Low-Cost Robust CI/CD Architecture for 7-8 Microservices

Executive Summary

This document outlines a cost-effective, secure CI/CD pipeline architecture for 7-8 microservices using GitHub Actions, AWS EC2, and managed services. The solution prioritizes minimal cost while maintaining enterprise-grade security, reliability, and operational excellence.

Architecture Overview

Infrastructure Components

Compute: - 2x EC2 instances (t3.small): UAT + Production ($30/month total) - Self-hosted GitHub runner on UAT instance (cost optimization)

Networking: - Application Load Balancer (ALB) with path-based routing ($16/month) - API Gateway for external traffic management ($3.50/million requests)

Storage & Database: - RDS PostgreSQL t3.micro Multi-AZ for shared config ($25/month) - ECR for container images ($0.10/GB/month) - S3 for artifacts and logs ($5/month)

Security & Monitoring: - AWS Systems Manager Parameter Store (free tier) - CloudWatch Logs and Metrics ($10/month) - AWS Secrets Manager for sensitive data ($0.40/secret/month)

Estimated Monthly Cost: $90-120

Detailed Architecture

1. Repository Structure

```
microservice-{name}/
├── .github/workflows/
│   ├── ci.yml                # Build, test, security scans
│   ├── deploy-uat.yml        # UAT deployment
│   └── deploy-prod.yml       # Production deployment
├── Dockerfile
├── docker-compose.yml
├── terraform/                # Infrastructure as Code
├── k8s/ or docker/          # Container orchestration
└── config/
    ├── uat.env
```

```
    └── prod.env
```

2. CI/CD Pipeline Flow

Phase 1: Continuous Integration 1. Code push triggers GitHub Actions 2. Run parallel jobs: - Unit tests (Jest/PyTest) - Code quality (ESLint/Pylint, SonarCloud) - Security scans (Snyk, Trivy, GitLeaks) - IaC validation (Terraform plan) 3. Build Docker image 4. Push to ECR with semantic versioning 5. Update deployment manifests

Phase 2: UAT Deployment 1. Manual trigger via deployment UI 2. RBAC validation (GitHub Teams integration) 3. Deploy to UAT EC2 via self-hosted runner 4. Health checks and smoke tests 5. Notification to stakeholders

Phase 3: Production Gate 1. UAT sign-off required (manual approval) 2. Production deployment with blue-green strategy 3. Automated rollback on failure 4. Post-deployment verification

3. Security & RBAC Implementation

GitHub OIDC Integration:

```yaml
permissions:
  id-token: write
  contents: read

- name: Configure AWS credentials
  uses: aws-actions/configure-aws-credentials@v2
  with:
    role-to-assume: arn:aws:iam::ACCOUNT:role/GitHubActionsRole
    aws-region: us-east-1
```

Role-Based Access Control: - **Developers**: Can trigger UAT deployments for their services - **QA Team**: Can approve UAT sign-offs - **DevOps/SRE**: Can trigger production deployments - **Managers**: Read-only access to deployment status

Security Measures: - No long-lived AWS keys - Encrypted secrets in AWS Secrets Manager - Container image vulnerability scanning - Network segmentation with security groups - WAF protection for public endpoints

4. Deployment UI Architecture

Technology Stack: - Frontend: React.js with AWS Amplify hosting - Backend: Node.js API on UAT EC2 - Authentication: AWS Cognito with GitHub OAuth - Database: RDS PostgreSQL for deployment metadata

Features: - Service selection dropdown - Real-time deployment logs - Approval workflow management - Deployment history and rollback - Role-based UI components

5. Container Orchestration

Docker Compose Setup per EC2:

```yaml
version: '3.8'
services:
  nginx:
    image: nginx:alpine
    ports: ["80:80", "443:443"]
    volumes: ["./nginx.conf:/etc/nginx/nginx.conf"]

  microservice-1:
    image: ${ECR_URI}/microservice-1:${VERSION}
    environment: ["NODE_ENV=production"]

  microservice-2:
    image: ${ECR_URI}/microservice-2:${VERSION}
    environment: ["NODE_ENV=production"]
```

