        200:
          description: Event updated

    delete:
      summary: Delete event
      parameters:
        - name: eventId
          in: path
          required: true
          schema:
            type: string
            format: uuid
      responses:
        204:
          description: Event deleted

components:
```

```yaml
schemas:
  Event:
    type: object
    properties:
      id:
        type: string
        format: uuid
      title:
        type: string
        maxLength: 200
      description:
        type: string
      startDateTime:
        type: string
        format: date-time
      endDateTime:
        type: string
        format: date-time
      # ... additional fields
```

Frontend Developer Tasks

- [ ] Create React project structure
- [ ] Set up routing (React Router)
- [ ] Design component hierarchy
- [ ] Create wireframes/mockups
- [ ] Set up state management (Redux/Zustand)

Day 5: Documentation and Review

All Team Members

- [ ] Review architecture document
- [ ] Review API specifications
- [ ] Review database schema
- [ ] Provide feedback on designs
- [ ] Update documentation based on feedback

Deliverables for Week 1

- ✅ Complete architecture document
- ✅ Git repository with structure
- ✅ Terraform modules (VPC, EKS, RDS)
- ✅ Database schema and migrations

- ✅ API specification (OpenAPI)
- ✅ Frontend component design
- ✅ Project board with all tasks
- ✅ Team charter and communication plan

Week 2: Containerization

Goals

- Develop all microservices locally
- Create Dockerfiles for all services
- Test with Docker Compose
- Set up Amazon ECR
- Implement CI pipeline for builds

Day 6-7: Application Development

Backend Developers Tasks

Events API (Node.js/Express):

```
Shell
# Create project structure
cd applications/events-api
npm init -y
npm install express pg dotenv joi helmet cors morgan winston

# File structure
events-api/
├── src/
│   ├── config/
│   │   └── database.js
│   ├── controllers/
│   │   ├── eventController.js
│   │   └── rsvpController.js
│   ├── middleware/
│   │   ├── auth.js
│   │   ├── errorHandler.js
│   │   └── validator.js
│   ├── models/
│   │   ├── Event.js
│   │   └── RSVP.js
│   ├── routes/
│   │   ├── events.js
│   │   └── rsvps.js
│   ├── services/
│   │   └── eventService.js
│   ├── utils/
│   │   └── logger.js
│   └── app.js
├── tests/
├── Dockerfile
├── .dockerignore
├── package.json
└── README.md
```

Tasks:

- [ ] Implement event CRUD operations
- [ ] Implement RSVP functionality
- [ ] Add input validation
- [ ] Implement error handling
- [ ] Add logging middleware
- [ ] Write unit tests
- [ ] Add health check endpoints

Notification Service (Python/FastAPI):

```shell
# Create project structure
cd applications/notification-service
python -m venv venv
source venv/bin/activate
pip install fastapi uvicorn boto3 jinja2 pydantic

# File structure
notification-service/
├── app/
│   ├── api/
│   │   └── endpoints/
│   ├── core/
│   │   ├── config.py
│   │   └── logging.py
│   ├── models/
│   │   └── notification.py
│   ├── services/
│   │   ├── email_service.py
│   │   ├── sms_service.py
│   │   └── queue_service.py
│   ├── templates/
│   │   └── email/
│   └── main.py
├── tests/
├── Dockerfile
├── requirements.txt
└── README.md
```

Tasks:

- [ ] Implement notification service
- [ ] Add SNS integration
- [ ] Add SES integration
- [ ] Create email templates
- [ ] Implement SQS consumer
- [ ] Add retry logic
- [ ] Write unit tests

Frontend Developer Tasks

React Application:

```shell
# Create React app
npx create-react-app frontend
cd frontend

# Install dependencies
npm install react-router-dom axios @mui/material @emotion/react @emotion/styled

# File structure
frontend/
├── public/
├── src/
│   ├── components/
│   │   ├── EventList/
│   │   ├── EventDetail/
│   │   ├── EventForm/
│   │   └── Navigation/
│   ├── pages/
│   │   ├── Home.jsx
│   │   ├── Events.jsx
│   │   ├── EventDetail.jsx
│   │   └── CreateEvent.jsx
│   ├── services/
│   │   └── api.js
│   ├── utils/
│   ├── App.jsx
│   └── index.jsx
├── Dockerfile
├── nginx.conf
└── package.json
```

Tasks:

- [ ] Implement event listing page
- [ ] Implement event detail page
- [ ] Implement event creation form
- [ ] Implement RSVP functionality
- [ ] Add search and filters
- [ ] Add responsive design
- [ ] Write component tests

Day 8-9: Containerization

All Developers Tasks

Frontend Dockerfile (Multi-stage):

```
None
# Build stage
FROM node:20-alpine AS build

WORKDIR /app

# Copy package files
COPY package*.json ./
```

```
# Install dependencies
RUN npm ci --only=production

# Copy source code
COPY . .

# Build application
RUN npm run build

# Production stage
FROM nginx:1.25-alpine

# Copy built assets
COPY --from=build /app/build /usr/share/nginx/html

# Copy nginx configuration
COPY nginx.conf /etc/nginx/conf.d/default.conf

# Create non-root user
RUN addgroup -g 101 -S nginx && \
    adduser -S -D -H -u 101 -h /var/cache/nginx -s /sbin/nologin -G nginx -g
nginx nginx && \
    chown -R nginx:nginx /usr/share/nginx/html && \
    chown -R nginx:nginx /var/cache/nginx && \
    chown -R nginx:nginx /var/log/nginx && \
    touch /var/run/nginx.pid && \
    chown -R nginx:nginx /var/run/nginx.pid

USER nginx

EXPOSE 3000

HEALTHCHECK --interval=30s --timeout=3s --start-period=10s --retries=3 \
  CMD wget --no-verbose --tries=1 --spider http://localhost:3000/health || exit
1

CMD ["nginx", "-g", "daemon off;"]
```

Events API Dockerfile:

```
None
# Build stage
FROM node:20-alpine AS build
```

```
WORKDIR /app

# Copy package files
COPY package*.json ./

# Install all dependencies
RUN npm ci

# Copy source code
COPY . .

# Production stage
FROM node:20-alpine

WORKDIR /app

# Install dumb-init for proper signal handling
RUN apk add --no-cache dumb-init

# Create non-root user
RUN addgroup -g 1001 -S nodejs && \
    adduser -S nodejs -u 1001

# Copy dependencies and built code
COPY --from=build --chown=nodejs:nodejs /app/node_modules ./node_modules
COPY --chown=nodejs:nodejs . .

