Deployment Guide

Real-World AR ChatGPT for Farmers

Version: 1.0.0

Last Updated: January 2025

Table of Contents

1. Prerequisites

2. Architecture Overview

- 3. Environment Setup
- 4. Database Configuration
- 5. <u>Application Deployment</u>
- 6. CI/CD Pipeline
- 7. Monitoring Setup
- 8. Security Configuration
- 9. Rollback Procedures
- 10. Troubleshooting

1. Prerequisites

1.1 Required Tools

Tool	Version	Purpose
Node.js	18.x LTS	Backend runtime
Python	3.9+	Data processing services
Docker	20.10+	Containerization
Kubernetes	1.25+	Orchestration
Terraform	1.3+	Infrastructure as Code
Git	2.30+	Version control
AWS CLI	2.9+	Cloud provider interface

1.2 Cloud Resources

AWS Services Required:

- EC2 (Application servers)
- RDS PostgreSQL (Database)
- ElastiCache Redis (Cache)
- S3 (Static assets)
- CloudFront (CDN)
- Route 53 (DNS)
- ALB (Load balancer)
- CloudWatch (Monitoring)
- Secrets Manager (Credentials)

Estimated Costs:

Development: ~\$200/month Staging: ~\$400/month

Production: ~\$1,200/month (1000 users)

1.3 Access Requirements

```
# AWS Credentials

export AWS_ACCESS_KEY_ID=your_access_key

export AWS_SECRET_ACCESS_KEY=your_secret_key

export AWS_DEFAULT_REGION=us-east-1

# NASA API Keys

export NASA_API_KEY=your_nasa_key

export MODIS_API_TOKEN=your_modis_token

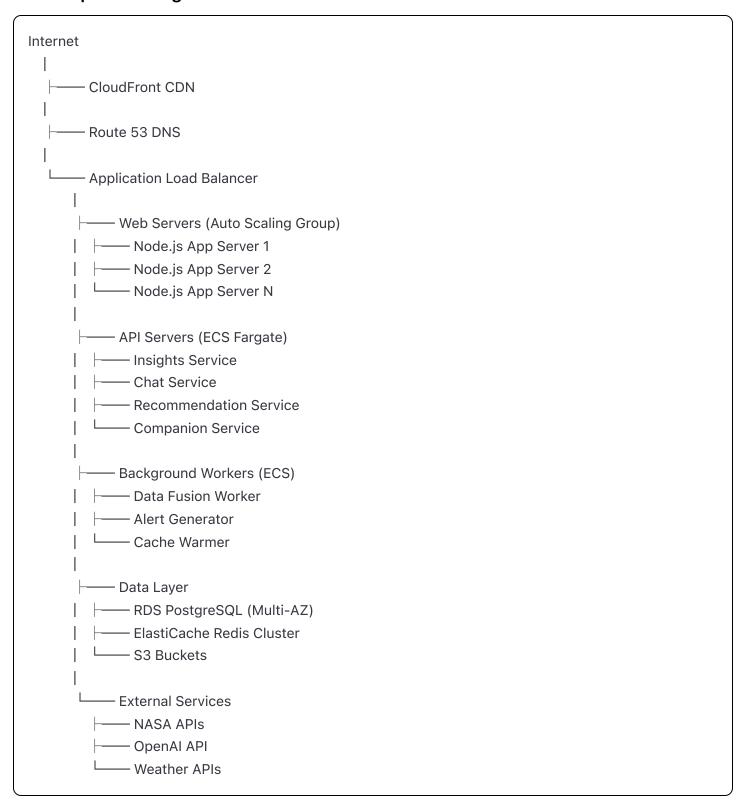
# Third-party Services

export OPENAI_API_KEY=your_openai_key

export GOOGLE_SPEECH_API_KEY=your_google_key
```

2. Architecture Overview

2.1 Component Diagram



2.2 Network Architecture

yaml	

VPC CIDR: 10.0.0.0/16 Subnets: Public: - 10.0.1.0/24 (AZ-1a) - 10.0.2.0/24 (AZ-1b) Private (App): - 10.0.10.0/24 (AZ-1a) - 10.0.11.0/24 (AZ-1b) Private (Data): - 10.0.20.0/24 (AZ-1a) - 10.0.21.0/24 (AZ-1b) **Security Groups:** - sg-alb: 80, 443 from 0.0.0.0/0 - sg-app: 3000 from sg-alb - sg-db: 5432 from sg-app - sg-redis: 6379 from sg-app