Nginx Configuration: - Reverse proxy with health checks - SSL termination - Rate limiting - Request routing based on path

6. Configuration Management

Environment-Specific Configs: - AWS Systems Manager Parameter Store for non-sensitive config - AWS Secrets Manager for database credentials, API keys - Environment variables injected at container runtime

Database Configuration: - Shared RDS instance with separate databases per service - Connection pooling and read replicas for performance - Automated backups and point-in-time recovery

7. Monitoring & Observability

Logging Strategy: - Application logs → CloudWatch Logs - Nginx access logs → CloudWatch Logs - Deployment logs → S3 + CloudWatch - Log retention: 30 days (cost optimization)

Metrics & Alerting: - CloudWatch custom metrics for business KPIs - ALB target group health monitoring - EC2 instance health checks - SNS notifications for critical alerts

Cost Monitoring: - AWS Cost Explorer integration - Budget alerts at $150/month threshold - Resource utilization dashboards

8. Disaster Recovery & Business Continuity

Backup Strategy: - RDS automated backups (7-day retention) - ECR image replication to secondary region - Infrastructure as Code in Git (version controlled) - Configuration backup to S3

Recovery Procedures: - Automated EC2 instance replacement via Launch Templates - Database point-in-time recovery - Container rollback to previous stable version - Cross-region failover capability

Implementation Phases

Phase 1: Foundation (Week 1-2)

- AWS account setup and IAM roles
- EC2 instances with Docker
- RDS database setup
- Basic CI/CD pipeline for 1 service

Phase 2: Core Pipeline (Week 3-4)

- GitHub Actions workflows
- Security scanning integration
- ECR setup and image management
- Nginx reverse proxy configuration

Phase 3: Deployment UI (Week 5-6)

- React deployment interface
- RBAC implementation
- Approval workflow system
- Real-time log streaming

Phase 4: Production Hardening (Week 7-8)

- Blue-green deployment strategy
- Comprehensive monitoring
- Load testing and performance optimization
- Documentation and runbooks

Cost Optimization Strategies

1. **Right-sizing**: Start with t3.small, scale based on metrics
2. **Reserved Instances**: 1-year term for 40% savings after validation
3. **Spot Instances**: For non-critical workloads and testing
4. **S3 Lifecycle**: Move old logs to IA/Glacier
5. **CloudWatch**: Custom retention policies per log group
6. **ECR**: Lifecycle policies to remove old images

Security Hardening Recommendations

1. **Network Security:**
    - VPC with private subnets
    - NAT Gateway for outbound traffic
    - Security groups with least privilege
    - WAF with OWASP Top 10 rules
2. **Application Security:**

    - Container image scanning (Trivy/Snyk)
    - Dependency vulnerability checks
    - Secret scanning in repositories
    - SAST/DAST integration
3. **Operational Security:**
    - AWS Config for compliance monitoring
    - CloudTrail for audit logging
    - GuardDuty for threat detection
    - Regular security assessments

Risk Mitigation

Single Points of Failure: - Multi-AZ RDS deployment - ALB with multiple targets - Automated instance replacement - Cross-region backup strategy

Performance Risks: - Auto-scaling groups for EC2 - CloudWatch alarms for proactive scaling - Load testing in UAT environment - Performance budgets in CI/CD

Security Risks: - Regular security updates via automation - Vulnerability scanning in pipeline - Incident response procedures - Security training for team

Success Metrics

Operational: - Deployment frequency: Daily - Lead time: < 2 hours - MTTR: < 30 minutes - Change failure rate: < 5%

Cost: - Monthly AWS bill: < $120 - Cost per deployment: < $2 - Resource utilization: > 70%

Security: - Zero critical vulnerabilities in production - 100% secret scanning coverage - Security incident response: < 15 minutes

Next Steps

1. **Review and Approval**: Stakeholder review of this approach
2. **Environment Setup**: AWS account and initial infrastructure
3. **Pilot Implementation**: Start with 1-2 microservices
4. **Iterative Rollout**: Add remaining services progressively
5. **Optimization**: Monitor and optimize based on usage patterns

---

Document Version: 1.0
Date: September 19, 2025
Author: CI/CD Architecture Team
Review Required: Yes