# Remove dev dependencies
RUN npm prune --production

USER nodejs

EXPOSE 8080

HEALTHCHECK --interval=30s --timeout=5s --start-period=10s --retries=3 \
  CMD node healthcheck.js || exit 1

ENTRYPOINT ["dumb-init", "--"]
CMD ["node", "src/app.js"]
```

Notification Service Dockerfile:

```
None
# Build stage
```

```
FROM python:3.12-alpine AS build

WORKDIR /app

# Install build dependencies
RUN apk add --no-cache gcc musl-dev

# Copy requirements
COPY requirements.txt .

# Install Python dependencies
RUN pip install --no-cache-dir --user -r requirements.txt

# Production stage
FROM python:3.12-alpine

WORKDIR /app

# Create non-root user
RUN addgroup -g 1001 -S python && \
    adduser -S python -u 1001

# Copy dependencies
COPY --from=build --chown=python:python /root/.local /home/python/.local
COPY --chown=python:python . .

# Add local bin to PATH
ENV PATH=/home/python/.local/bin:$PATH

USER python

EXPOSE 8000

HEALTHCHECK --interval=30s --timeout=5s --start-period=10s --retries=3 \
  CMD python healthcheck.py || exit 1

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Docker Ignore Files:

```
None
# .dockerignore
node_modules
npm-debug.log
```

```
.git
.env
.env.local
.DS_Store
*.md
tests
*.test.js
coverage
dist
.vscode
.idea
```

Tasks for All Services

- ☐ Create optimized Dockerfiles
- ☐ Implement health check endpoints
- ☐ Create .dockerignore files
- ☐ Test builds locally
- ☐ Optimize image sizes
- ☐ Scan for vulnerabilities

Day 10: Docker Compose Testing

Project Lead Tasks

Create Docker Compose for Local Testing:

```
None
# docker-compose.yml
version: '3.9'

services:
  postgres:
    image: postgres:16-alpine
    environment:
      POSTGRES_DB: campus_events
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    ports:
      - "5432:5432"
    volumes:
      - postgres-data:/var/lib/postgresql/data
      - ./scripts/init-db.sql:/docker-entrypoint-initdb.d/init.sql
```

```yaml
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 10s
      timeout: 5s
      retries: 5

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 3s
      retries: 5

  events-api:
    build:
      context: ./applications/events-api
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
    environment:
      NODE_ENV: development
      DATABASE_URL: postgresql://postgres:postgres@postgres:5432/campus_events
      REDIS_URL: redis://redis:6379
      PORT: 8080
    depends_on:
      postgres:
        condition: service_healthy
      redis:
        condition: service_healthy
    volumes:
      - ./applications/events-api/src:/app/src

  notification-service:
    build:
      context: ./applications/notification-service
      dockerfile: Dockerfile
    ports:
      - "8000:8000"
    environment:
      DATABASE_URL: postgresql://postgres:postgres@postgres:5432/campus_events
      AWS_REGION: us-east-1
      AWS_ACCESS_KEY_ID: ${AWS_ACCESS_KEY_ID}
      AWS_SECRET_ACCESS_KEY: ${AWS_SECRET_ACCESS_KEY}
    depends_on:
      postgres:
        condition: service_healthy
    volumes:
      - ./applications/notification-service/app:/app/app

  frontend:
    build:
      context: ./applications/frontend
      dockerfile: Dockerfile
      target: build
    ports:
      - "3000:3000"
    environment:
      REACT_APP_API_URL: http://localhost:8080
    volumes:
      - ./applications/frontend/src:/app/src
      - /app/node_modules
    depends_on:
      - events-api

volumes:
  postgres-data:
  redis-data:
```

Tasks:

- [ ] Create Docker Compose file
- [ ] Test all services together
- [ ] Verify service communication
- [ ] Test database connections
- [ ] Test API endpoints
- [ ] Document any issues

Infrastructure Engineer Tasks

- [ ] Set up Amazon ECR repositories
- [ ] Configure ECR scanning
- [ ] Create IAM roles for ECR access
- [ ] Document ECR push process

ECR Setup Script:

```
Shell
#!/bin/bash

# Create ECR repositories
aws ecr create-repository \
  --repository-name campus-events/frontend \
  --image-scanning-configuration scanOnPush=true \
  --encryption-configuration encryptionType=AES256

aws ecr create-repository \
  --repository-name campus-events/events-api \
  --image-scanning-configuration scanOnPush=true \
  --encryption-configuration encryptionType=AES256

aws ecr create-repository \
  --repository-name campus-events/notification-service \
  --image-scanning-configuration scanOnPush=true \
  --encryption-configuration encryptionType=AES256

# Set lifecycle policies
aws ecr put-lifecycle-policy \
  --repository-name campus-events/frontend \
  --lifecycle-policy-text file://ecr-lifecycle-policy.json
```

