RADIANT Platform v4.18.2 - Complete Documentation

RADIANT Team

2025-12-28

Contents

RADIANT v4.17.0 - Deployed System Guide

Operations and infrastructure documentation for the deployed RADIANT platform

Last Updated: {{BUILD_DATE}} Version: {{RADIANT_VERSION}}

Table of Contents

---

1. System Overview

1.1 Platform Summary

RADIANT is a multi-tenant AWS SaaS platform deployed across multiple AWS services:

Component	AWS Service	Purpose
Compute	Lambda	Serverless API handlers
Database	Aurora PostgreSQL	Primary data store
Cache	ElastiCache Redis	Session & response caching
Storage	S3	File storage, artifacts
Auth	Cognito	User authentication
API	API Gateway	REST & WebSocket APIs
AI	SageMaker, Bedrock	Model hosting
CDN	CloudFront	Static asset delivery
DNS	Route 53	Domain management

1.2 Environment Tiers

Environment	Purpose	Domain
Production	Live customer traffic	`api.{{RADIANT_DOMAIN}}`
Staging	Pre-release testing	`staging-api.{{RADIANT_DOMAIN}}`
Development	Development & testing	`dev-api.{{RADIANT_DOMAIN}}`

1.3 Region Configuration

Region	Role	Services
`us-east-1`	Primary	All services
`us-west-2`	DR Failover	Database replica, S3 replication
`eu-west-1`	EU Data Residency	Optional for GDPR compliance

---

2. Infrastructure Components

2.1 VPC Architecture

```
                    RADIANT VPC (10.0.0.0/16)



                   Public Subnets

     us-east-1a      us-east-1b      us-east-1c
     10.0.1.0/24     10.0.2.0/24     10.0.3.0/24
     NAT Gateway     NAT Gateway




                   Private Subnets (App)

     us-east-1a      us-east-1b      us-east-1c
     10.0.11.0/24    10.0.12.0/24    10.0.13.0/24
     Lambda ENIs     Lambda ENIs     Lambda ENIs




                   Private Subnets (Data)

     us-east-1a      us-east-1b      us-east-1c
     10.0.21.0/24    10.0.22.0/24    10.0.23.0/24
     Aurora, RDS     ElastiCache     Backup
```

2.2 Security Groups

Security Group	Inbound Rules	Purpose
`sg-lambda`	None (outbound only)	Lambda functions
`sg-aurora`	5432 from sg-lambda	Database access
`sg-redis`	6379 from sg-lambda	Cache access
`sg-alb`	443 from 0.0.0.0/0	Load balancer

2.3 IAM Roles

Role	Attached Policies	Used By
`radiant-lambda-role`	AWSLambdaVPCAccess, SecretsManager, S3, SageMaker	Lambda functions
`radiant-api-gw-role`	CloudWatchLogs, Lambda invoke	API Gateway
`radiant-sagemaker-role`	SageMakerFullAccess, S3	SageMaker endpoints

---

3. Deployment Architecture

3.1 CDK Stacks

Stack	Resources	Dependencies
`NetworkingStack`	VPC, Subnets, NAT	None
`SecurityStack`	IAM, KMS, Security Groups	Networking
`DataStack`	Aurora, ElastiCache, S3	Security
`AuthStack`	Cognito User Pool	Data
`ApiStack`	API Gateway, Lambda	Auth, Data
`AiStack`	SageMaker, Bedrock config	Api
`MonitoringStack`	CloudWatch, Alarms	All
`AdminStack`	Admin Dashboard hosting	Api

3.2 Deployment Commands

```
# Deploy all stacks to development
npm run deploy:dev

# Deploy to staging with approval
npm run deploy:staging

# Deploy to production (requires approval)
npm run deploy:prod

# Deploy specific stack
cd packages/infrastructure
cdk deploy RadiantApiStack --context environment=prod
```

3.3 Environment Variables

Variable	Description	Source
`AURORA_CLUSTER_ARN`	Database cluster ARN	CDK output
`AURORA_SECRET_ARN`	Database credentials	Secrets Manager
`COGNITO_USER_POOL_ID`	Auth pool ID	CDK output
`REDIS_URL`	Cache connection string	CDK output
`S3_BUCKET`	Storage bucket name	CDK output

---

4. Service Endpoints

4.1 API Gateway Endpoints

Endpoint	Method	Description
`/v1/chat`	POST	Send chat message
`/v1/chat/stream`	WebSocket	Streaming responses
`/v1/models`	GET	List available models
`/v1/usage`	GET	Usage statistics
`/admin/v1/*`	Various	Admin API

4.2 Health Check Endpoints

Endpoint	Expected Response	Checks
`/health`	`{"status":"healthy"}`	Lambda running
`/health/db`	`{"status":"connected"}`	Aurora connection
`/health/cache`	`{"status":"connected"}`	Redis connection
`/health/providers`	Provider status array	AI provider APIs

4.3 Internal Endpoints

Service	Endpoint	Port
Aurora PostgreSQL	`radiant-cluster.xxx.us-east-1.rds.amazonaws.com`	5432
ElastiCache Redis	`radiant-cache.xxx.cache.amazonaws.com`	6379
SageMaker Runtime	Via AWS SDK	-

---

5. Database Operations

5.1 Aurora Configuration

Setting	Value
Engine	PostgreSQL 15.4
Instance Type	Serverless v2
Min ACUs	2 (dev), 8 (prod)
Max ACUs	16 (dev), 128 (prod)

Setting	Value
Storage	Auto-scaling to 128 TB
Encryption	AES-256 (KMS)

5.2 Connection Management

```sql
-- Check active connections
SELECT count(*) FROM pg_stat_activity;

-- View connection by application
SELECT application_name, count(*)
FROM pg_stat_activity
GROUP BY application_name;

-- Terminate idle connections
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE state = 'idle'
AND query_start < now() - interval '10 minutes';
```

5.3 Row-Level Security

All tenant data is protected by RLS:

```sql
-- Verify RLS is enabled
SELECT tablename, rowsecurity
FROM pg_tables
WHERE schemaname = 'public' AND rowsecurity = true;

-- Set tenant context (done automatically by Lambda)
SET app.current_tenant_id = 'tenant-uuid';

-- Queries automatically filter by tenant
SELECT * FROM users; -- Only returns current tenant's users
```

5.4 Database Migrations

```bash
# Run pending migrations
cd packages/infrastructure
npm run migrate:up

# Rollback last migration
npm run migrate:down

# View migration status
npm run migrate:status
```

---

6. Lambda Functions

6.1 Function Inventory

Function	Memory	Timeout	Trigger
`radiant-api`	1024 MB	30s	API Gateway
`radiant-brain`	2048 MB	60s	API Gateway
`radiant-webhooks`	512 MB	30s	EventBridge
`radiant-scheduler`	512 MB	300s	EventBridge (cron)
`radiant-batch`	3008 MB	900s	SQS

6.2 Monitoring Lambda

```
# View recent invocations
aws lambda get-function \
  --function-name radiant-api \
  --query 'Configuration.[FunctionName,MemorySize,Timeout,LastModified]'

# Check concurrent executions
aws cloudwatch get-metric-statistics \
  --namespace AWS/Lambda \
  --metric-name ConcurrentExecutions \
  --dimensions Name=FunctionName,Value=radiant-api \
  --start-time $(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%SZ) \
  --end-time $(date -u +%Y-%m-%dT%H:%M:%SZ) \
  --period 300 \
  --statistics Maximum
```

6.3 Cold Start Optimization

Strategy	Implementation
Provisioned Concurrency	10 instances for `radiant-api`
Keep-Warm	CloudWatch scheduled ping every 5 min
Connection Pooling	RDS Proxy for database
Lazy Loading	Defer non-critical imports

---

7. AI Provider Integration

7.1 External Providers

Provider	Models	Authentication
OpenAI	GPT-4, GPT-3.5	API Key
Anthropic	Claude 3, Claude 2	API Key
Google	Gemini Pro, PaLM 2	Service Account
Cohere	Command, Embed	API Key
Mistral	Mistral Large, Medium	API Key

7.2 Self-Hosted Models (SageMaker)

Model	Instance Type	Thermal Default
LLaMA-2-70B	ml.g5.48xlarge	COLD
Stable Diffusion XL	ml.g5.2xlarge	COLD
Whisper Large	ml.g4dn.xlarge	COLD
CodeLlama-34B	ml.g5.12xlarge	COLD

7.3 Model Thermal States

```
# Check endpoint status
aws sagemaker describe-endpoint \
  --endpoint-name radiant-llama-70b

# Scale endpoint to warm
aws sagemaker update-endpoint-weights-and-capacities \
  --endpoint-name radiant-llama-70b \
  --desired-weights-and-capacities '[{"VariantName":"AllTraffic","DesiredInstanceCount":1}]'

# Scale to zero (cold)
aws sagemaker update-endpoint-weights-and-capacities \
  --endpoint-name radiant-llama-70b \
  --desired-weights-and-capacities '[{"VariantName":"AllTraffic","DesiredInstanceCount":0}]'
```