3. Environment Setup

3.1 Local Development

ash			

```
# Clone repository
git clone https://github.com/farmnavigator/app.git
cd app
# Install dependencies
npm install
cd services/python && pip install -r requirements.txt
# Setup environment file
cp .env.example .env.local
# Edit .env.local with your credentials
# Start Docker services
docker-compose -f docker-compose.dev.yml up -d
# Run migrations
npm run migrate:dev
# Seed database
npm run seed:dev
# Start development server
npm run dev
```

3.2 Infrastructure Provisioning

Using Terraform:

```
cd infrastructure/terraform

# Initialize Terraform
terraform init

# Plan infrastructure
terraform plan -var-file="environments/production.tfvars"

# Apply infrastructure
terraform apply -var-file="environments/production.tfvars"
```

Terraform Configuration (main.tf):

```
hcl
provider "aws" {
 region = var.aws_region
}
module "vpc" {
 source = "./modules/vpc"
 cidr_block = "10.0.0.0/16"
 availability_zones = ["us-east-1a", "us-east-1b"]
}
module "rds" {
 source = "./modules/rds"
 vpc_id = module.vpc.vpc_id
 subnet_ids = module.vpc.private_subnet_ids
 instance_class = "db.t3.medium"
 allocated_storage = 100
 multi_az = true
}
module "ecs" {
 source = "./modules/ecs"
 vpc_id = module.vpc.vpc_id
 subnet_ids = module.vpc.private_subnet_ids
 cluster_name = "farmnavigator-cluster"
}
module "elasticache" {
 source = "./modules/elasticache"
 vpc_id = module.vpc.vpc_id
 subnet_ids = module.vpc.private_subnet_ids
 node_type = "cache.t3.micro"
 num_nodes = 2
```

3.3 Environment Variables

Production Environment (.env.production):

bash

```
# Application
NODE_ENV=production
PORT=3000
API_URL=https://api.farmnavigator.app
# Database
DATABASE_URL=postgresql://user:pass@rds-endpoint:5432/farmnavigator
DATABASE_POOL_SIZE=20
# Redis
REDIS_URL=redis://elasticache-endpoint:6379
REDIS_TTL=1800
# NASA APIs
NASA_API_KEY=your_production_key
NASA_API_RATE_LIMIT=100
SMAP_API_URL=https://n5eil01u.ecs.nsidc.org
MODIS_API_URL=https://modis.ornl.gov/rst/api/v1
# AI Services
OPENAI_API_KEY=your_production_key
OPENAI_MODEL=gpt-4
RAG_INDEX_NAME=farmnavigator-prod
# Security
JWT_SECRET=your_jwt_secret
JWT_EXPIRY=86400
CORS_ORIGINS=https://farmnavigator.app
# Monitoring
DATADOG_API_KEY=your_datadog_key
SENTRY_DSN=https://xxx@sentry.io/xxx
# Feature Flags
ENABLE_VOICE=true
ENABLE_COMPANION=true
ENABLE_OFFLINE_MODE=true
```

4. Database Configuration

4.1 PostgreSQL Setup

```
sql
-- Create database
CREATE DATABASE farmnavigator;
-- Create user
CREATE USER farmapp WITH ENCRYPTED PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE farmnavigator TO farmapp;
-- Enable extensions
\( c farmnavigator;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "postgis";
CREATE EXTENSION IF NOT EXISTS "postgis";
-- Create schemas
CREATE SCHEMA IF NOT EXISTS app;
CREATE SCHEMA IF NOT EXISTS analytics;
CREATE SCHEMA IF NOT EXISTS cache;
```

4.2 Database Migrations

```
bash

# Run migrations

npm run migrate:up

# Rollback migration

npm run migrate:down

# Create new migration

npm run migrate:create add_user_preferences
```

Migration Example (001_initial_schema.sql):

1		
	sql	
l		

```
-- Up Migration
CREATE TABLE app.users (
 id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  email VARCHAR(255) UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE app.locations (
 id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES app.users(id),
 name VARCHAR(255) NOT NULL,
  coordinates GEOGRAPHY(POINT, 4326),
 area_m2 INTEGER,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE INDEX idx_locations_user ON app.locations(user_id);
CREATE INDEX idx_locations_coords ON app.locations USING GIST(coordinates);
-- Down Migration
DROP TABLE IF EXISTS app.locations;
DROP TABLE IF EXISTS app.users;
```