Day 11: CI Pipeline Setup

Project Lead / DevOps Tasks

GitHub Actions Workflow for Build:

```
None
# .github/workflows/build.yml
name: Build and Push Images

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

env:
  AWS_REGION: us-east-1
  ECR_REGISTRY: ${{ secrets.AWS_ACCOUNT_ID }}.dkr.ecr.us-east-1.amazonaws.com
```

```yaml
jobs:
  build-frontend:
    runs-on: ubuntu-latest
    permissions:
      id-token: write
      contents: read
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: ${{ secrets.AWS_ROLE_ARN }}
          aws-region: ${{ env.AWS_REGION }}

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: ./applications/frontend
          push: ${{ github.event_name != 'pull_request' }}
          tags: |
            ${{ env.ECR_REGISTRY }}/campus-events/frontend:${{ github.sha }}
            ${{ env.ECR_REGISTRY }}/campus-events/frontend:latest
          cache-from: type=gha
          cache-to: type=gha,mode=max

      - name: Scan image with Trivy
        uses: aquasecurity/trivy-action@master
        with:
          image-ref: ${{ env.ECR_REGISTRY }}/campus-events/frontend:${{
github.sha }}
          format: 'sarif'
          output: 'trivy-results.sarif'

      - name: Upload Trivy results to GitHub Security
        uses: github/codeql-action/upload-sarif@v2
        if: always()
        with:
          sarif_file: 'trivy-results.sarif'
```

```
# Repeat for events-api and notification-service
```

Deliverables for Week 2

- ✅ All microservices developed and tested locally
- ✅ Dockerfiles for all services
- ✅ Docker Compose setup working
- ✅ Amazon ECR repositories created
- ✅ CI pipeline for image builds
- ✅ Vulnerability scanning integrated
- ✅ Images pushed to ECR

Week 3: EKS Cluster Setup

Goals

- Provision EKS cluster using Terraform
- Configure node groups
- Set up kubectl access
- Deploy platform add-ons
- Test basic deployments

Day 12-13: Infrastructure Provisioning

Infrastructure Engineer Tasks

Main Terraform Configuration:

```None
# terraform/environments/dev/main.tf
terraform {
  required_version = ">= 1.9"

  backend "s3" {
    bucket         = "campus-events-terraform-state"
    key            = "dev/terraform.tfstate"
    region         = "us-east-1"
    encrypt        = true
    dynamodb_table = "terraform-state-lock"
  }

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
    kubernetes = {
      source  = "hashicorp/kubernetes"
      version = "~> 2.23"
    }
    helm = {
      source  = "hashicorp/helm"
      version = "~> 2.11"
    }
  }
}

provider "aws" {
  region = var.aws_region

  default_tags {
    tags = {
      Project     = "campus-events"
      Environment = var.environment
      ManagedBy   = "terraform"
    }
  }
}

# Local variables
locals {
  cluster_name = "campus-events-${var.environment}"

  tags = {
    Project     = "campus-events"
    Environment = var.environment
  }
}

# VPC Module
module "vpc" {
  source = "../../modules/vpc"

  cluster_name = local.cluster_name
  vpc_cidr     = "10.0.0.0/16"
  availability_zones = ["us-east-1a", "us-east-1b", "us-east-1c"]

  private_subnet_cidrs  = ["10.0.11.0/24", "10.0.12.0/24", "10.0.13.0/24"]
  public_subnet_cidrs   = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  database_subnet_cidrs = ["10.0.21.0/24", "10.0.22.0/24", "10.0.23.0/24"]

  tags = local.tags
}

# EKS Module (using EKS Blueprints pattern)
module "eks" {
  source  = "terraform-aws-modules/eks/aws"
  version = "~> 20.0"

  cluster_name    = local.cluster_name
  cluster_version = "1.31"

  # Cluster endpoint access
  cluster_endpoint_public_access  = false
  cluster_endpoint_private_access = true

  # Enable IRSA
  enable_irsa = true

  # Control plane logging
  cluster_enabled_log_types = [
    "api",
    "audit",
    "authenticator",
    "controllerManager",
    "scheduler"
  ]

  vpc_id     = module.vpc.vpc_id
  subnet_ids = module.vpc.private_subnets

  # Node groups
  eks_managed_node_groups = {
    general = {
      name           = "${local.cluster_name}-general"
      instance_types = ["m5.xlarge"]
      capacity_type  = "SPOT"

      min_size     = 2
      max_size     = 10
      desired_size = 3
      subnet_ids = module.vpc.private_subnets

      labels = {
        Environment  = var.environment
        WorkloadType = "general"
      }

      metadata_options = {
        http_endpoint               = "enabled"
        http_tokens                 = "required"
        http_put_response_hop_limit = 1
        instance_metadata_tags      = "enabled"
      }

      block_device_mappings = {
        xvda = {
          device_name = "/dev/xvda"
          ebs = {
            volume_size           = 100
            volume_type           = "gp3"
            iops                  = 3000
            throughput            = 125
            encrypted             = true
            delete_on_termination = true
          }
        }
      }
    }
  }

  # Cluster add-ons
  cluster_addons = {
    coredns = {
      most_recent = true
    }
    kube-proxy = {
      most_recent = true
    }
    vpc-cni = {
      most_recent = true
      configuration_values = jsonencode({
        env = {
          ENABLE_PREFIX_DELEGATION          = "true"
          ENABLE_POD_ENI                    = "true"
          POD_SECURITY_GROUP_ENFORCING_MODE = "standard"
        }
      })
    }
    aws-ebs-csi-driver = {
      most_recent              = true
      service_account_role_arn = module.ebs_csi_driver_irsa.iam_role_arn
    }
  }

  tags = local.tags
}

# EKS Blueprints Addons
module "eks_blueprints_addons" {
  source  = "aws-ia/eks-blueprints-addons/aws"
  version = "~> 1.0"

  cluster_name      = module.eks.cluster_name
  cluster_endpoint  = module.eks.cluster_endpoint
  cluster_version   = module.eks.cluster_version
  oidc_provider_arn = module.eks.oidc_provider_arn

  # Add-ons
  enable_aws_load_balancer_controller = true
  enable_external_dns                 = true
  enable_external_secrets             = true
  enable_metrics_server               = true
  enable_kube_prometheus_stack        = true
  enable_karpenter                    = true

  # Karpenter configuration
  karpenter = {
    repository_username =
data.aws_ecrpublic_authorization_token.token.user_name
    repository_password = data.aws_ecrpublic_authorization_token.token.password
  }

  # External DNS configuration
  external_dns = {
    values = [
      <<-EOT
      provider: aws
      domainFilters:
        - ${var.domain_name}
      policy: sync
      txtOwnerId: ${local.cluster_name}
      EOT
    ]
  }

  # Prometheus stack configuration
  kube_prometheus_stack = {
    values = [
      <<-EOT
      prometheus:
        prometheusSpec:
          retention: 15d
          storageSpec:
            volumeClaimTemplate:
              spec:
                accessModes: ["ReadWriteOnce"]
                resources:
                  requests:
                    storage: 50Gi
      grafana:
        adminPassword: ${random_password.grafana_admin.result}
        persistence:
          enabled: true
          size: 10Gi
      EOT
    ]
  }

  tags = local.tags
}

# RDS Module
module "rds" {
  source = "../../modules/rds"

  identifier     = "${local.cluster_name}-postgres"
  engine_version = "16.3"

  instance_class    = "db.t3.medium"
  allocated_storage = 100
  storage_type      = "gp3"

  db_name  = "campus_events"
  username = "postgres"

  vpc_id               = module.vpc.vpc_id
  subnet_ids           = module.vpc.database_subnets
  allowed_cidr_blocks  = module.vpc.private_subnets_cidr_blocks

  multi_az               = true
  backup_retention_period = 7
  enabled_cloudwatch_logs_exports = ["postgresql", "upgrade"]

  tags = local.tags
}

# Outputs
output "cluster_endpoint" {
  description = "Endpoint for EKS control plane"
  value       = module.eks.cluster_endpoint
}

output "cluster_name" {
  description = "Kubernetes Cluster Name"
  value       = module.eks.cluster_name
}

output "configure_kubectl" {
  description = "Configure kubectl"
  value       = "aws eks update-kubeconfig --region ${var.aws_region} --name
${module.eks.cluster_name}"
}

output "rds_endpoint" {
  description = "RDS endpoint"
  value       = module.rds.db_instance_endpoint
  sensitive   = true
}
```