---

8. Monitoring & Observability

8.1 CloudWatch Dashboards

Dashboard	Metrics
Platform Overview	Request rate, latency, errors
Database Performance	Connections, ACUs, query time
AI Provider Health	Success rate, latency by provider
Billing & Usage	Credits consumed, revenue

8.2 Key Metrics

Metric	Warning	Critical
API Error Rate	> 1%	> 5%
P99 Latency	> 5s	> 15s
Aurora ACU Usage	> 80%	> 95%
Lambda Errors	> 10/min	> 50/min

8.3 Alerting

```
# List configured alarms
aws cloudwatch describe-alarms \
  --alarm-name-prefix radiant

# View alarm state
aws cloudwatch describe-alarm-history \
  --alarm-name radiant-high-error-rate
```

8.4 Log Groups

Log Group	Retention	Content
`/aws/lambda/radiant-api`	30 days	API request logs
`/aws/lambda/radiant-brain`	30 days	Model routing logs
`/radiant/audit`	365 days	Security audit logs
`/radiant/billing`	2 years	Billing events

---

9. Security Configuration

9.1 Encryption

Data	Encryption
Database at rest	AES-256 (KMS CMK)
S3 objects	AES-256 (SSE-S3)
Secrets	KMS envelope encryption
API traffic	TLS 1.3

9.2 Secrets Management

```
# List RADIANT secrets
aws secretsmanager list-secrets \
  --filters Key=name,Values=radiant

# Rotate database credentials
aws secretsmanager rotate-secret \
  --secret-id radiant/aurora-credentials

# View secret metadata
aws secretsmanager describe-secret \
  --secret-id radiant/openai-api-key
```

9.3 WAF Rules

Rule	Action	Purpose
Rate Limiting	Block	> 1000 req/min per IP
SQL Injection	Block	SQLi pattern detection
XSS	Block	Cross-site scripting
Geo Blocking	Block	Sanctioned countries

---

10. Backup & Recovery

10.1 Automated Backups

Resource	Frequency	Retention
Aurora Snapshots	Daily	35 days
S3 Objects	Continuous	Versioned
DynamoDB	Point-in-time	35 days
Secrets	Auto-versioned	30 versions

10.2 Recovery Procedures

Database Point-in-Time Recovery:

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-cluster \
  --db-cluster-identifier radiant-cluster-restored \
  --restore-to-time 2024-01-15T10:00:00Z
```

S3 Object Recovery:

```
# List object versions
aws s3api list-object-versions \
  --bucket radiant-storage \
  --prefix tenant/123/

# Restore specific version
aws s3api copy-object \
  --bucket radiant-storage \
  --copy-source radiant-storage/file.txt?versionId=xxx \
  --key file.txt
```

10.3 Disaster Recovery

RPO	RTO	Strategy
1 hour	4 hours	Cross-region Aurora replica, S3 CRR

---

11. Maintenance Procedures

11.1 Routine Maintenance

Task	Frequency	Procedure
Security patches	Weekly	Automated via CDK
Log rotation	Daily	Automated
Database vacuum	Weekly	Automated
Certificate renewal	60 days	ACM auto-renewal

11.2 Scaling Operations

```
# Scale Aurora
aws rds modify-db-cluster \
  --db-cluster-identifier radiant-cluster \
  --serverless-v2-scaling-configuration MinCapacity=16,MaxCapacity=64
```

```
# Update Lambda memory
aws lambda update-function-configuration \
  --function-name radiant-api \
  --memory-size 2048
```

11.3 Deployment Checklist

- ☐ Run tests in staging
- ☐ Review CloudWatch for anomalies
- ☐ Announce maintenance window
- ☐ Deploy with `--require-approval`
- ☐ Verify health checks
- ☐ Monitor error rates for 30 min
- ☐ Update status page

---

12. Troubleshooting

12.1 Common Issues

High Latency:

```
# Check database performance
aws cloudwatch get-metric-statistics \
  --namespace AWS/RDS \
  --metric-name DatabaseConnections \
  --dimensions Name=DBClusterIdentifier,Value=radiant-cluster

# Check Lambda duration
aws logs filter-log-events \
  --log-group-name /aws/lambda/radiant-api \
  --filter-pattern "REPORT Duration"
```

Connection Errors:

```
# Verify security group rules
aws ec2 describe-security-groups --group-ids sg-xxx

# Check VPC endpoints
aws ec2 describe-vpc-endpoints \
  --filters Name=vpc-id,Values=vpc-xxx
```

12.2 Emergency Procedures

Rollback Deployment:

```
# List recent deployments
aws cloudformation list-stacks \
  --stack-status-filter UPDATE_COMPLETE

# Rollback to previous version
cdk deploy --rollback
```

Disable Problematic Feature:

```
# Update feature flag
aws ssm put-parameter \
  --name /radiant/features/new-feature \
  --value "false" \
  --overwrite
```

Version History

Version	Date	Changes
4.17.0	{{BUILD_DATE}}	Initial system guide

This documentation is automatically generated as part of the RADIANT build process.

RADIANT Architecture v4.17.0

Overview

RADIANT is a multi-tenant AWS SaaS platform for AI model access and orchestration. It consists of:

1. **Swift Deployer App** - macOS GUI for infrastructure deployment
2. **AWS Infrastructure** - CDK-based cloud infrastructure
3. **Admin Dashboard** - Next.js admin UI (Phase 3)

Technology Stack

Layer	Technology
Swift App	SwiftUI, macOS 13.0+, Swift 5.9+, GRDB
Infrastructure	AWS CDK (TypeScript), Aurora PostgreSQL, Lambda
Dashboard	Next.js 14, TypeScript, Tailwind CSS
AI Integration	106+ models, LiteLLM, SageMaker

Monorepo Structure

```
radiant/
  packages/
     shared/              # @radiant/shared - Types & constants
     infrastructure/      # @radiant/infrastructure - CDK stacks
  apps/
     swift-deployer/      # RadiantDeployer macOS app
  functions/              # Lambda functions (Phase 2)
  migrations/             # Database migrations (Phase 2)
  docs/                   # Specifications
```

Phase 1 Architecture

@radiant/shared Package

Single source of truth for all types and constants:

```
// Types
- app.types.ts         // ManagedApp, DeploymentStatus, etc.
- environment.types.ts // Environment, TierConfig, etc.
- ai.types.ts          // AIProvider, AIModel, etc.
- admin.types.ts       // Administrator, Permissions
- billing.types.ts    // UsageEvent, Invoice, etc.
- compliance.types.ts // PHI, AuditLog, etc.

// Constants
- version.ts          // RADIANT_VERSION = "4.17.0"
- tiers.ts            // Tier 1-5 configurations
- regions.ts          // AWS region configs
- providers.ts        // AI provider definitions
```

Swift Deployer App

```
RadiantDeployer/
  RadiantDeployerApp.swift  # @main entry point
  AppState.swift            # Global state management
  Models/
     ManagedApp.swift      # App configuration model
     Credentials.swift     # AWS credentials model
     Deployment.swift      # Deployment state/progress
  Services/
     CredentialService.swift  # Keychain credential storage
     CDKService.swift         # CDK command execution
     AWSService.swift         # AWS API interactions
  Views/
      MainView.swift        # NavigationSplitView container
      AppsView.swift        # Application grid
      DeployView.swift      # Deployment wizard
      ProvidersView.swift   # AI provider list
      ModelsView.swift      # AI model catalog
      SettingsView.swift    # Preferences
```

CDK Infrastructure Stacks

```
packages/infrastructure/
  bin/radiant.ts            # CDK app entry point
  lib/stacks/
      foundation-stack.ts   # KMS, SSM parameters
      networking-stack.ts   # VPC, subnets, endpoints
      security-stack.ts     # Security groups, WAF
```

Infrastructure Tiers

Tier	Name	VPC CIDR	AZs	NAT	Aurora ACU	Features
1	SEED	/20	2	1	0.5-2	Dev only
2	STARTUP	/18	2	1	1-8	WAF, GuardDuty
3	GROWTH	/17	3	2	2-16	Self-hosted models, HIPAA
4	SCALE	/16	3	3	4-64	Multi-region, Global DB
5	ENTERPRISE	/14	3	3	8-128	Full enterprise

Deployment Flow

```
 Swift Deployer
```

```
  (macOS)

        1. Configure credentials
        2. Select app/tier/env
        3. Initiate deployment


 CDK Deploy
(via Process)