4.3 Redis Configuration

```
# Redis configuration (redis.conf)
maxmemory 2gb
maxmemory-policy allkeys-Iru
save 900 1
save 300 10
save 60 10000
appendonly yes
```

5. Application Deployment

5.1 Docker Configuration

Dockerfile (Node.js Service):

```
dockerfile
# Multi-stage build
FROM node:18-alpine AS builder
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY..
RUN npm run build
# Production stage
FROM node:18-alpine
RUN apk add --no-cache tini
WORKDIR /app
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/package.json ./
EXPOSE 3000
USER node
ENTRYPOINT ["/sbin/tini", "--"]
CMD ["node", "dist/server.js"]
```

Dockerfile (Python Service):

dockerfile

```
FROM python:3.9-slim

WORKDIR /app

# Install system dependencies

RUN apt-get update && apt-get install -y \
gdal-bin \
libgdal-dev \
&& rm -rf /var/lib/apt/lists/*

# Install Python dependencies

COPY requirements.txt .

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

COPY . .

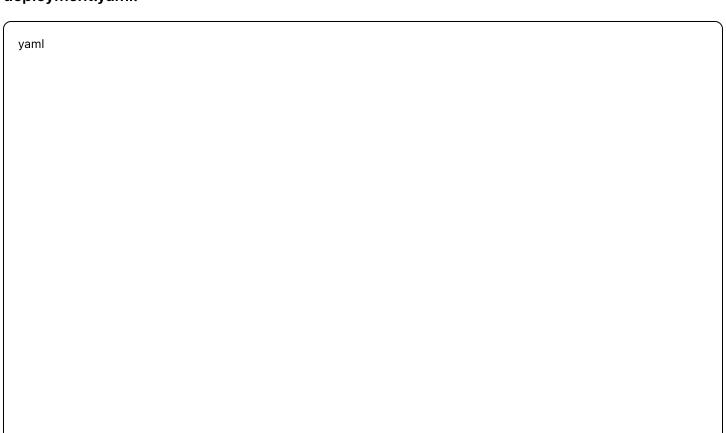
EXPOSE 5000

USER nobody

CMD ["gunicorn", "--bind", "0.0.0.0:5000", "--workers", "4", "app:application"]
```

5.2 Kubernetes Deployment

deployment.yaml:



```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: farmnavigator-api
 namespace: production
spec:
 replicas: 3
 selector:
  matchLabels:
   app: farmnavigator-api
 template:
  metadata:
   labels:
    app: farmnavigator-api
  spec:
   containers:
   - name: api
    image: farmnavigator/api:latest
    ports:
    - containerPort: 3000
    env:
    - name: NODE_ENV
     value: production
    - name: DATABASE_URL
     valueFrom:
      secretKeyRef:
       name: db-credentials
       key: url
    resources:
     requests:
      memory: "256Mi"
      cpu: "250m"
     limits:
      memory: "512Mi"
      cpu: "500m"
    livenessProbe:
     httpGet:
      path: /health
      port: 3000
     initialDelaySeconds: 30
     periodSeconds: 10
    readinessProbe:
     httpGet:
```

```
path: /ready
      port: 3000
     initialDelaySeconds: 5
     periodSeconds: 5
apiVersion: v1
kind: Service
metadata:
 name: farmnavigator-api
spec:
 selector:
  app: farmnavigator-api
 ports:
 - port: 80
 targetPort: 3000
 type: LoadBalancer
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
 name: farmnavigator-api-hpa
spec:
 scaleTargetRef:
  apiVersion: apps/v1
  kind: Deployment
  name: farmnavigator-api
 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
```

5.3 Deployment Steps

```
bash
# 1. Build and push Docker images
docker build -t farmnavigator/api:v1.0.0.
docker tag farmnavigator/api:v1.0.0 farmnavigator/api:latest
docker push farmnavigator/api:v1.0.0
docker push farmnavigator/api:latest
# 2. Deploy to Kubernetes
kubectl apply -f k8s/namespace.yaml
kubectl apply -f k8s/secrets.yaml
kubectl apply -f k8s/configmap.yaml
kubectl apply -f k8s/deployment.yaml
kubectl apply -f k8s/service.yaml
kubectl apply -f k8s/ingress.yaml
# 3. Verify deployment
kubectl get pods -n production
kubectl get svc -n production
kubectl logs -f deployment/farmnavigator-api -n production
# 4. Run smoke tests
npm run test:smoke -- --env=production
# 5. Update DNS
aws route53 change-resource-record-sets \
 --hosted-zone-id Z123456789 \
 --change-batch file://dns-update.json
```