Tasks:

- [ ] Review and validate Terraform code
- [ ] Initialize Terraform backend
- [ ] Plan infrastructure changes
- [ ] Apply Terraform (create VPC first)
- [ ] Apply Terraform (create EKS cluster)
- [ ] Apply Terraform (create add-ons)
- [ ] Verify cluster is healthy

Execution Steps:

```
Shell
# Initialize Terraform
cd terraform/environments/dev
terraform init

# Plan and apply VPC
terraform plan -target=module.vpc
terraform apply -target=module.vpc -auto-approve

# Plan and apply EKS
terraform plan -target=module.eks
terraform apply -target=module.eks -auto-approve

# Plan and apply add-ons
terraform plan
terraform apply -auto-approve

# Configure kubectl
aws eks update-kubeconfig --region us-east-1 --name campus-events-dev

# Verify cluster
kubectl get nodes
kubectl get pods -A
```

Day 14: Kubernetes Configuration

All Team Members Tasks

Create Namespaces:

```
None
# kubernetes/base/namespaces.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: dev
  labels:
    name: dev
    pod-security.kubernetes.io/enforce: restricted
---
apiVersion: v1
kind: Namespace
metadata:
  name: staging
  labels:
    name: staging
    pod-security.kubernetes.io/enforce: restricted
---
apiVersion: v1
kind: Namespace
metadata:
  name: prod
  labels:
    name: prod
    pod-security.kubernetes.io/enforce: restricted
---
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring
  labels:
    name: monitoring
```

Set up RBAC:

```None
# kubernetes/base/rbac.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-secrets
  namespace: prod
  annotations:
    eks.amazonaws.com/role-arn:
arn:aws:iam::ACCOUNT_ID:role/external-secrets-role
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: developer-role
rules:
- apiGroups: ["", "apps", "batch"]
  resources: ["pods", "deployments", "jobs", "services"]
  verbs: ["get", "list", "watch", "create", "update", "patch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: developer-binding
```

```
subjects:
- kind: Group
  name: developers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: developer-role
  apiGroup: rbac.authorization.k8s.io
```

Tasks:

- ☐ Create all namespaces
- ☐ Configure RBAC policies
- ☐ Set up service accounts
- ☐ Configure network policies
- ☐ Test kubectl access
- ☐ Document access procedures

Day 15-16: Deploy Platform Services

Monitoring & Security Lead Tasks

Configure External Secrets Operator:

```
None
# kubernetes/base/external-secrets.yaml
apiVersion: external-secrets.io/v1beta1
kind: SecretStore
metadata:
  name: aws-secrets-manager
  namespace: prod
spec:
  provider:
    aws:
      service: SecretsManager
      region: us-east-1
      auth:
        jwt:
          serviceAccountRef:
            name: external-secrets
```

Configure Prometheus/Grafana:

```shell
# Access Grafana
kubectl port-forward -n monitoring svc/kube-prometheus-stack-grafana 3000:80

# Get Grafana password
kubectl get secret -n monitoring kube-prometheus-stack-grafana \
  -o jsonpath="{.data.admin-password}" | base64 --decode
```

Tasks:

- [ ] Configure External Secrets Operator
- [ ] Set up Prometheus scrapers
- [ ] Import Grafana dashboards
- [ ] Configure alert rules
- [ ] Test monitoring stack
- [ ] Document access procedures

Deliverables for Week 3

- ✅ EKS cluster running in 3 AZs
- ✅ RDS PostgreSQL database provisioned
- ✅ All platform add-ons installed
- ✅ Namespaces and RBAC configured
- ✅ Monitoring stack operational
- ✅ kubectl access documented
- ✅ Basic smoke tests passed

Week 4: Load Balancing & Scaling

Goals

- Deploy applications to EKS
- Configure ALB Ingress
- Set up auto-scaling
- Implement network policies
- Performance testing

