Contents

Deployment Runbook

Overview

This runbook covers deployment procedures for the RADIANT platform.

Environments

Environment	Purpose	Auto-Deploy	Approval
`dev`	Development testing	On PR merge to develop	None
`staging`	Pre-production	On PR merge to main	None
`production`	Live environment	Manual trigger	Required

Pre-Deployment Checklist

Code Quality

- ☐ All tests passing in CI
- ☐ Code review approved
- ☐ No security vulnerabilities
- ☐ Documentation updated

Infrastructure

- ☐ CDK diff reviewed
- ☐ Database migrations reviewed
- ☐ No breaking API changes (or versioned)
- ☐ Feature flags in place if needed

Operations

- ☐ Monitoring dashboards ready
- ☐ Rollback plan documented
- ☐ On-call notified (production)
- ☐ Low-traffic window selected (production)

Deployment Steps

1. Development Environment

```
# Automatic via GitHub Actions on develop branch
# Or manual:
make deploy-dev
```

2. Staging Environment

```
# Automatic via GitHub Actions on main branch
# Or manual:
make deploy-staging
```

3. Production Environment

Step 1: Create Release

```
# Create release tag
git tag -a v4.17.1 -m "Release 4.17.1"
git push origin v4.17.1

# Or use GitHub Releases UI
```

Step 2: Deploy Infrastructure

```
# CDK diff first
ENVIRONMENT=production pnpm --filter @radiant/infrastructure cdk diff
```

```
# Deploy with approval
make deploy-prod
```

Step 3: Database Migrations

1. Go to Admin Dashboard → Migrations
2. Create migration approval request
3. Get second admin approval
4. Execute migration
5. Verify data integrity

Step 4: Deploy Dashboard

```
# Build dashboard
pnpm --filter @radiant/admin-dashboard build

# Deploy to S3
aws s3 sync apps/admin-dashboard/out s3://radiant-dashboard-production --delete

# Invalidate CloudFront
aws cloudfront create-invalidation \
  --distribution-id <DIST_ID> \
  --paths "/*"
```

Step 5: Verify Deployment

```
# Health check
curl https://api.radiant.example.com/v2/health

# Check dashboard
open https://admin.radiant.example.com

# Monitor CloudWatch
open https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#dashboards:name=radiant-pr
```

Database Migrations

Creating Migrations

```
# Create new migration file
touch packages/infrastructure/migrations/037_new_feature.sql
```

Migration File Format

```
-- Migration: 037_new_feature
-- Description: Add new feature tables
-- Author: developer@example.com
-- Date: 2024-12-24

-- Up Migration
```

```sql
CREATE TABLE IF NOT EXISTS new_feature (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Enable RLS
ALTER TABLE new_feature ENABLE ROW LEVEL SECURITY;

-- RLS Policy
CREATE POLICY new_feature_tenant_isolation ON new_feature
  USING (tenant_id = current_setting('app.current_tenant_id')::uuid);

-- Rollback: DROP TABLE new_feature;
```

Running Migrations

```bash
# Dev/Staging (local)
./scripts/run-migrations.sh

# Production (via Admin Dashboard)
# Use Migration Approval workflow
```

Rollback Procedures

Quick Rollback (Lambda)

```bash
# List recent versions
aws lambda list-versions-by-function \
  --function-name radiant-production-router \
  --max-items 5

# Update alias to previous version
aws lambda update-alias \
  --function-name radiant-production-router \
  --name live \
  --function-version 42
```

Full Rollback (CDK)

```bash
# Check recent deployments
aws cloudformation describe-stack-events \
  --stack-name RadiantProductionApi \
  --max-items 20

# Rollback (if still in progress)
aws cloudformation cancel-update-stack \
  --stack-name RadiantProductionApi
```

```
# Or redeploy previous version
git checkout v4.17.0
make deploy-prod
```

Dashboard Rollback

```
# Sync previous build from backup
aws s3 sync s3://radiant-backups/dashboard/v4.17.0 s3://radiant-dashboard-production --delete

# Invalidate cache
aws cloudfront create-invalidation \
  --distribution-id <DIST_ID> \
  --paths "/*"
```

Blue-Green Deployment (Optional)

For zero-downtime deployments:

1. Deploy new version to "green" stack
2. Run smoke tests against green
3. Switch Route 53 weighted routing
4. Monitor for errors
5. Remove "blue" stack after verification

Canary Deployment (Optional)

For gradual rollouts:

1. Deploy new Lambda version
2. Configure alias with 10% weight
3. Monitor error rates
4. Gradually increase to 100%

```
# Configure canary
aws lambda update-alias \
  --function-name radiant-production-router \
  --name live \
  --routing-config AdditionalVersionWeights={"43"=0.1}
```

Monitoring During Deployment

Key Metrics to Watch

- API error rate (< 1%)
- API latency p99 (< 2s)
- Lambda errors (0)
- Database connections (< 80)

Alerts to Monitor

- `radiant-production-api-5xx-errors`

- radiant-production-api-latency
- radiant-production-lambda-errors
- radiant-production-db-cpu

Post-Deployment

Verification Checklist

- ☐ Health endpoints responding
- ☐ Login working
- ☐ Key user flows functional
- ☐ No error spike in CloudWatch
- ☐ No increase in support tickets

Documentation

- ☐ Update CHANGELOG.md
- ☐ Create GitHub Release
- ☐ Notify stakeholders
- ☐ Update status page (if applicable)

Troubleshooting

Deployment Stuck

```
# Check CloudFormation status
aws cloudformation describe-stacks --stack-name RadiantProductionApi

# Check for stack events
aws cloudformation describe-stack-events --stack-name RadiantProductionApi

# Force continue if stuck
aws cloudformation continue-update-rollback --stack-name RadiantProductionApi
```

Lambda Not Updating

```
# Force function update
aws lambda update-function-code \
  --function-name radiant-production-router \
  --s3-bucket radiant-artifacts \
  --s3-key lambda/latest.zip
```

Dashboard Not Updating

```
# Check CloudFront status
aws cloudfront get-distribution --id <DIST_ID>

# Create aggressive invalidation
aws cloudfront create-invalidation \
  --distribution-id <DIST_ID> \
  --paths "/*"
```