6. CI/CD Pipeline

6.1 GitHub Actions Workflow

.github/workflows/deploy.yml:

yaml

```
name: Deploy to Production
on:
 push:
  branches: [main]
 workflow_dispatch:
env:
 AWS_REGION: us-east-1
 ECR_REPOSITORY: farmnavigator
 ECS_CLUSTER: farmnavigator-cluster
 ECS_SERVICE: farmnavigator-service
jobs:
 test:
  runs-on: ubuntu-latest
  steps:
  - uses: actions/checkout@v3
  - name: Setup Node.js
   uses: actions/setup-node@v3
   with:
    node-version: '18'
    cache: 'npm'
  - name: Install dependencies
   run: npm ci
  - name: Run tests
   run: |
    npm run test
    npm run test:integration
  - name: SonarQube Scan
   uses: sonarsource/sonarqube-scan-action@master
   env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
 build:
  needs: test
  runs-on: ubuntu-latest
  steps:
```

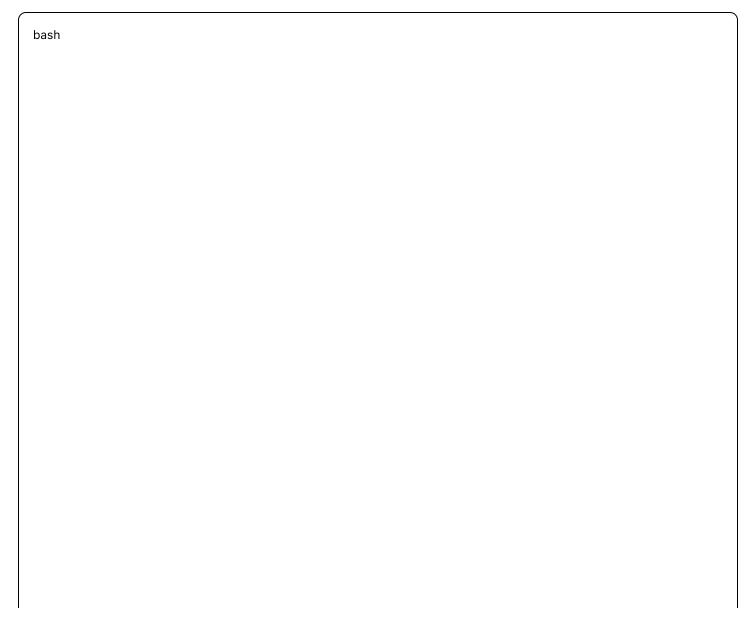
```
- uses: actions/checkout@v3
 - name: Configure AWS credentials
 uses: aws-actions/configure-aws-credentials@v2
 with:
   aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
   aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
   aws-region: ${{ env.AWS_REGION }}
 - name: Login to Amazon ECR
 id: login-ecr
 uses: aws-actions/amazon-ecr-login@v1
 - name: Build and push Docker image
   ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
   IMAGE_TAG: ${{ github.sha }}
 run:
   docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG.
   docker tag $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG $ECR_REGISTRY/$ECR_REPOSITORY:lat
   docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
   docker push $ECR_REGISTRY/$ECR_REPOSITORY:latest
deploy:
needs: build
runs-on: ubuntu-latest
steps:
uses: actions/checkout@v3
- name: Configure AWS credentials
 uses: aws-actions/configure-aws-credentials@v2
 with:
   aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
   aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
   aws-region: ${{ env.AWS_REGION }}
- name: Deploy to ECS
 run: |
   aws ecs update-service \
    --cluster ${{ env.ECS_CLUSTER }} \
    --service ${{ env.ECS_SERVICE }} \
    --force-new-deployment
- name: Wait for deployment
```

```
run: |
aws ecs wait services-stable \
--cluster ${{ env.ECS_CLUSTER }} \
--services ${{ env.ECS_SERVICE }}

- name: Run smoke tests
run: |
curl -f https://api.farmnavigator.app/health || exit 1

- name: Notify Slack
uses: 8398a7/action-slack@v3
with:
status: ${{ job.status }}
text: 'Deployment to production completed'
webhook_url: ${{ secrets.SLACK_WEBHOOK }}
```