Day 17-18: Application Deployment

All Developers Tasks

Kubernetes Manifests Using Kustomize:

```
None
# kubernetes/base/frontend/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      securityContext:
        runAsNonRoot: true
        runAsUser: 101
        fsGroup: 101
        seccompProfile:
          type: RuntimeDefault

      containers:
      - name: frontend
        image:
ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/campus-events/frontend:latest
        imagePullPolicy: Always

        ports:
        - containerPort: 3000
          protocol: TCP

        env:
        - name: REACT_APP_API_URL
          valueFrom:
            configMapKeyRef:
              name: frontend-config
```

```yaml
            key: api_url

      resources:
        requests:
          cpu: 500m
          memory: 512Mi
        limits:
          cpu: 1000m
          memory: 1Gi

      securityContext:
        allowPrivilegeEscalation: false
        readOnlyRootFilesystem: true
        runAsNonRoot: true
        runAsUser: 101
        capabilities:
          drop:
          - ALL

      livenessProbe:
        httpGet:
          path: /health
          port: 3000
        initialDelaySeconds: 10
        periodSeconds: 10
        timeoutSeconds: 5
        failureThreshold: 3

      readinessProbe:
        httpGet:
          path: /ready
          port: 3000
        initialDelaySeconds: 5
        periodSeconds: 5
        timeoutSeconds: 3
        failureThreshold: 3

      volumeMounts:
      - name: tmp
        mountPath: /tmp
      - name: cache
        mountPath: /var/cache/nginx
      - name: run
        mountPath: /var/run

    volumes:
    - name: tmp
```

```yaml
        emptyDir: {}
      - name: cache
        emptyDir: {}
      - name: run
        emptyDir: {}

---
# kubernetes/base/frontend/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: frontend
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 3000
    protocol: TCP
    name: http
  selector:
    app: frontend

---
# kubernetes/base/frontend/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: frontend-config
data:
  api_url: "http://events-api.prod.svc.cluster.local:8080"

---
# kubernetes/base/frontend/hpa.yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: frontend-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: frontend
  minReplicas: 3
  maxReplicas: 10
  metrics:
```

```
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
```

Kustomization Files:

None
```
# kubernetes/base/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- namespaces.yaml
- frontend/deployment.yaml
- frontend/service.yaml
- frontend/configmap.yaml
- frontend/hpa.yaml
- events-api/deployment.yaml
- events-api/service.yaml
- events-api/secret.yaml
- events-api/hpa.yaml
- notification-service/deployment.yaml
- notification-service/service.yaml
- notification-service/hpa.yaml
- network-policies.yaml

---
# kubernetes/overlays/dev/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

namespace: dev

bases:
- ../../base
```

```
patchesStrategicMerge:
- replicas.yaml

images:
- name: ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/campus-events/frontend
  newTag: dev-latest
- name: ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/campus-events/events-api
  newTag: dev-latest
```

Deployment Commands:

Shell
```
# Deploy to dev
kubectl apply -k kubernetes/overlays/dev/

# Verify deployments
kubectl get pods -n dev
kubectl get svc -n dev
kubectl get hpa -n dev

# Check pod logs
kubectl logs -n dev -l app=frontend --tail=100
kubectl logs -n dev -l app=events-api --tail=100
```

Day 19: Ingress Configuration

Infrastructure Engineer Tasks

ALB Ingress Configuration:

None
```
# kubernetes/base/ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: campus-events
  namespace: prod
  annotations:
    # ALB configuration
    kubernetes.io/ingress.class: alb
```

```yaml
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb.ingress.kubernetes.io/ssl-redirect: '443'

    # Certificate
    alb.ingress.kubernetes.io/certificate-arn:
arn:aws:acm:us-east-1:ACCOUNT_ID:certificate/CERT_ID

    # Health check
    alb.ingress.kubernetes.io/healthcheck-path: /health
    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'
    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'
    alb.ingress.kubernetes.io/healthy-threshold-count: '2'
    alb.ingress.kubernetes.io/unhealthy-threshold-count: '2'

    # Security
    alb.ingress.kubernetes.io/security-groups: sg-xxxxx
    alb.ingress.kubernetes.io/ssl-policy: ELBSecurityPolicy-TLS-1-2-2017-01

    # Tags
    alb.ingress.kubernetes.io/tags: Environment=prod,Project=campus-events

spec:
  rules:
  - host: events.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: frontend
            port:
              number: 80

      - path: /api
        pathType: Prefix
        backend:
          service:
            name: events-api
            port:
              number: 8080
```

Tasks:

- ☐ Create ACM certificate
- ☐ Configure ALB Ingress
- ☐ Set up Route53 DNS
- ☐ Configure External DNS
- ☐ Test HTTPS access
- ☐ Verify SSL redirect

Day 20: Network Policies

Monitoring & Security Lead Tasks

Network Policy Implementation:

```
None
# kubernetes/base/network-policies.yaml

# Default deny all
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
  namespace: prod
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress

# Allow frontend to events-api
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: frontend-to-api
  namespace: prod
spec:
  podSelector:
    matchLabels:
      app: events-api
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
```

```yaml
        matchLabels:
          app: frontend
    ports:
    - protocol: TCP
      port: 8080

# Allow events-api database access
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: api-database-egress
  namespace: prod
spec:
  podSelector:
    matchLabels:
      app: events-api
  policyTypes:
  - Egress
  egress:
  # Database access
  - to:
    - podSelector: {}
    ports:
    - protocol: TCP
      port: 5432
  # DNS
  - to:
    - namespaceSelector:
        matchLabels:
          name: kube-system
    ports:
    - protocol: UDP
      port: 53

# Allow monitoring
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-prometheus
  namespace: prod
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
```

```
  - from:
    - namespaceSelector:
        matchLabels:
          name: monitoring
    ports:
    - protocol: TCP
      port: 8080
```

Tasks:

- [ ] Implement network policies
- [ ] Test connectivity between services
- [ ] Verify blocked connections
- [ ] Document network architecture
- [ ] Create troubleshooting guide