        4. Bootstrap AWS
        5. Synth stacks
        6. Deploy in order


AWS Account
(VPC, RDS,
 Lambda, etc)
```

Phase 1 Deliverables

- ☒ Monorepo structure (pnpm workspace)
- ☒ @radiant/shared package with all types
- ☒ @radiant/infrastructure base CDK stacks
- ☒ RadiantDeployer Swift app with full UI
- ☒ Credential management via Keychain
- ☒ CDK service for deployment execution
- ☒ Documentation

Future Phases

Phase	Sections	Description
2	3-7	AI stacks, Lambdas, Database
3	8-9	Admin Dashboard, Deployment Guide
4	10-17	Visual AI, Brain, Analytics
5	18-28	Think Tank, Collaboration
6	29-35	Registry, Time Machine
7	36-39	Neural Engine, Workflows
8	40-42	Isolation, i18n, Config
9	43-46	Billing System

RADIANT v4.18.0 - Deployment Guide

Overview

This guide covers deploying the RADIANT platform from development to production. The platform consists of:

1. **AWS Infrastructure** - CDK stacks for all cloud resources
2. **Admin Dashboard** - Next.js admin interface
3. **Swift Deployer App** - macOS deployment tool with AI assistant

What's New in v4.18.0

- **Unified Package System** - Deploy with atomic component versioning
- **AI Assistant** - Claude-powered deployment guidance in Swift app
- **Configurable Timeouts** - SSM-synced operation timeouts
- **Cost Management** - Real-time cost tracking and alerts
- **Compliance Reports** - SOC2, HIPAA, GDPR, ISO27001 reporting

Prerequisites

AWS Account Setup

Requirement	Action	Verification
AWS Account	Create or use existing	Account ID available
IAM User	Create with AdministratorAccess	Access keys configured
AWS CLI	Install and configure	`aws sts get-caller-identity`
Route 53 Domain (optional)	Register domain	Domain in hosted zone
ACM Certificate (optional)	Request in us-east-1	Certificate validated

Development Environment

Requirement	Version	Verification
Node.js	20.x LTS	`node --version`
pnpm	8.x+	`pnpm --version`
AWS CDK CLI	2.x	`cdk --version`
Xcode	15.x+	`xcode-select -p`

Requirement	Version	Verification
Swift	5.9+	`swift --version`

Quick Start

Automated Deployment

Use the deployment script for a streamlined deployment:

```
# Deploy to dev environment
./scripts/deploy.sh --environment dev

# Deploy to staging
./scripts/deploy.sh --environment staging

# Deploy to production
./scripts/deploy.sh --environment prod
```

> **Note:** RADIANT uses a unified deployment model. All features are available in every deployment. Licensing restrictions are handled at the application level, not infrastructure level.

Verify Deployment

After deployment, verify all resources:

```
./scripts/verify-deployment.sh --environment dev
```

Manual Deployment

1. Install Dependencies

```
cd radiant
npx pnpm install
```

2. Build Shared Package

```
cd packages/shared
npm run build
```

3. Build Lambda Functions

```
cd packages/infrastructure/lambda
npm install --legacy-peer-deps
npm run build
```

4. Build Admin Dashboard

```
cd apps/admin-dashboard
npm install
npm run build
```

CDK Deployment

Bootstrap CDK (One-time per account/region)

```
cd packages/infrastructure
npx cdk bootstrap aws://ACCOUNT_ID/us-east-1 --qualifier radiant
```

Deploy All Stacks

```
# Deploy in order with dependencies
npx cdk deploy --all \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Deploy Individual Stacks

```
# Phase 1: Foundation
npx cdk deploy Radiant-dev-Foundation --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Networking --context environment=dev --context tier=1

# Phase 2: Security & Data
npx cdk deploy Radiant-dev-Security --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Data --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Storage --context environment=dev --context tier=1

# Phase 3: Auth & AI
npx cdk deploy Radiant-dev-Auth --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-AI --context environment=dev --context tier=1

# Phase 4: API & Admin
npx cdk deploy Radiant-dev-API --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Admin --context environment=dev --context tier=1
```

Database Migrations

After infrastructure is deployed, run database migrations:

```
# Connect to Aurora and run migrations
cd packages/infrastructure/migrations
./run-migrations.sh --environment dev
```

Migration files are applied in order (44 total): 1. `001_initial_schema.sql` - Base tables 2. `002_tenant_isolation.sql` - RLS policies 3. `003_ai_models.sql` - Providers and models 4. `004_usage_billing.sql` - Usage tracking 5. `005_admin_approval.sql` - Audit logs 6. `006_self_hosted_models.sql` - SageMaker config 7. `007_external_providers.sql` - Provider settings 8. … (see `migrations/` for full list) 44. `044_cost_experiments_security.sql` - Cost tracking, A/B testing, security

Post-Deployment Configuration

Create First Super Admin

```
aws cognito-idp admin-create-user \
  --user-pool-id YOUR_ADMIN_POOL_ID \
  --username admin@example.com \
  --user-attributes Name=email,Value=admin@example.com \
  --temporary-password TempPass123! \
  --message-action SUPPRESS

aws cognito-idp admin-add-user-to-group \
  --user-pool-id YOUR_ADMIN_POOL_ID \
  --username admin@example.com \
  --group-name super_admin
```

Configure AI Providers

1. Navigate to Admin Dashboard → Providers
2. Add API keys for external providers:
   - OpenAI
   - Anthropic
   - Google AI
   - xAI (Grok)
   - DeepSeek
3. Verify connectivity with test requests

Environment Configuration

Tiers

Tier	Name	Use Case
1	SEED	Development, testing
2	STARTUP	Small production
3	GROWTH	Medium production
4	SCALE	Large production
5	ENTERPRISE	Enterprise with compliance

Environment Variables

Required environment variables for deployment:

```
export AWS_REGION=us-east-1
export AWS_ACCOUNT_ID=123456789012
export RADIANT_ENVIRONMENT=dev  # dev, staging, prod
export RADIANT_TIER=1           # 1-5
export RADIANT_DOMAIN=example.com
```

Verification

Health Checks

```
# API Health
curl https://YOUR_API_ENDPOINT/health

# Expected response:
# {"status":"healthy","version":"4.18.0"}
```

Smoke Tests

```
# Test chat completions
curl -X POST https://YOUR_API_ENDPOINT/v1/chat/completions \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"model":"gpt-4o-mini","messages":[{"role":"user","content":"Hello"}]}'
```

Troubleshooting

Common Issues

Issue	Cause	Solution
CDK bootstrap fails	Missing permissions	Ensure IAM user has AdministratorAccess
Aurora connection timeout	Security group	Check VPC endpoint and security group rules
Lambda cold starts	Function size	Enable provisioned concurrency for critical functions
Cognito auth fails	Pool configuration	Verify callback URLs and client settings

Logs

```
# View Lambda logs
aws logs tail /aws/lambda/Radiant-dev-router --follow

# View ECS logs (LiteLLM)
aws logs tail /ecs/radiant-dev-litellm --follow
```

Cleanup

To destroy all resources:

```
cd packages/infrastructure
npx cdk destroy --all --context environment=dev --context tier=1
```

Warning: This will delete all data including databases. Export data before destroying.

CI/CD Pipeline

RADIANT includes a GitHub Actions CI/CD pipeline:

Workflow Stages

Stage	Trigger	Actions
Lint	All PRs	ESLint, TypeScript check
Build	All PRs	Build shared, infrastructure, dashboard
Test	All PRs	Unit tests, coverage report
CDK Synth	All PRs	Validate CDK templates
Deploy Dev	Merge to develop	Auto-deploy to dev
Deploy Prod	Merge to main	Deploy with approval

Pre-commit Hooks

Pre-commit hooks run automatically via Husky:

```
# Hooks run on every commit:
- lint-staged (ESLint, Prettier)
- Secret detection
- Version bump enforcement
- Discrete validation
```

To bypass (not recommended):

```
git commit --no-verify -m "message"
```

Manual Deployment from CI

```
# Trigger deployment workflow
gh workflow run deploy.yml -f environment=staging -f tier=2
```

Testing Before Deployment

Always run tests before deploying:

```
# Run all tests
pnpm test

# Run E2E tests
cd apps/admin-dashboard && pnpm test:e2e

# Run CDK synthesis (validates templates)
cd packages/infrastructure && npx cdk synth
```

See Testing Guide for comprehensive testing information.

Support

For issues, check: 1. CloudWatch Logs for error details 2. CDK diff to verify expected changes 3. AWS Console for resource status 4. Troubleshooting Guide 5. Error Codes Reference

RADIANT Platform - Administrator Guide

Complete guide for managing the RADIANT AI Platform via the Admin Dashboard

Version: 4.18.1 | Last Updated: December 2024

Table of Contents

---

1. Introduction

1.1 What is RADIANT?

RADIANT is a multi-tenant AWS SaaS platform providing unified access to 106+ AI models through:

- **50 External Provider Models**: OpenAI, Anthropic, Google, xAI, DeepSeek, and more
- **56 Self-Hosted Models**: Running on AWS SageMaker for cost control and privacy
- **Intelligent Routing**: Brain router for optimal model selection
- **Neural Engine**: Personalization learning from user interactions

1.2 Administrator Roles

Role	Permissions	Use Case
Super Admin	Full access to all features	Platform owner
Admin	Tenant management, billing, models	Operations team
Operator	Read access, limited actions	Support team
Auditor	Read-only access to logs	Compliance team

Role Details

Super Admin - The highest privilege level with unrestricted access: - Create and delete tenants - Manage all administrators - Access all billing and financial data - Modify system-wide configuration - Approve production database migrations - Impersonate any tenant for debugging - Access compliance and audit reports - Typically limited to 1-3 people (CTO, lead engineer)

Admin - Day-to-day operations management: - Create and modify tenants (cannot delete) - Manage users within tenants - Configure AI models and providers - View billing data (cannot modify pricing) - Monitor system health - Cannot access other admin accounts - Typically assigned to operations team members

Operator - Limited support and monitoring: - View tenant information (read-only) - View user issues and support tickets - Monitor system health dashboards - Cannot modify any configuration - Cannot access billing or sensitive data - Typically assigned to support staff

Auditor - Compliance and security review: - Full read access to audit logs - Access to compliance reports - Cannot modify anything - Cannot view sensitive data (API keys, passwords) - Access is logged for compliance - Typically assigned to compliance officers or external auditors

1.3 Key Concepts

Concept	Description
Tenant	Organization with isolated data
User	End-user within a tenant
Subscription	Billing tier (1-7)
Credits	Currency for AI usage
API Key	Authentication for API access
App	Consumer application (Think Tank, etc.)