6.2 Blue-Green Deployment



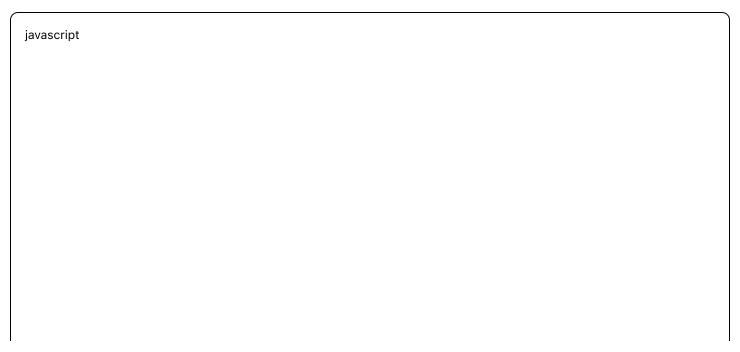
```
#!/bin/bash
# blue-green-deploy.sh
# Variables
BLUE_ENV="production-blue"
GREEN_ENV="production-green"
CURRENT_ENV=$(aws elasticbeanstalk describe-environments --environment-names $BLUE_ENV --query
# Determine target environment
if [[ $CURRENT_ENV == *"blue"* ]]; then
  TARGET_ENV=$GREEN_ENV
  OLD_ENV=$BLUE_ENV
else
  TARGET_ENV=$BLUE_ENV
  OLD_ENV=$GREEN_ENV
fi
echo "Deploying to $TARGET_ENV..."
# Deploy to target environment
eb deploy $TARGET_ENV
# Health check
for i in {1..30}; do
  HEALTH=$(aws elasticbeanstalk describe-environments --environment-names $TARGET_ENV --query 'En
  if [[ $HEALTH == "Green" ]]; then
    echo "Deployment successful"
    break
  fi
  echo "Waiting for environment to be healthy... ($i/30)"
  sleep 10
done
# Swap CNAMEs
echo "Swapping URLs..."
aws elasticbeanstalk swap-environment-cnames \
  --source-environment-name $OLD_ENV \
  --destination-environment-name $TARGET_ENV
echo "Blue-green deployment completed"
```

7. Monitoring Setup

7.1 CloudWatch Configuration

```
yaml
# cloudwatch-dashboard.json
 "name": "FarmNavigator-Production",
 "widgets": [
   "type": "metric",
   "properties": {
    "metrics": [
     ["AWS/ECS", "CPUUtilization", {"stat": "Average"}],
     ["AWS/ECS", "MemoryUtilization", {"stat": "Average"}],
     ["AWS/ApplicationELB", "TargetResponseTime", {"stat": "p99"}],
     ["AWS/ApplicationELB", "HTTPCode_Target_5XX_Count", {"stat": "Sum"}]
    ],
    "period": 300,
    "stat": "Average",
    "region": "us-east-1",
    "title": "Application Metrics"
   }
  }
 ]
}
```

7.2 Logging Configuration



```
// logger.js
const winston = require('winston');
const { CloudWatchTransport } = require('winston-cloudwatch');
const logger = winston.createLogger({
 level: process.env.LOG_LEVEL || 'info',
 format: winston.format.json(),
 defaultMeta: {
  service: 'farmnavigator',
  environment: process.env.NODE_ENV
 },
 transports: [
  new winston.transports.Console({
   format: winston.format.simple()
  }),
  new CloudWatchTransport({
   logGroupName: '/aws/ecs/farmnavigator',
   logStreamName: `${process.env.NODE_ENV}-${new Date().tolSOString().split('T')[0]}`,
   awsRegion: process.env.AWS_REGION
  })
 ]
});
module.exports = logger;
```

