Contents

RADIANT Disaster Recovery Guide

Overview

This document outlines disaster recovery (DR) procedures for the RADIANT platform, including backup strategies, recovery procedures, and business continuity plans.

Recovery Objectives

Metric	Target	Maximum
RTO (Recovery Time Objective)	1 hour	4 hours
RPO (Recovery Point Objective)	5 minutes	1 hour

Backup Strategy

Database Backups

Automated Backups (Aurora)

```
// CDK Configuration
const database = new rds.DatabaseCluster(this, 'Database', {
```

```
  backup: {
    retention: cdk.Duration.days(35),      // 35 days retention
    preferredWindow: '03:00-04:00',        // 3-4 AM UTC
  },
  deletionProtection: true,
  storageEncrypted: true,
});
```

Point-in-Time Recovery Aurora supports point-in-time recovery (PITR) to any second within the retention period.

```
# Restore to specific point in time
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-production \
  --db-cluster-identifier radiant-production-restored \
  --restore-to-time "2024-12-24T10:30:00Z" \
  --vpc-security-group-ids sg-xxx \
  --db-subnet-group-name radiant-production
```

Manual Snapshots

```
# Create manual snapshot before major changes
aws rds create-db-cluster-snapshot \
  --db-cluster-identifier radiant-production \
  --db-cluster-snapshot-identifier radiant-production-pre-migration-$(date +%Y%m%d)
```

S3 Backups

Versioning

```
// All S3 buckets have versioning enabled
const bucket = new s3.Bucket(this, 'Storage', {
  versioned: true,
  lifecycleRules: [
    {
      noncurrentVersionExpiration: cdk.Duration.days(90),
    },
  ],
});
```

Cross-Region Replication

```
// Production buckets replicate to DR region
const replicationRule = {
  destination: {
    bucket: drBucket.bucketArn,
    storageClass: s3.StorageClass.STANDARD_IA,
  },
  status: 'Enabled',
};
```

Secrets Backup

```
# Export secrets for DR (store securely!)
aws secretsmanager get-secret-value \
  --secret-id radiant-production-db \
  --query SecretString \
  --output text > /secure/path/db-credentials.json
```

Failure Scenarios

Scenario 1: Single AZ Failure

Impact: Partial service degradation **Recovery:** Automatic (Multi-AZ)

Aurora automatically fails over to a read replica in another AZ.

```
# Monitor failover
aws rds describe-events \
  --source-type db-cluster \
  --source-identifier radiant-production \
  --duration 60
```

Scenario 2: Database Corruption

Impact: Data integrity issues **Recovery:** Point-in-time restore

1. Identify corruption time
2. Restore to point before corruption
3. Validate data integrity
4. Switch traffic to restored database

```
# Step 1: Identify issue time from logs
aws logs filter-log-events \
  --log-group-name /aws/rds/cluster/radiant-production/error \
  --start-time $(date -d '24 hours ago' +%s000)

# Step 2: Restore
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-production \
  --db-cluster-identifier radiant-dr-$(date +%Y%m%d%H%M) \
  --restore-to-time "2024-12-24T09:00:00Z"

# Step 3: Update Lambda environment to use new cluster
aws lambda update-function-configuration \
  --function-name radiant-production-router \
  --environment "Variables={DB_CLUSTER_ARN=arn:aws:rds:...}"
```

Scenario 3: Region Failure

Impact: Complete service outage **Recovery:** Failover to DR region

1. Activate DR region infrastructure

2. Promote Aurora Global Database secondary
3. Update Route 53 to point to DR region
4. Verify service health

```
# Step 1: Promote DR database
aws rds failover-global-cluster \
  --global-cluster-identifier radiant-global \
  --target-db-cluster-identifier radiant-dr-cluster

# Step 2: Update DNS
aws route53 change-resource-record-sets \
  --hosted-zone-id Z123456 \
  --change-batch file://dr-dns-failover.json
```

Scenario 4: Accidental Deletion

Impact: Data loss **Recovery:** Restore from backup

```
# Restore deleted S3 objects
aws s3api list-object-versions \
  --bucket radiant-storage-production \
  --prefix "deleted/path/" \
  --query 'DeleteMarkers[?IsLatest==`true`]'

# Restore specific version
aws s3api delete-object \
  --bucket radiant-storage-production \
  --key "path/to/file" \
  --version-id "delete-marker-version-id"
```

Scenario 5: Security Breach

Impact: Potential data exposure **Recovery:** Isolation and investigation

1. Isolate affected systems
2. Rotate all credentials
3. Investigate scope
4. Restore from known-good backup
5. Notify affected parties

```
# Step 1: Disable API access
aws apigateway update-stage \
  --rest-api-id abc123 \
  --stage-name v2 \
  --patch-operations op=replace,path=/throttling/rateLimit,value=0

# Step 2: Rotate database credentials
aws secretsmanager rotate-secret \
  --secret-id radiant-production-db
```

```
# Step 3: Invalidate all sessions
aws cognito-idp admin-user-global-sign-out \
  --user-pool-id us-east-1_xxx \
  --username "*"
```