Day 21: Performance Testing

All Team Members Tasks

Load Testing with k6:

```javascript
// scripts/load-test.js
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
  stages: [
    { duration: '2m', target: 100 },  // Ramp up to 100 users
    { duration: '5m', target: 100 },  // Stay at 100 users
    { duration: '2m', target: 200 },  // Ramp up to 200 users
    { duration: '5m', target: 200 },  // Stay at 200 users
    { duration: '2m', target: 0 },    // Ramp down to 0 users
  ],
};

export default function () {
  // Test event listing
  const listResponse = http.get('https://events.example.com/api/v1/events');
  check(listResponse, {
    'list events status is 200': (r) => r.status === 200,
    'list events response time < 500ms': (r) => r.timings.duration < 500,
  });

  // Test event detail
  const detailResponse =
http.get('https://events.example.com/api/v1/events/1');
  check(detailResponse, {
    'event detail status is 200': (r) => r.status === 200,
    'event detail response time < 300ms': (r) => r.timings.duration < 300,
  });

  sleep(1);
}
```

Run Load Tests:

```shell
# Install k6
brew install k6

# Run load test
k6 run scripts/load-test.js

# Monitor during test
kubectl top pods -n prod
kubectl get hpa -n prod --watch
```

Tasks:

- ☐ Create load test scenarios
- ☐ Run load tests
- ☐ Monitor pod scaling
- ☐ Monitor node scaling (Karpenter)
- ☐ Analyze metrics in Grafana
- ☐ Document performance baseline
- ☐ Optimize resource limits if needed

Deliverables for Week 4

- ✅ Applications deployed to EKS
- ✅ ALB Ingress configured with HTTPS

- ✅ HPA configured for all services
- ✅ Karpenter scaling nodes
- ✅ Network policies implemented
- ✅ Load testing completed
- ✅ Performance metrics documented

Week 5: CI/CD & Observability

Goals

- Complete CI/CD pipeline
- Configure alerting
- Create Grafana dashboards
- Documentation
- Final presentation preparation

Day 22-23: CI/CD Pipeline

Project Lead / DevOps Tasks

Complete GitHub Actions Workflow:

```
None
# .github/workflows/deploy.yml
name: Deploy to EKS

on:
  push:
    branches: [main]
  workflow_dispatch:
    inputs:
      environment:
        description: 'Environment to deploy to'
        required: true
        default: 'dev'
        type: choice
        options:
        - dev
        - staging
        - prod

env:
```

```yaml
  AWS_REGION: us-east-1

jobs:
  build-and-push:
    runs-on: ubuntu-latest
    permissions:
      id-token: write
      contents: read
    outputs:
      image-tag: ${{ steps.image-tag.outputs.tag }}

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: ${{ secrets.AWS_ROLE_ARN }}
          aws-region: ${{ env.AWS_REGION }}

      - name: Login to Amazon ECR
        id: login-ecr
        uses: aws-actions/amazon-ecr-login@v2

      - name: Generate image tag
        id: image-tag
        run: |
          echo "tag=${{ github.sha }}" >> $GITHUB_OUTPUT

      - name: Build, tag, and push frontend
        env:
          ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          ECR_REPOSITORY: campus-events/frontend
          IMAGE_TAG: ${{ steps.image-tag.outputs.tag }}
        run: |
          docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG \
            -t $ECR_REGISTRY/$ECR_REPOSITORY:latest \
            applications/frontend
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:latest

      - name: Build, tag, and push events-api
        env:
          ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          ECR_REPOSITORY: campus-events/events-api
          IMAGE_TAG: ${{ steps.image-tag.outputs.tag }}
        run: |
          docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG \
            -t $ECR_REGISTRY/$ECR_REPOSITORY:latest \
            applications/events-api
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:latest

      - name: Build, tag, and push notification-service
        env:
          ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
          ECR_REPOSITORY: campus-events/notification-service
          IMAGE_TAG: ${{ steps.image-tag.outputs.tag }}
        run: |
          docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG \
            -t $ECR_REGISTRY/$ECR_REPOSITORY:latest \
            applications/notification-service
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
          docker push $ECR_REGISTRY/$ECR_REPOSITORY:latest

      - name: Scan images with Trivy
        uses: aquasecurity/trivy-action@master
        with:
          scan-type: 'image'
          image-ref: '${{ steps.login-ecr.outputs.registry
}}/campus-events/frontend:${{ steps.image-tag.outputs.tag }}'
          format: 'sarif'
          output: 'trivy-results.sarif'

      - name: Upload Trivy results
        uses: github/codeql-action/upload-sarif@v2
        if: always()
        with:
          sarif_file: 'trivy-results.sarif'

  deploy:
    needs: build-and-push
    runs-on: ubuntu-latest
    environment:
      name: ${{ github.event.inputs.environment || 'dev' }}
    permissions:
      id-token: write
      contents: read

    steps:
      - name: Checkout code
        uses: actions/checkout@v4
      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: ${{ secrets.AWS_ROLE_ARN }}
          aws-region: ${{ env.AWS_REGION }}

      - name: Update kubeconfig
        run: |
          aws eks update-kubeconfig \
            --region ${{ env.AWS_REGION }} \
            --name campus-events-${{ github.event.inputs.environment || 'dev'
}}

      - name: Deploy to EKS
        env:
          ENVIRONMENT: ${{ github.event.inputs.environment || 'dev' }}
          IMAGE_TAG: ${{ needs.build-and-push.outputs.image-tag }}
        run: |
          cd kubernetes/overlays/$ENVIRONMENT
          kustomize edit set image \

frontend=ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/campus-events/frontend:$IMA
GE_TAG \

events-api=ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/campus-events/events-api:
$IMAGE_TAG \

notification-service=ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/campus-events/n
otification-service:$IMAGE_TAG

          kubectl apply -k .

      - name: Wait for deployment
        run: |
          kubectl rollout status deployment/frontend -n ${{
github.event.inputs.environment || 'dev' }} --timeout=5m
          kubectl rollout status deployment/events-api -n ${{
github.event.inputs.environment || 'dev' }} --timeout=5m
          kubectl rollout status deployment/notification-service -n ${{
github.event.inputs.environment || 'dev' }} --timeout=5m

      - name: Run smoke tests
        run: |
          bash scripts/smoke-tests.sh ${{ github.event.inputs.environment ||
'dev' }}

      - name: Notify on failure
```

```
        if: failure()
        uses: 8398a7/action-slack@v3
        with:
          status: ${{ job.status }}
          text: 'Deployment to ${{ github.event.inputs.environment || "dev" }}
failed!'
          webhook_url: ${{ secrets.SLACK_WEBHOOK }}
```