Tenant Architecture Explained

A **Tenant** represents a complete organization using RADIANT. Each tenant has:

- **Complete Data Isolation**: All data is stored with tenant IDs and protected by PostgreSQL Row-Level Security (RLS). One tenant can never access another tenant's data, even if there's a bug in application code.
- **Separate Billing**: Each tenant has its own subscription, credit balance, and usage tracking. Costs are attributed to the correct tenant automatically.
- **Custom Configuration**: Tenants can customize model access, rate limits, and feature flags without affecting other tenants.
- **User Management**: Each tenant manages their own users, roles, and permissions independently.

User vs Administrator

Users are end-users who interact with RADIANT-powered applications like Think Tank. They: - Sign up and log in via Cognito - Use AI models through the API or applications - Have credits deducted for usage - Cannot access the Admin Dashboard

Administrators manage the RADIANT platform itself. They: - Access the Admin Dashboard - Manage tenants, users, and billing - Configure AI models and providers - Have no credits (administrative access is separate)

Credit System Explained

Credits are RADIANT's universal currency for AI usage:

- **1 credit = $0.01 USD** (configurable per deployment)
- Different models cost different amounts based on their API pricing
- Credits are deducted in real-time as requests complete
- Tenants can purchase credits or receive them through subscriptions
- Credits can be tracked, audited, and reported on

Example Credit Costs: | Model | Cost per 1K tokens | |——-|——————-| | GPT-4o | 5 credits input, 15 credits output | | GPT-4o-mini | 0.5 credits input, 1.5 credits output | | Claude 3.5 Sonnet | 3 credits input, 15 credits output | | Self-hosted Llama | 0.2 credits (all) |

API Key Types

RADIANT supports multiple API key types:

- **User API Keys**: Tied to a specific user, inherit user's permissions
- **Service API Keys**: For server-to-server communication, not tied to a user
- **Admin API Keys**: For administrative operations, require elevated permissions
- **Scoped Keys**: Limited to specific models, endpoints, or rate limits

---

2. Accessing the Admin Dashboard

2.1 URL and Login

1. Navigate to: `https://admin.your-domain.com`
2. Enter your email address
3. Enter your password
4. Complete MFA verification (required)

2.2 First Login

On first login:

1. You'll receive a temporary password via email
2. Enter the temporary password
3. Set a new password (12+ characters, mixed case, numbers, symbols)
4. Set up MFA using an authenticator app
5. You'll be redirected to the dashboard

2.3 Session Management

Setting	Value
Session Duration	8 hours
Idle Timeout	30 minutes
Concurrent Sessions	3 maximum
Remember Device	30 days

2.4 Password Requirements

- Minimum 12 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- At least one special character
- Cannot reuse last 10 passwords

---

3. Dashboard Overview

3.1 Main Dashboard

The dashboard displays key metrics at a glance:

```
RADIANT Admin Dashboard                        Welcome, Admin



  Tenants          Users          Requests       Revenue
    142            8,456            2.3M         $45,230
   +12%            +8%              +23%          +15%



                    Request Volume (7 days)

  Mon     Tue     Wed     Thu     Fri     Sat     Sun


Recent Activity:
• New tenant: Acme Corp (2 minutes ago)
• Model enabled: claude-3-opus (15 minutes ago)
• Alert: High API error rate (1 hour ago)
```

3.2 Navigation Menu

Section	Description
Dashboard	Overview and metrics
Tenants	Tenant management
Users	User management
Models	AI model configuration
Providers	Provider management
Billing	Subscriptions and credits
Storage	Storage usage
Orchestration	Neural engine settings
Localization	Translation management
Configuration	System settings
Security	Security monitoring

Section	Description
Compliance	Compliance reports
Experiments	A/B testing
Cost	Cost analytics
Audit	Audit logs
Migrations	Database migrations
Notifications	System alerts
Settings	Personal settings

---

4. Tenant Management

4.1 Viewing Tenants

Navigate to **Tenants** to see all organizations:

Column	Description
Name	Organization name
Plan	Subscription tier
Users	User count
Status	Active/Suspended/Trial
Created	Creation date
Last Active	Last API call

4.2 Creating a Tenant

1. Click **"+ New Tenant"**
2. Fill in required fields:
   - **Name**: Organization name
   - **Slug**: URL-friendly identifier
   - **Plan**: Initial subscription tier
   - **Admin Email**: Primary admin email
3. Configure optional settings:
   - Custom domain
   - Branding settings
   - Feature flags
4. Click **"Create Tenant"**

4.3 Tenant Details

View comprehensive tenant information:

```
Tenant: Acme Corporation


Overview          Users          Billing          Settings


Tenant ID:        tn_abc123xyz
Status:             Active
```

```
Plan:            Professional (Tier 4)
Created:         2024-01-15
Last Active:     2 minutes ago

Usage This Month:
   API Requests:     145,234
   Tokens Used:      12.5M
   Storage:          2.3 GB
   Credits Used:     $1,234.56

[Edit]  [Suspend]  [Delete]  [Impersonate]
```

4.4 Tenant Actions

Action	Description	Permission
Edit	Modify tenant settings	Admin
Suspend	Temporarily disable	Admin
Delete	Permanently remove	Super Admin
Impersonate	Login as tenant admin	Super Admin
Export	Export tenant data	Admin

4.5 Data Isolation

Each tenant has complete data isolation:

- Separate database rows with RLS
- Unique API keys
- Isolated storage buckets
- Independent usage tracking

---

5. User & Administrator Management

5.1 Administrator Roles

Role	Dashboard Access	API Access	Billing	Audit
Super Admin	Full	Full	Full	Full
Admin	Full	Full	Read	Read
Operator	Read	Read	None	Read
Auditor	Logs only	None	None	Full

5.2 Managing Administrators

Navigate to **Administrators** to:

1. **Invite New Admin**:
   - Click **"+ Invite Administrator"**
   - Enter email address
   - Select role

    - Click **“Send Invitation”**
2. **Modify Admin**:
    - Click on administrator row
    - Edit role or permissions
    - Click **“Save Changes”**
3. **Remove Admin**:
    - Click **“Remove”** button
    - Confirm removal
    - Admin’s sessions are invalidated immediately

5.3 Viewing Tenant Users

Navigate to **Tenants** → [**Tenant**] → **Users** to see:

Field	Description
Email	User email
Name	Display name
Role	Tenant role
Status	Active/Invited/Disabled
Last Login	Last authentication
API Keys	Number of active keys

5.4 User Actions

Action	Description
Reset Password	Send password reset email
Disable	Prevent login
Enable	Restore access
Delete	Remove user data
View Sessions	See active sessions

---

6. AI Model Configuration

6.1 Model Registry

Navigate to **Models** to see all available models:

```
AI Models                                       106 Total


Filter: [All ]  Category: [All ]  Status: [Enabled ]

Model               Provider    Category   Tier   Status

gpt-4o              OpenAI      Chat       1        Enabled
gpt-4o-mini         OpenAI      Chat       1        Enabled
claude-3-opus       Anthropic   Chat       2        Enabled
claude-3-sonnet     Anthropic   Chat       1        Enabled
gemini-pro          Google      Chat       1        Enabled
```

```
llama-3.1-70b       Self-Host   Chat       3        Enabled
whisper-large       Self-Host   Audio      3        Disabled

[+ Add Model]  [Import Models]  [Export Config]
```

6.2 Model Categories

Category	Description	Example Models
Chat/LLM	Text generation	GPT-4o, Claude 3, Gemini
Embedding	Vector embeddings	text-embedding-3-large
Vision	Image understanding	GPT-4V, Claude Vision
Audio	Speech-to-text	Whisper, Deepgram
Image	Image generation	DALL-E 3, Stable Diffusion
Code	Code generation	Codestral, DeepSeek Coder
Scientific	Research models	BioGPT, ChemLLM

Category Details

Chat/LLM (Large Language Models): The core of RADIANT. These models handle conversational AI, content generation, summarization, and general-purpose text tasks. They're the most commonly used and include flagship models from OpenAI, Anthropic, Google, and open-source alternatives.