7.3 Alerts Configuration



```
# alerts.yaml
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
 name: farmnavigator-alerts
spec:
 groups:
 - name: application
  rules:
  - alert: HighErrorRate
   expr: rate(http_requests_total{status=~"5.."}[5m]) > 0.05
   for: 5m
   annotations:
    summary: "High error rate detected"
    description: "Error rate is {{ $value }} errors per second"
  - alert: HighLatency
   expr: histogram_quantile(0.95, http_request_duration_seconds_bucket) > 2
   for: 10m
   annotations:
    summary: "High latency detected"
    description: "95th percentile latency is {{ $value }} seconds"
  - alert: LowDiskSpace
   expr: node_filesystem_avail_bytes / node_filesystem_size_bytes < 0.1
   for: 5m
   annotations:
    summary: "Low disk space"
    description: "Less than 10% disk space remaining"
```

8. Security Configuration

8.1 SSL/TLS Setup

```
# Generate SSL certificate using Let's Encrypt
sudo certbot certonly --nginx -d farmnavigator.app -d api.farmnavigator.app

# Auto-renewal cron job
echo "0 0,12 * * * python -c 'import random; import time; time.sleep(random.random() * 3600)' && certbot re
```

8.2 Security Headers

```
javascript
// security.js
const helmet = require('helmet');
app.use(helmet({
 contentSecurityPolicy: {
  directives: {
   defaultSrc: ["'self'"],
   styleSrc: ["'self'", "'unsafe-inline'"],
   scriptSrc: ["'self'", "'unsafe-inline'"],
   imgSrc: ["'self'", "data:", "https:"],
   connectSrc: ["'self'", "https://api.farmnavigator.app"]
  }
 },
 hsts: {
  maxAge: 31536000,
  includeSubDomains: true,
  preload: true
 }
}));
```

8.3 Secrets Management

```
# Store secrets in AWS Secrets Manager
aws secretsmanager create-secret \
--name farmnavigator/production/database \
--secret-string '{"username":"dbuser","password":"dbpass","engine":"postgres","host":"rds.amazonaws.co
# Retrieve secrets in application
const AWS = require('aws-sdk');
const secretsManager = new AWS.SecretsManager();

async function getSecret(secretName) {
    const data = await secretsManager.getSecretValue({ SecretId: secretName }).promise();
    return JSON.parse(data.SecretString);
}
```

9. Rollback Procedures

9.1 Database Rollback

```
#!/bin/bash
# db-rollback.sh

# Take snapshot before deployment
aws rds create-db-snapshot \
--db-instance-identifier farmnavigator-prod \
--db-snapshot-identifier farmnavigator-prod-$(date +%Y%m%d-%H%M%S)

# Rollback to previous snapshot
aws rds restore-db-instance-from-db-snapshot \
--db-instance-identifier farmnavigator-prod-rollback \
--db-snapshot-identifier farmnavigator-prod-20250115-120000
```

9.2 Application Rollback

```
# Kubernetes rollback
kubectl rollout undo deployment/farmnavigator-api -n production
kubectl rollout status deployment/farmnavigator-api -n production

# Docker rollback
docker pull farmnavigator/api:v1.0.0-previous
docker tag farmnavigator/api:v1.0.0-previous farmnavigator/api:latest
docker push farmnavigator/api:latest

# ECS rollback
aws ecs update-service \
--cluster farmnavigator-cluster \
--service farmnavigator-service \
--task-definition farmnavigator:previous-revision
```

9.3 Emergency Procedures

bash

Enable maintenance mode

kubectl apply -f k8s/maintenance-mode.yaml

Scale down to minimum

kubectl scale deployment farmnavigator-api --replicas=1

Clear cache

redis-cli -h redis.farmnavigator.app FLUSHALL

Restart services

kubectl rollout restart deployment/farmnavigator-api

Disable problematic features

kubectl set env deployment/farmnavigator-api ENABLE_CHAT=false

10. Troubleshooting

10.1 Common Issues

Issue	Symptoms	Solution	
High Memory Usage	OOM kills, slow	Increase memory limits, check for leaks	
High Memory Osage	responses	increase memory innits, check for leaks	
Database Connection Pool	Timeout errors	Increase pool size, check for connection	
Exhausted	Timeout errors	leaks	
Redis Cache Misses	Slow API responses	Check TTL settings, warm cache	
NASA API Rate Limit	429 errors	Implement better caching, request	
INASA AFI Rate LIIIII	429 611015	batching	
WebAR Not Working	Camera not detecting	Check browser compatibility, permissions	