Smoke Test Script:

Shell
```
#!/bin/bash
# scripts/smoke-tests.sh

ENVIRONMENT=$1
NAMESPACE=$ENVIRONMENT

echo "Running smoke tests for $ENVIRONMENT environment..."

# Get ALB URL
ALB_URL=$(kubectl get ingress campus-events -n $NAMESPACE \
  -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')

echo "Testing $ALB_URL..."

# Test frontend
echo "Testing frontend..."
FRONTEND_STATUS=$(curl -s -o /dev/null -w "%{http_code}" https://$ALB_URL/)
if [ $FRONTEND_STATUS -eq 200 ]; then
  echo "✓ Frontend is healthy"
else
  echo "✗ Frontend returned $FRONTEND_STATUS"
  exit 1
fi

# Test API health
echo "Testing API..."
API_STATUS=$(curl -s -o /dev/null -w "%{http_code}"
https://$ALB_URL/api/v1/health)
if [ $API_STATUS -eq 200 ]; then
  echo "✓ API is healthy"
else
  echo "✗ API returned $API_STATUS"
  exit 1
```

```
fi

# Test database connectivity
echo "Testing database connectivity..."
DB_STATUS=$(curl -s https://$ALB_URL/api/v1/health/db | jq -r '.status')
if [ "$DB_STATUS" == "healthy" ]; then
  echo "✓ Database is healthy"
else
  echo "✗ Database is unhealthy"
  exit 1
fi

echo "All smoke tests passed! ✓"
```

Tasks:

- [ ] Complete CI/CD workflows
- [ ] Set up GitHub environments
- [ ] Configure required approvals for prod
- [ ] Test automated deployments
- [ ] Test rollback procedures
- [ ] Document CI/CD process

Day 24: Observability Configuration

Monitoring & Security Lead Tasks

Custom Grafana Dashboards:

```json
// dashboards/application-dashboard.json
{
  "dashboard": {
    "title": "Campus Events Application Dashboard",
    "panels": [
      {
        "title": "Request Rate",
        "targets": [
          {
            "expr": "rate(http_requests_total[5m])"
          }
        ]
      },
      {
        "title": "Error Rate",
        "targets": [
          {
            "expr": "rate(http_requests_total{status=~\"5..\"}[5m]) /
rate(http_requests_total[5m])"
          }
        ]
      },
      {
        "title": "Response Time (p95)",
        "targets": [
          {
            "expr": "histogram_quantile(0.95,
rate(http_request_duration_seconds_bucket[5m]))"
          }
        ]
      }
    ]
  }
}
```

Alert Rules:

```
None
# kubernetes/monitoring/prometheus-rules.yaml
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: campus-events-alerts
  namespace: monitoring
spec:
  groups:
  - name: application
    interval: 30s
    rules:
    - alert: HighErrorRate
      expr: |
        (
          rate(http_requests_total{status=~"5.."}[5m])
          /
          rate(http_requests_total[5m])
        ) > 0.05
      for: 5m
      labels:
        severity: critical
      annotations:
        summary: "High error rate in {{ $labels.service }}"
        description: "Error rate is {{ $value | humanizePercentage }}"

    - alert: HighLatency
      expr: |
        histogram_quantile(0.99,
          rate(http_request_duration_seconds_bucket[5m])
        ) > 1
      for: 10m
      labels:
        severity: warning
      annotations:
        summary: "High latency in {{ $labels.service }}"
        description: "P99 latency is {{ $value }}s"

    - alert: PodCrashLooping
      expr: rate(kube_pod_container_status_restarts_total[15m]) > 0
      for: 5m
      labels:
        severity: warning
      annotations:
        summary: "Pod {{ $labels.pod }} is crash looping"
```

Alertmanager Configuration:

```
None
# kubernetes/monitoring/alertmanager-config.yaml
apiVersion: v1
kind: Secret
metadata:
  name: alertmanager-config
  namespace: monitoring
type: Opaque
stringData:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m

    route:
      group_by: ['alertname', 'cluster', 'service']
```

```yaml
    group_wait: 10s
    group_interval: 10s
    repeat_interval: 12h
    receiver: 'slack-notifications'

    routes:
    - match:
        severity: critical
      receiver: 'pagerduty-critical'
      continue: true

    - match:
        severity: warning
      receiver: 'slack-notifications'

  receivers:
  - name: 'slack-notifications'
    slack_configs:
    - api_url: 'SLACK_WEBHOOK_URL'
      channel: '#alerts'
      title: '{{ .GroupLabels.alertname }}'
      text: '{{ range .Alerts }}{{ .Annotations.description }}{{ end }}'

  - name: 'pagerduty-critical'
    pagerduty_configs:
    - service_key: 'PAGERDUTY_SERVICE_KEY'
```

Tasks:

- [ ] Import Grafana dashboards
- [ ] Configure Prometheus alert rules
- [ ] Set up Alertmanager
- [ ] Test alerting (trigger alerts)
- [ ] Configure notification channels
- [ ] Document monitoring setup

Day 25-26: Documentation and Testing

All Team Members Tasks

Final Documentation Checklist:

- [ ] Architecture document (complete)

- [ ] API documentation (OpenAPI spec)
- [ ] Deployment guide
- [ ] Operations runbook
- [ ] Troubleshooting guide
- [ ] Security documentation
- [ ] Cost analysis
- [ ] Disaster recovery plan