Recovery Procedures

Database Recovery Runbook

```
#!/bin/bash
# database-recovery.sh

set -e

CLUSTER_ID="radiant-production"
RESTORE_TIME="${1:-$(date -d '1 hour ago' -Iseconds)}"
NEW_CLUSTER_ID="radiant-dr-$(date +%Y%m%d%H%M)"

echo "  Starting database recovery..."
echo "   Source: $CLUSTER_ID"
echo "   Restore time: $RESTORE_TIME"
echo "   New cluster: $NEW_CLUSTER_ID"

# Create restored cluster
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier "$CLUSTER_ID" \
  --db-cluster-identifier "$NEW_CLUSTER_ID" \
  --restore-to-time "$RESTORE_TIME" \
  --db-subnet-group-name radiant-production \
  --vpc-security-group-ids sg-xxx

echo "  Waiting for cluster to be available..."
aws rds wait db-cluster-available \
  --db-cluster-identifier "$NEW_CLUSTER_ID"

# Create instance
aws rds create-db-instance \
  --db-instance-identifier "${NEW_CLUSTER_ID}-instance-1" \
  --db-cluster-identifier "$NEW_CLUSTER_ID" \
  --db-instance-class db.r6g.large \
  --engine aurora-postgresql

echo "  Waiting for instance to be available..."
aws rds wait db-instance-available \
  --db-instance-identifier "${NEW_CLUSTER_ID}-instance-1"

echo "  Database restored successfully!"
echo "   Endpoint: $(aws rds describe-db-clusters \
```

```
  --db-cluster-identifier "$NEW_CLUSTER_ID" \
  --query 'DBClusters[0].Endpoint' --output text)"
```

Full Service Recovery Runbook

```
#!/bin/bash
# full-recovery.sh

set -e

echo "  RADIANT Full Service Recovery"
echo "================================="

# Step 1: Database
echo "Step 1: Recovering database..."
./scripts/dr/database-recovery.sh

# Step 2: Update Lambda configurations
echo "Step 2: Updating Lambda configurations..."
for fn in router admin billing localization configuration; do
  aws lambda update-function-configuration \
    --function-name "radiant-production-$fn" \
    --environment "Variables={DB_CLUSTER_ARN=$NEW_DB_ARN}"
done

# Step 3: Clear caches
echo "Step 3: Clearing caches..."
redis-cli -h radiant-cache.xxx.cache.amazonaws.com FLUSHALL

# Step 4: Verify health
echo "Step 4: Verifying service health..."
curl -f https://api.radiant.example.com/v2/health || exit 1

# Step 5: Run smoke tests
echo "Step 5: Running smoke tests..."
k6 run --env BASE_URL=https://api.radiant.example.com tests/load/k6-config.js

echo "  Recovery complete!"
```

Testing DR Procedures

Quarterly DR Drill

1. **Preparation**
   - Schedule maintenance window
   - Notify stakeholders
   - Prepare rollback plan
2. **Execution**
   - Simulate failure scenario

- Execute recovery procedures
- Measure RTO/RPO

3. **Validation**
   - Verify data integrity
   - Run integration tests
   - Check all services
4. **Documentation**
   - Record actual RTO/RPO
   - Document issues encountered
   - Update procedures

DR Test Checklist

- ☐ Database point-in-time recovery tested
- ☐ S3 object recovery tested
- ☐ Secret rotation tested
- ☐ Lambda rollback tested
- ☐ DNS failover tested
- ☐ Communication plan executed
- ☐ Recovery time recorded
- ☐ Post-mortem completed

Communication Plan

Escalation Matrix

Severity	Response Time	Notify
SEV1	15 min	Eng Lead, CTO, Status Page
SEV2	30 min	Eng Lead, Status Page
SEV3	2 hours	On-call team

Status Page Updates

```
# Update status page (example with Statuspage.io)
curl -X POST https://api.statuspage.io/v1/pages/xxx/incidents \
  -H "Authorization: OAuth $STATUSPAGE_API_KEY" \
  -d '{
    "incident": {
      "name": "Service Degradation",
      "status": "investigating",
      "body": "We are investigating reports of API errors."
    }
  }'
```

Infrastructure as Code

All DR infrastructure is defined in CDK:

```typescript
// lib/stacks/dr-stack.ts
export class DRStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props: DRStackProps) {
    super(scope, id, props);

    // Global Database for cross-region replication
    const globalCluster = new rds.CfnGlobalCluster(this, 'GlobalCluster', {
      globalClusterIdentifier: 'radiant-global',
      sourceDbClusterIdentifier: props.primaryClusterArn,
    });

    // S3 Cross-Region Replication
    const drBucket = new s3.Bucket(this, 'DRBucket', {
      bucketName: `radiant-storage-dr-${props.drRegion}`,
    });
  }
}
```

Contacts

Role	Contact	Backup
DR Coordinator	dr@radiant.example.com	cto@radiant.example.com
Database Admin	dba@radiant.example.com	platform@radiant.example.com
Security	security@radiant.example.com	cto@radiant.example.com