10.2 Debugging Commands

bash			

```
# Check pod logs
kubectl logs -f pod/farmnavigator-api-xxx -n production

# Shell into container
kubectl exec -it pod/farmnavigator-api-xxx -n production -- /bin/sh

# Database connections
psql -h rds.amazonaws.com -U farmapp -d farmnavigator -c "SELECT count(*) FROM pg_stat_activity;"

# Redis status
redis-cli -h redis.farmnavigator.app INFO stats

# API health check
curl -v https://api.farmnavigator.app/health

# Load testing
artillery run load-test.yml --target https://api.farmnavigator.app
```

10.3 Performance Tuning

javascript			

```
// Connection pooling
const pool = new Pool({
 connectionString: process.env.DATABASE_URL,
 max: 20,
 idleTimeoutMillis: 30000,
 connectionTimeoutMillis: 2000,
});
// Redis caching strategy
const cacheKey = `insights:${lat}:${lon}:${Math.floor(Date.now() / 1800000)}`;
const cached = await redis.get(cacheKey);
if (cached) return JSON.parse(cached);
// Batch processing
const batchProcess = async (items, batchSize = 10) => {
 const results = [];
 for (let i = 0; i < items.length; i += batchSize) {
  const batch = items.slice(i, i + batchSize);
  const batchResults = await Promise.all(batch.map(processItem));
  results.push(...batchResults);
 }
 return results;
};
```

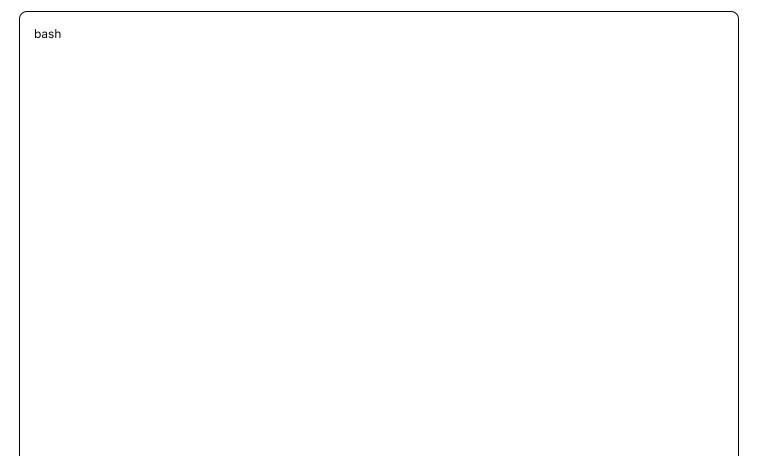
11. Backup and Recovery

11.1 Backup Strategy



```
# backup-cronjob.yaml
apiVersion: batch/v1
kind: CronJob
metadata:
name: database-backup
schedule: "0 2 * * * "
jobTemplate:
  spec:
  template:
    spec:
     containers:
     - name: postgres-backup
      image: postgres:14
      command:
      - /bin/bash
      - -C
      - |
       DATE=$(date +%Y%m%d-%H%M%S)
       pg_dump $DATABASE_URL | gzip > backup-$DATE.sql.gz
       aws s3 cp backup-$DATE.sql.gz s3://farmnavigator-backups/
```

11.2 Disaster Recovery Plan



```
# Full recovery procedure
#!/bin/bash
# 1. Provision new infrastructure
terraform apply -var-file=disaster-recovery.tfvars
# 2. Restore database
aws s3 cp s3://farmnavigator-backups/latest.sql.gz .
gunzip latest.sql.gz
psql $NEW_DATABASE_URL < latest.sql
# 3. Update DNS
aws route53 change-resource-record-sets \
 --hosted-zone-id Z123456789 \
 --change-batch file://dr-dns-update.json
# 4. Deploy application
kubectl apply -f k8s/ -n disaster-recovery
# 5. Verify services
./smoke-tests.sh --env=disaster-recovery
```

12. Post-Deployment Checklist

All pods running and healthy
Database migrations completed
Cache warmed with critical data
SSL certificates valid
■ DNS propagated
☐ Monitoring dashboards active
Alerts configured
☐ Smoke tests passed
Performance benchmarks met
Security scan completed
\square Documentation updated
□ Team notified
☐ Backup verified
Rollback plan ready
Customor communication cont

Appendices

Appendix A: Environment Variables Reference

Complete list of all environment variables

Appendix B: Infrastructure Costs

Detailed breakdown of AWS costs

Appendix C: Scaling Guidelines

When and how to scale each component

Appendix D: Compliance Checklist

Security and regulatory compliance items