Embedding Models: Convert text into numerical vectors for semantic search, similarity matching, and retrieval-augmented generation (RAG). Essential for building knowledge bases and search functionality. Vectors are typically 1536-3072 dimensions.

Vision Models: Analyze images, extract text (OCR), describe visual content, and answer questions about images. Increasingly important for document processing, accessibility, and multimodal applications.

Audio Models: Transcribe speech to text, translate audio, and identify speakers. Whisper is the most popular, offering excellent accuracy across 99 languages. Used for meeting transcription, accessibility, and voice interfaces.

Image Generation: Create images from text descriptions. DALL-E 3 offers the best prompt following, while Stable Diffusion provides more customization options. Consider content policies when enabling these.

Code Models: Specialized for programming tasks including code generation, explanation, debugging, and refactoring. Some are fine-tuned on specific languages or frameworks.

Scientific Models: Domain-specific models trained on scientific literature. Useful for research applications but require careful evaluation for accuracy.

6.3 Model Configuration

Click on a model to configure:

Setting	Description
Enabled	Available for use
Min Tier	Minimum subscription tier
Rate Limits	Requests per minute
Max Tokens	Maximum context/output
Temperature Range	Allowed temperature values
Price Override	Custom pricing

Configuration Settings Explained

Enabled: When disabled, the model is hidden from users and API requests return "model not found". Use this to temporarily remove models during maintenance or to restrict access to specific models.

Min Tier: Sets the minimum subscription tier required to access this model. For example, setting GPT-4 to Tier 2 means Free tier users cannot use it. This helps control costs and create upgrade incentives.

Rate Limits: Controls requests per minute per user for this model. Prevents abuse and ensures fair access. Set based on the provider's rate limits and your capacity: - Conservative: 10-20 requests/minute - Standard: 50-100 requests/minute
- High: 200+ requests/minute (requires provider rate limit increases)

Max Tokens: Limits context window and output length. Useful for controlling costs since longer contexts cost more. Set based on use case: - Short tasks (Q&A): 4,096 tokens - Medium tasks (writing): 16,384 tokens - Long tasks (analysis): 32,768+ tokens

Temperature Range: Restricts the temperature parameter users can set. Temperature controls randomness: - 0.0: Deterministic, consistent outputs - 0.7: Balanced creativity and consistency - 1.0+: More creative, less predictable

Restricting range (e.g., 0.0-1.0) prevents users from setting extreme values that produce poor results.

Price Override: Allows custom pricing different from the default. Useful for: - Offering discounts on specific models - Increasing prices for premium models - Matching competitor pricing - A/B testing pricing strategies

6.4 Self-Hosted Models

For Tier 3+ deployments:

1. Navigate to **Models → Self-Hosted**
2. Click **"+ Add Self-Hosted Model"**
3. Configure:
   - **Model ID**: Unique identifier
   - **SageMaker Endpoint**: Endpoint name
   - **Instance Type**: ml.g5.xlarge, etc.
   - **Auto-Scaling**: Min/max instances
4. Deploy model to SageMaker

6.5 Thermal States (Self-Hosted)

State	Description	Response Time
HOT	Always running	<100ms
WARM	Scaled down	<5s
COLD	Stopped	30-60s
OFF	Disabled	N/A

---

7. Provider Management

7.1 External Providers

Navigate to **Providers** to manage API integrations:

Provider	Models	Status	Health
OpenAI	12	Configured	99.9%
Anthropic	6	Configured	99.8%
Google AI	8	Configured	99.7%
xAI	2	Configured	99.5%
DeepSeek	4	Not configured	-

7.2 Adding Provider Credentials

1. Click on provider name
2. Click **"Configure"**
3. Enter API credentials:
    - **API Key**: Provider API key
    - **Organization ID**: (if applicable)
    - **Base URL**: (for custom endpoints)
4. Click **"Test Connection"**
5. Click **"Save"**

7.3 Provider Health Monitoring

View real-time provider health:

```
Provider Health: OpenAI


Status:            Healthy
Uptime (30d):     99.94%
Avg Latency:      245ms
P95 Latency:      520ms
Error Rate:       0.02%

Last 24 Hours:

12am        6am        12pm        6pm        12am
```

7.4 Fallback Configuration

Configure provider fallbacks:

1. Navigate to **Providers** → **Fallbacks**
2. Set priority order for each model category
3. Configure automatic failover rules
4. Set retry policies

8. Billing & Subscriptions

8.1 Subscription Tiers

Tier	Name	Monthly	Features
1	Free	$0	Basic models, 1K requests
2	Starter	$29	More models, 10K requests
3	Professional	$99	All external models, 100K requests
4	Business	$299	Priority support, 500K requests
5	Enterprise	$999	Self-hosted, unlimited
6	Enterprise+	Custom	Custom SLAs, dedicated support
7	Ultimate	Custom	On-premise options

8.2 Credit System

Credits are the universal currency for AI usage:

Model Type	Cost per 1M Tokens
GPT-4o	500 credits
GPT-4o-mini	50 credits
Claude 3 Opus	600 credits
Claude 3 Sonnet	150 credits
Self-hosted	20 credits

8.3 Managing Subscriptions

Navigate to **Billing → Subscriptions**:

1. View current subscription
2. Upgrade/downgrade tier
3. Add credit packages
4. View invoices
5. Update payment method

8.4 Usage Reports

Generate usage reports:

1. Navigate to **Billing → Reports**
2. Select date range
3. Choose grouping (by tenant/model/user)
4. Export as CSV/PDF

8.5 Billing Alerts

Configure alerts for:

- Credit balance low
- Usage spike
- Approaching quota
- Failed payments

---

9. Storage Management

9.1 Storage Overview

Navigate to **Storage** to monitor:

```
Storage Overview


Total Used:       234.5 GB of 500 GB (47%)


By Type:
  Documents:     120.3 GB (51%)
  Images:        45.2 GB (19%)
  Audio:         38.7 GB (17%)
  Video:         22.1 GB (9%)
  Other:         8.2 GB (4%)

Top Tenants:
1. Acme Corp       45.2 GB
2. TechStart       32.1 GB
3. DataCo          28.4 GB
```

9.2 Storage Tiers

Tier	Included	Additional
Free	1 GB	N/A
Starter	10 GB	$0.10/GB
Professional	100 GB	$0.08/GB
Business	500 GB	$0.05/GB
Enterprise	2 TB	$0.03/GB

9.3 File Management

Manage uploaded files:

- View file metadata
- Download files
- Delete files
- Set retention policies

---

10. Orchestration & Neural Engine

10.1 Brain Router

The Brain Router automatically selects optimal models:

Factor	Weight	Description
Cost	30%	Price optimization
Quality	30%	Output quality
Speed	20%	Response latency
Availability	20%	Provider health

10.2 Neural Patterns

Configure orchestration patterns:

Pattern	Description	Use Case
Single	One model	Simple requests
Fallback	Primary + backup	High availability
Parallel	Multiple simultaneous	Consensus
Chain	Sequential models	Complex tasks

10.3 Workflow Templates

Create reusable workflows:

1. Navigate to **Orchestration** → **Workflows**
2. Click **"+ New Workflow"**
3. Define steps and conditions
4. Set triggers and parameters
5. Save and activate

---

11. Localization

11.1 Translation Management

Navigate to **Localization** to manage:

- Supported languages
- Translation strings
- AI translation settings

11.2 Supported Languages

Language	Code	Status
English	en	Default
Spanish	es	Enabled
French	fr	Enabled
German	de	Enabled
Japanese	ja	Enabled
Chinese	zh	Enabled

11.3 AI Translation

Enable AI-powered translation:

1. Navigate to **Localization → Settings**
2. Enable **"AI Translation"**
3. Select translation model
4. Configure quality settings

---

12. Configuration Management

12.1 System Configuration

Navigate to **Configuration** to manage:

Category	Settings
General	Platform name, domain, timezone
Email	SMTP settings, templates
Security	Password policy, MFA settings
API	Rate limits, CORS settings
Features	Feature flags

12.2 Tenant Overrides

Allow tenant-specific configuration:

1. Navigate to **Configuration → Tenant Overrides**
2. Select tenant
3. Override specific settings
4. Save changes

12.3 SSM Parameters

System configuration is stored in AWS SSM:

Parameter	Description
`/radiant/prod/database/url`	Database connection
`/radiant/prod/api/rate-limit`	API rate limits
`/radiant/prod/features/*`	Feature flags

---

13. Security & Compliance

13.1 Security Dashboard

Navigate to **Security** to monitor:

```
Security Dashboard                     Threat Level: Low


Active Threats:     0
Failed Logins:      23 (last 24h)
Suspicious IPs:     2 blocked
```

```
MFA Adoption:       94%

Recent Alerts:

   Unusual login location - user@acme.com (2h ago)
   Resolved: Brute force attempt blocked (5h ago)
   Resolved: API key rotated - tenant xyz (1d ago)
```