README.md Template:

````
None
# Campus Events Management System

## Overview
Brief description of the project

## Architecture
Link to architecture documentation

## Prerequisites
- AWS account with appropriate permissions
- Tools: kubectl, terraform, helm, aws-cli

## Quick Start
\`\`\`bash
# Clone repository
git clone ...

# Set up infrastructure
cd terraform/environments/dev
terraform init
terraform apply

# Deploy applications
kubectl apply -k kubernetes/overlays/dev/
\`\`\`

## Project Structure
Explain directory structure

## CI/CD
Explain CI/CD pipeline

## Monitoring
How to access Grafana, Prometheus
````

```
## Contributing
Link to CONTRIBUTING.md

## Team
List team members and roles

## License
MIT
```

Testing Checklist:

- [ ] All microservices deployed and healthy
- [ ] Frontend accessible via HTTPS
- [ ] API endpoints working
- [ ] RSVP functionality working
- [ ] Notifications being sent
- [ ] Monitoring dashboards showing data
- [ ] Alerts triggering correctly
- [ ] Auto-scaling working
- [ ] Network policies enforced
- [ ] Security scans passing

Day 27: Presentation Preparation

All Team Members Tasks

Presentation Structure:

1. **Introduction** (2 min)

    - Team members
    - Project overview
    - Architecture at a glance

2. **Technical Implementation** (10 min)

    - Infrastructure as Code approach
    - EKS Blueprints usage
    - Microservices architecture
    - CI/CD pipeline

- Security implementation

3. **Live Demo** (10 min)

    - Access application
    - Create an event
    - Show monitoring
    - Trigger auto-scaling
    - Show CI/CD deployment

4. **Challenges and Learnings** (3 min)

    - What went well
    - Challenges faced
    - Key learnings
    - Future improvements

5. **Q&A** (5 min)

Presentation Artifacts:

- ☐ PowerPoint/Google Slides
- ☐ Architecture diagrams
- ☐ Demo script
- ☐ Backup plan if demo fails
- ☐ Video recording of demo
- ☐ GitHub repository link

Deliverables for Week 5

- ✅ Complete CI/CD pipeline
- ✅ All monitoring configured
- ✅ Alerting working
- ✅ Complete documentation
- ✅ Final presentation ready
- ✅ Project repository organized
- ✅ All acceptance criteria met

Daily Standup Structure

Time: 9:00 AM daily (15 minutes)

Format:

- What did you complete yesterday?
- What will you work on today?
- Any blockers or help needed?
- Quick demo of progress (optional)

Tools:

- Slack for async updates
- Zoom/Teams for video standups
- GitHub Projects for tracking

Risk Management

Identified Risks

Risk	**Probability**	**Impact**	**Mitigation**
AWS cost overrun	Medium	High	Use Spot instances, set billing alerts, daily cost review
Team member availability	Medium	Medium	Cross-training, documentation, paired programming
EKS cluster provisioning delays	Low	High	Start infrastructure early, have rollback plan
Integration issues between services	Medium	Medium	Define API contracts early, use mocks, integration tests
Security vulnerabilities in dependencies	Medium	High	Automated scanning, regular updates, security reviews

Risk	Probability	Impact	Mitigation
Time management	High	High	Daily standups, clear milestones, buffer time
Knowledge gaps	Medium	Medium	Pair programming, documentation, office hours with instructor

Contingency Plans

If infrastructure provisioning fails:

- Fall back to eksctl for quick cluster creation
- Use smaller cluster configuration
- Document issues for learning

If services don't communicate:

- Debug with kubectl logs and describe
- Use port-forwarding for direct access
- Check network policies and security groups

If running behind schedule:

- Prioritize core features
- Simplify optional components
- Extend working hours if needed

Success Criteria

Technical Success Criteria

- [ ] EKS cluster running in 3 AZs with at least 2 nodes
- [ ] All 3 microservices deployed and healthy
- [ ] HTTPS access via ALB working
- [ ] Auto-scaling (HPA and Karpenter) functional
- [ ] Monitoring stack showing metrics
- [ ] Alerts firing correctly

- [ ] CI/CD pipeline deploying automatically
- [ ] All security requirements met
- [ ] No critical vulnerabilities in images
- [ ] Database backups configured

Documentation Success Criteria

- [ ] Architecture document complete
- [ ] API documentation (OpenAPI)
- [ ] Deployment guide
- [ ] Runbook for operations
- [ ] All code commented
- [ ] README files in all directories
- [ ] Contribution guidelines

Presentation Success Criteria

- [ ] Live demo working
- [ ] All team members present
- [ ] Architecture clearly explained
- [ ] Security measures demonstrated
- [ ] Questions answered confidently
- [ ] Within time limit (30 minutes)

Post-Project Activities

Week 6 (Optional)

Cleanup:

Shell
```shell
# Destroy infrastructure
cd terraform/environments/dev
terraform destroy -auto-approve

# Delete ECR repositories
aws ecr delete-repository --repository-name campus-events/frontend --force
aws ecr delete-repository --repository-name campus-events/events-api --force
aws ecr delete-repository --repository-name campus-events/notification-service
--force
```

Final Report:

- Cost analysis (actual vs. estimated)
- Performance metrics
- Lessons learned
- Recommendations for improvements
- Team retrospective

Individual Contributions:

- Document individual contributions
- Update personal portfolios
- Write blog posts (optional)
- Share on LinkedIn

Additional Resources

Learning Resources

- AWS EKS Best Practices Guide
- Kubernetes Documentation
- Terraform EKS Blueprints
- Container Security Best Practices

Tools Documentation

- kubectl Cheat Sheet
- Helm Documentation
- Kustomize Documentation
- GitHub Actions Documentation

Support

- Office hours: [Schedule with instructor]
- Slack workspace: [Invite link]
- Emergency contact: [Contact info]

Document Version: 1.0
Last Updated: [Current Date]
Maintained By: Project Team