13.2 Anomaly Detection

Automatic detection of:

- Impossible travel (geographic anomalies)
- Session hijacking attempts
- Brute force attacks
- Unusual API patterns

13.3 Compliance Reports

Navigate to **Compliance** to generate:

Framework	Description
SOC 2	Service organization controls
HIPAA	Healthcare data protection
GDPR	EU data protection
ISO 27001	Information security

13.4 Generating Reports

1. Click **"Generate Report"**
2. Select framework
3. Choose date range
4. Select metrics to include
5. Generate PDF/CSV

---

14. Cost Analytics

14.1 Cost Dashboard

Navigate to **Cost** to view:

```
Cost Analytics                          Period: Last 30 Days


Total Spend:        $12,456.78          (+12% vs last month)
Projected:          $14,200.00          (this month)

By Provider:
    OpenAI:        $6,234.56 (50%)
    Anthropic:     $3,456.78 (28%)
    Self-hosted:   $1,234.56 (10%)
    Other:         $1,530.88 (12%)

  AI Recommendations:
   Switch 23% of GPT-4 calls to GPT-4-mini (save $890/mo)
   Enable caching for repeated queries (save $340/mo)
```

14.2 Cost Alerts

Configure alerts:

- Daily budget exceeded
- Weekly spend spike
- Per-tenant limits
- Per-model thresholds

14.3 Cost Optimization

Review AI-powered recommendations:

1. Navigate to **Cost → Insights**
2. Review suggestions
3. Click **"Apply"** to implement (requires approval)
4. Track savings over time

---

15. A/B Testing & Experiments

15.1 Experiment Dashboard

Navigate to **Experiments** to manage:

Experiment	Status	Variants	Sample Size
Model routing v2	Running	3	45,234
Prompt optimization	Running	2	12,456
Temperature test	Completed	4	89,123

15.2 Creating an Experiment

1. Click **"+ New Experiment"**
2. Configure:
   - **Name**: Descriptive name
   - **Hypothesis**: What you're testing
   - **Variants**: Control + treatments
   - **Traffic Split**: Percentage per variant
   - **Success Metric**: What to measure
3. Set targeting rules
4. Start experiment

15.3 Statistical Analysis

View results with:

- Conversion rates per variant
- Statistical significance (p-value)
- Confidence intervals
- Sample size recommendations

---

16. Audit & Monitoring

16.1 Audit Logs

Navigate to **Audit** to view all actions:

Column	Description
Timestamp	When action occurred
Actor	Who performed action
Action	What was done
Resource	What was affected
IP Address	Source IP
Details	Additional context

16.2 Log Filtering

Filter by:

- Date range
- Actor (user/admin)
- Action type
- Resource type
- Severity level

16.3 Log Export

Export logs for compliance:

1. Set filter criteria
2. Click **"Export"**
3. Choose format (CSV/JSON)
4. Download file

16.4 Real-Time Monitoring

Navigate to **Monitoring** for:

- Live request stream
- Error rate graphs
- Latency percentiles
- Active users count

---

17. Database Migrations

17.1 Migration Workflow

RADIANT uses dual-admin approval for production migrations:

1. **Submit**: Admin submits migration
2. **Review**: Second admin reviews
3. **Approve**: Second admin approves
4. **Execute**: Migration runs
5. **Verify**: Automatic verification

17.2 Pending Migrations

Navigate to **Migrations** to see:

```
Database Migrations


Pending Approval:

  #045 - Add user preferences table
  Submitted by: alice@company.com (2 hours ago)
  [View SQL]  [Approve]  [Reject]


Recent Migrations:
  #044 - Cost tracking tables (applied 2024-12-24)
  #043 - Experiment framework (applied 2024-12-20)
  #042 - Security anomalies (applied 2024-12-15)
```

17.3 Approving Migrations

1. Review the SQL in **"View SQL"**
2. Check for potential issues
3. Click **"Approve"** or **"Reject"**
4. Add comment explaining decision

---

18. API Management

18.1 API Keys

Manage platform API keys:

1. Navigate to **Settings → API Keys**
2. View existing keys
3. Create new keys with scopes
4. Revoke compromised keys

18.2 Rate Limiting

Configure rate limits:

Level	Default	Configurable
Global	10,000/min	Yes
Per-Tenant	1,000/min	Yes
Per-User	100/min	Yes
Per-Key	60/min	Yes

18.3 Webhooks

Configure outgoing webhooks:

1. Navigate to **Settings** → **Webhooks**
2. Add webhook URL
3. Select events to send
4. Test webhook
5. Enable webhook

---

19. Troubleshooting

19.1 Common Issues

High Error Rate

1. Check **Providers** for unhealthy providers
2. Review **Audit** logs for patterns
3. Check **Monitoring** for load spikes
4. Verify API key validity

Slow Response Times

1. Check provider latency in **Providers**
2. Review model selection in **Orchestration**
3. Check for cold-start issues (self-hosted)
4. Verify database performance

Authentication Failures

1. Check user status in **Users**
2. Verify MFA configuration
3. Review **Audit** logs for login attempts
4. Check for IP blocks in **Security**

19.2 Support Resources

Resource	Description
Documentation	This guide + online docs
Status Page	status.radiant.example.com
Support Email	support@radiant.example.com
Emergency	+1-555-RADIANT

19.3 Log Locations

Service	Log Group
API Gateway	/aws/apigateway/radiant
Lambda	/aws/lambda/radiant-*
Admin Dashboard	/aws/cloudfront/admin
Database	/aws/rds/cluster/radiant

---

20. Delight System Administration

The Delight System provides personality, achievements, and engagement features for Think Tank users.

20.1 Accessing Delight Admin

Navigate to **Think Tank → Delight** in the admin sidebar.

20.2 Dashboard Overview

The Delight dashboard shows:

Metric	Description
Messages Shown	Total delight messages displayed
Achievements Unlocked	Total achievements earned by users
Easter Eggs Found	Hidden features discovered
Active Users	Users with Delight enabled

20.3 Managing Categories

Toggle entire categories on/off:

Category	Purpose
Domain Loading	Messages while loading domain expertise
Domain Transition	Messages when switching topics
Time Awareness	Time-of-day contextual messages
Model Dynamics	Messages about AI consensus/disagreement
Complexity Signals	Feedback on query complexity
Synthesis Quality	Post-response quality indicators
Achievements	Milestone celebrations
Wellbeing	Break/health reminders
Easter Eggs	Hidden features
Sounds	Audio feedback

20.4 Managing Messages

- **Create**: Add new delight messages with targeting options
- **Edit**: Modify text, triggers, and display settings
- **Delete**: Remove messages (soft delete)
- **Toggle**: Enable/disable individual messages

Message Targeting Options

Option	Values
Injection Point	pre_execution, during_execution, post_execution
Trigger Type	domain_loading, time_aware, model_dynamics, etc.
Domain Families	science, humanities, creative, technical, etc.
Time Contexts	morning, afternoon, evening, night, weekend
Display Style	subtle, moderate, expressive

20.5 Statistics Dashboard

Access detailed usage statistics at **Delight → View Statistics**:

- **Weekly Trends**: 12-week activity history
- **Top Messages**: Most-shown messages with engagement data
- **Achievement Stats**: Unlock rates, time-to-unlock averages
- **Easter Egg Stats**: Discovery rates by egg
- **User Engagement**: Leaderboard by achievement points

20.6 Managing Achievements

Configure achievement unlock criteria:

Setting	Description
Threshold	Number required to unlock
Rarity	common, uncommon, rare, epic, legendary
Points	Score value for leaderboards
Hidden	Only visible after unlock

20.7 Managing Easter Eggs

Configure hidden features:

Setting	Description
Trigger Type	key_sequence, text_input, time_based, random
Trigger Value	The activation pattern
Effect Type	mode_change, visual_transform, sound_play
Duration	How long the effect lasts

20.8 API Endpoints

Endpoint	Method	Description
`/api/admin/delight/dashboard`	GET	Dashboard data
`/api/admin/delight/statistics`	GET	Detailed statistics
`/api/admin/delight/categories/:id`	PATCH	Toggle category
`/api/admin/delight/messages`	POST	Create message
`/api/admin/delight/messages/:id`	PUT/DELETE	Update/delete
`/api/admin/delight/user-engagement`	GET	User leaderboard

Appendix: Quick Reference

Keyboard Shortcuts

Shortcut	Action
`G + D`	Go to Dashboard
`G + T`	Go to Tenants
`G + M`	Go to Models
`G + B`	Go to Billing
`G + A`	Go to Audit
`?`	Show shortcuts

Status Indicators

Icon	Meaning
	Healthy/Success
	Warning
	Error/Failed
	In Progress
	Disabled/Pending

Document Version: 4.18.1 Last Updated: December 2024

RADIANT Deployer - Administrator Guide

Complete guide for deploying and managing RADIANT infrastructure using the Swift Deployer App

Version: 4.18.1 | Last Updated: December 2024

Table of Contents

---

1. Introduction

1.1 What is RADIANT Deployer?

RADIANT Deployer is a native macOS application that provides a complete deployment management solution for the RADIANT platform. It offers:

- **One-Click Deployments**: Deploy entire infrastructure stacks with a single click
- **AI-Powered Assistance**: Claude-powered assistant for deployment guidance
- **Snapshot Management**: Create and restore deployment snapshots
- **Multi-Region Support**: Deploy across multiple AWS regions
- **Health Monitoring**: Real-time health checks and status monitoring
- **Secure Credential Storage**: Keychain-integrated credential management

1.2 Architecture

```
                    RADIANT Deployer App


      Dashboard           Deployments          Settings
        View                 View                View



                         Services Layer

         AWS              CDK              AI Assistant
        Service          Service            Service


        Snapshot         Health           Local Storage
        Service          Check             Manager




                       Storage Layer

         Keychain        SQLCipher         File System
        (Secrets)        (Local DB)        (Snapshots)
```

1.3 Key Features

Feature	Description
Deployment Wizard	Step-by-step guided deployment process
Lock-Step Mode	Ensures component version consistency
Automatic Rollback	Reverts failed deployments automatically
Offline Mode	Core functionality works without internet
Audit Logging	Complete deployment history tracking

Deployment Wizard Explained

The Deployment Wizard breaks down the complex AWS infrastructure deployment into manageable steps. Instead of manually running CDK commands and configuring services, the wizard:

1. **Validates Prerequisites**: Checks that all required software (Node.js, AWS CLI, CDK) is installed and properly configured before starting
2. **Guides Configuration**: Walks you through selecting environment (dev/staging/prod), tier (1-5), and region settings with explanations for each option
3. **Shows Progress**: Displays real-time progress as each CloudFormation stack deploys, with estimated time remaining
4. **Handles Errors**: If any step fails, provides clear error messages and suggested remediation steps

Lock-Step Mode Explained

Lock-Step Mode prevents version mismatches between RADIANT components that could cause compatibility issues:

- **What it does**: Ensures the Admin Dashboard, Lambda functions, and database schema are all on compatible versions
- **Why it matters**: A v4.18 Lambda function might expect database columns that don't exist in a v4.17 schema, causing runtime errors
- **How it works**: Before deployment, the system checks version numbers across all components and blocks deployment if drift exceeds the configured maximum (default: 1 minor version)
- **When to disable**: Only disable during development when testing individual component changes

Automatic Rollback Explained

Automatic Rollback protects your production environment by reverting failed deployments:

- **Trigger conditions**: Activates when any deployment phase fails (CDK deploy error, health check failure, migration error)
- **Rollback process**: Restores the pre-deployment snapshot, which includes database state, Lambda code, and configuration
- **Recovery time**: Typically 5-10 minutes depending on the size of changes being reverted
- **Notification**: Sends alerts via configured channels (email, Slack) when rollback occurs

Offline Mode Explained

Offline Mode allows essential operations when internet connectivity is unavailable:

- **Available offline**: Viewing deployment history, browsing local snapshots, reviewing configuration, accessing cached documentation
- **Requires internet**: Deploying to AWS, health checks, AI assistant queries, credential validation
- **Data sync**: When connectivity returns, local changes sync automatically with the cloud state

Audit Logging Explained

Every action in the Deployer is logged for compliance and troubleshooting:

- **What's logged**: User identity, timestamp, action type, parameters, outcome, duration
- **Storage**: Logs stored locally in SQLCipher database and optionally synced to CloudWatch
- **Retention**: Local logs kept for 90 days by default (configurable)
- **Export**: Logs can be exported to CSV/JSON for compliance audits

---

2. System Requirements

2.1 Hardware Requirements

Component	Minimum	Recommended
macOS Version	13.0 (Ventura)	14.0+ (Sonoma)
Processor	Apple Silicon or Intel	Apple Silicon M1+
Memory	8 GB RAM	16 GB RAM
Storage	2 GB free	10 GB free
Display	1280x800	1440x900+

Why These Requirements?

macOS 13.0+: Required for Swift 5.9 runtime and modern SwiftUI features. Older versions lack required system APIs for secure keychain access and modern networking.

Apple Silicon Recommended: While Intel Macs are supported, Apple Silicon provides 2-3x faster CDK synthesis and compilation times. The app is built as a universal binary supporting both architectures.

8 GB RAM Minimum: CDK synthesis loads the entire infrastructure definition into memory. Complex deployments (Tier 3+) with many resources may require more memory. With 8 GB, you may experience slowdowns during synthesis.

16 GB RAM Recommended: Allows comfortable multitasking while deployments run in the background. Essential if you're also running Docker, IDEs, or other development tools.

2 GB Storage Minimum: Covers the application itself (~200 MB), local database (~50 MB), and several snapshots. However, snapshots can grow large over time.

10 GB Storage Recommended: Provides room for multiple deployment snapshots, comprehensive logs, and CDK cache. Each full snapshot can be 100-500 MB depending on your deployment size.

2.2 Software Requirements

Software	Version	Purpose	Installation
Xcode	15.0+	Swift runtime (Command Line Tools sufficient)	`xcode-select --install`
AWS CLI	2.x	AWS operations	`brew install awscli`
Node.js	20.x LTS	CDK operations	`brew install node@20`
AWS CDK	2.120+	Infrastructure deployment	`npm install -g aws-cdk`
pnpm	8.x+	Package management	`npm install -g pnpm`

Software Explained

Xcode Command Line Tools: Provides the Swift runtime and compiler. You don't need the full Xcode IDE - just the command line tools (4 GB vs 12 GB download). The Deployer uses Swift for its native performance and seamless macOS integration.

AWS CLI v2: The official AWS command-line interface. Used internally by the Deployer to execute AWS operations, assume IAM roles, and query service status. Version 2 is required for SSO support and improved credential handling.

Node.js 20 LTS: Required to run the AWS CDK, which is written in TypeScript. LTS (Long Term Support) versions receive security updates for 30 months. The Deployer manages Node.js processes internally during CDK operations.

AWS CDK 2.120+: The Cloud Development Kit that defines RADIANT's infrastructure as TypeScript code. Version 2.120+ includes critical bug fixes for Aurora PostgreSQL and Lambda layer handling. The CDK synthesizes your infrastructure into CloudFormation templates.

pnpm 8.x: A fast, disk-efficient package manager used to install CDK dependencies. Chosen over npm for its superior handling of monorepo workspaces and 2-3x faster installation times.

2.3 AWS Requirements

Requirement	Details	How to Verify
AWS Account	Active account with billing enabled	Check AWS Console billing page
IAM User	AdministratorAccess or equivalent	`aws sts get-caller-identity`
Regions	Access to us-east-1 (required) + additional regions	`aws ec2 describe-regions`
Service Quotas	Default quotas sufficient for Tier 1-2	AWS Service Quotas console

AWS Requirements Explained

Active AWS Account with Billing: RADIANT deploys real AWS resources that incur costs. Billing must be enabled and a valid payment method on file. New accounts have a $5 pending charge verification. Free tier covers some resources for 12 months but won't cover all RADIANT components.

IAM User with AdministratorAccess: The Deployer needs broad permissions to create and manage resources across many AWS services. For production, you can use a scoped-down policy (see Appendix A), but AdministratorAccess is recommended for initial setup to avoid permission errors.

Required Permissions Include: - CloudFormation (stack operations) - Lambda (function management) - API Gateway (REST API setup) - RDS/Aurora (database provisioning) - Cognito (user pool management) - S3 (storage buckets) - IAM (role creation) - SSM (parameter storage) - Secrets Manager (credential storage) - CloudWatch (logging and monitoring)

us-east-1 Required: This region is required even if you deploy to other regions because: - ACM certificates for CloudFront must be in us-east-1 - Some global services (IAM, Route 53) operate from us-east-1 - CDK bootstrap resources are region-specific

Service Quotas: Default AWS quotas are sufficient for Tier 1-2 deployments. Tier 3+ may require quota increases for: - Lambda concurrent executions (default: 1,000) - RDS instances (default: 40) - VPC Elastic IPs (default: 5)

To request quota increases: AWS Console > Service Quotas > Select service > Request increase

---

3. Installation

3.1 Download and Install

Option A: Pre-built Application (Recommended)

1. Download the latest release from GitHub Releases
2. Drag `RadiantDeployer.app` to `/Applications`
3. Right-click and select "Open" (first launch only)
4. Grant necessary permissions when prompted

Option B: Build from Source

```
# Clone the repository
git clone https://github.com/your-org/radiant.git
cd radiant/apps/swift-deployer

# Build the application
```

```
swift build -c release

# Run the application
swift run RadiantDeployer
```

3.2 Initial Permissions

The app requires the following permissions:

Permission	Purpose	How to Grant
Keychain Access	Store AWS credentials securely	Approve on first credential save
Network Access	Connect to AWS and AI services	Approve in System Settings
File Access	Save snapshots and logs	Approve when prompted

3.3 Verify Installation

1. Launch RadiantDeployer
2. Navigate to **Settings** → **About**
3. Verify version shows `4.18.1`
4. Check all services show green status

---

4. First-Time Setup

4.1 Setup Wizard

On first launch, the Setup Wizard guides you through:

```
                    Setup Wizard


  Step 1: Welcome                    Complete
  Step 2: AWS Credentials            In Progress
  Step 3: Environment Configuration  Pending
  Step 4: AI Assistant Setup         Pending
  Step 5: Verification               Pending
```

4.2 AWS Credentials Setup

1. Click **"Add AWS Credentials"**
2. Enter your credentials:
    - **Name**: Descriptive name (e.g., "Production Account")
    - **Access Key ID**: `AKIA...` (20 characters)
    - **Secret Access Key**: Your secret key (40 characters)
    - **Region**: Primary region (e.g., `us-east-1`)
3. Click **"Validate"** to test connectivity
4. Click **"Save"** to store securely in Keychain

4.3 Environment Configuration

Configure your deployment environment:

Setting	Description	Default
Environment	`dev`, `staging`, or `prod`	`dev`
Tier	Infrastructure tier (1-5)	`1`
Domain	Your domain name	Required for Tier 2+
Stack Prefix	CDK stack name prefix	`radiant`

Environment Types Explained

Development (dev): For active development and testing. Features relaxed security settings, smaller instance sizes, and no deletion protection. Data can be reset freely. Cost: ~$50-150/month for Tier 1.

Staging (staging): Pre-production environment that mirrors production configuration. Use this to test deployments before going live. Same security as production but can use smaller instances. Cost: ~60% of production.

Production (prod): Live environment serving real users. Includes deletion protection, multi-AZ deployments, automated backups, and enhanced monitoring. Never deploy untested changes directly to production.

Infrastructure Tiers Explained

Tier	Name	Monthly Cost	Use Case	Resources
1	SEED	$50-150	Development, POC	Single-AZ, t3.small instances, 20GB storage
2	STARTUP	$200-400	Small production	Multi-AZ database, WAF, basic monitoring
3	GROWTH	$1,000-2,500	Medium production	Self-hosted models, HIPAA compliance, enhanced security
4	SCALE	$4,000-8,000	Large production	Multi-region, global database, dedicated instances
5	ENTERPRISE	$15,000-35,000	Enterprise	Custom SLAs, dedicated support, on-premise options

Choosing the Right Tier: Start with Tier 1 for development. Move to Tier 2 when you have paying customers. Tier 3+ is for organizations with compliance requirements or high traffic.

Domain Configuration

For Tier 2+, you need a custom domain:

1. **Register a domain** in Route 53 or transfer an existing domain
2. **Create a hosted zone** in Route 53 for your domain
3. **Request an ACM certificate** in us-east-1 for `*.yourdomain.com`
4. **Validate the certificate** via DNS (automatic if using Route 53)

The Deployer will configure: - `api.yourdomain.com` - API Gateway endpoint - `admin.yourdomain.com` - Admin Dashboard - `app.yourdomain.com` - Think Tank application

4.4 AI Assistant Setup (Optional)

Enable the Claude-powered AI assistant:

1. Navigate to **Settings → AI Assistant**
2. Enter your Anthropic API key
3. Select response style:
   - **Concise**: Brief, action-focused responses
   - **Detailed**: In-depth explanations
   - **Tutorial**: Step-by-step guidance
4. Test the connection with a sample query

---

5. AWS Credentials Management

5.1 Credential Sets

Manage multiple AWS accounts:

Field	Description	Example
Name	Friendly identifier	"Production"
Access Key ID	AWS access key	`AKIAIOSFODNN7EXAMPLE`
Secret Access Key	AWS secret	(stored encrypted)
Region	Default region	`us-east-1`
Role ARN	Optional assume role	`arn:aws:iam::123:role/deploy`

5.2 Adding Credentials

1. Navigate to **Credentials** tab
2. Click **"+ Add Credential Set"**
3. Fill in the required fields
4. Click **"Validate"** to test
5. Click **"Save"**

5.3 Credential Validation

The app validates:

- Access key format (AKIA prefix, 20 chars)
- Secret key length (40+ chars)
- Region validity
- AWS connectivity (STS GetCallerIdentity)
- Required permissions

5.4 Security Best Practices

Practice	Recommendation
Rotate Keys	Every 90 days
Least Privilege	Use scoped IAM policies
MFA	Enable on AWS account
Audit	Review access logs regularly
Backup	Export credentials securely

5.5 Importing from AWS CLI

```
# The app can import from ~/.aws/credentials
# Navigate to Credentials → Import from AWS CLI
```

---

6. Deployment Operations

6.1 Deployment Dashboard

The main dashboard shows:

```
Environment: dev          Status:  Healthy
Version: 4.18.1           Last Deploy: 2024-12-25 10:30:00



   Deploy           Rollback          Settings
   [Button]         [Button]          [Button]


Recent Deployments:

  2024-12-25 10:30   v4.18.1   prod    Success   4m 32s
  2024-12-24 15:45   v4.18.0   prod    Success   5m 12s
  2024-12-24 09:00   v4.17.0   dev     Success   3m 45s
```

6.2 Starting a Deployment

1. Select target **Environment** (dev/staging/prod)
2. Select **Tier** (1-5)
3. Review deployment plan
4. Click **"Start Deployment"**
5. Monitor progress in real-time

6.3 Deployment Phases

Phase	Duration	Description
1. Validation	~30s	Credential and configuration check
2. Snapshot	~1m	Create pre-deployment backup
3. CDK Synth	~1m	Generate CloudFormation templates
4. CDK Deploy	~10-20m	Deploy infrastructure
5. Migration	~2m	Run database migrations
6. Health Check	~1m	Verify all services
7. Cleanup	~30s	Remove temporary resources

Phase Details

Phase 1 - Validation (30 seconds)

Before any changes are made, the system validates: - AWS credentials are valid and not expired - IAM permissions are sufficient for all required operations - Target environment exists or can be created - No conflicting deployments are in progress (deployment lock check) - Required software versions are installed (Node.js, CDK, etc.) - Network connectivity to AWS services

If validation fails, you'll see specific error messages explaining what needs to be fixed.

Phase 2 - Snapshot (1 minute)

A pre-deployment snapshot captures the current state so you can rollback if needed: - Database schema and critical data (not full data backup) - Current Lambda function code versions - SSM Parameter Store values - Current CloudFormation stack states - Configuration files

Snapshots are stored locally and can be managed in the Snapshots tab.

Phase 3 - CDK Synth (1 minute)

The CDK synthesizes TypeScript infrastructure code into CloudFormation templates: - Reads infrastructure definitions from `packages/infrastructure/` - Resolves all construct dependencies - Generates CloudFormation JSON/YAML templates - Calculates resource changes (what will be created/updated/deleted) - Validates templates against AWS CloudFormation rules

You can review the generated templates before proceeding.

Phase 4 - CDK Deploy (10-20 minutes)

The actual AWS resource deployment happens in this phase: - CloudFormation stacks are created or updated in dependency order - Resources are provisioned (databases, Lambda functions, API Gateway, etc.) - IAM roles and policies are configured - Networking (VPC, subnets, security groups) is set up - DNS records are created/updated

This is the longest phase. Progress shows which stack is currently deploying.

Phase 5 - Migration (2 minutes)

Database migrations ensure your schema matches the deployed code: - Connects to Aurora PostgreSQL using Data API - Runs pending migration files in sequence - Creates new tables, columns, indexes as needed - Updates RLS (Row-Level Security) policies - Verifies migration success with integrity checks

Migrations are idempotent - running them twice won't cause issues.

Phase 6 - Health Check (1 minute)

Verifies all deployed services are functioning: - API Gateway responds to health endpoint - Lambda functions can be invoked - Database connections succeed - Cognito user pools are accessible - S3 buckets are reachable - CloudFront distributions are deployed

If health checks fail, automatic rollback is triggered (if enabled).

Phase 7 - Cleanup (30 seconds)

Final cleanup tasks: - Removes temporary files created during deployment - Clears CDK staging buckets of old assets - Updates deployment history in local database - Releases deployment lock - Sends completion notification

6.4 Deployment Progress

```
Deploying to Production


[                ] 65%
```

```
Current Phase: CDK Deploy
Stack: Radiant-prod-API (5 of 9)
Elapsed: 8m 23s | Estimated: 4m remaining

  Validation complete
  Snapshot created: snap-20241225-103000
  CDK synthesis complete
  Deploying stacks...
    Radiant-prod-Foundation
    Radiant-prod-Networking
    Radiant-prod-Security
    Radiant-prod-Data
    Radiant-prod-API
    Radiant-prod-Auth
    Radiant-prod-AI
    Radiant-prod-Admin
    Radiant-prod-Monitoring

[Cancel Deployment]
```

6.5 Deployment Settings

Configure deployment behavior:

Setting	Description	Default
Auto-Rollback	Rollback on failure	Enabled
Lock-Step Mode	Require version consistency	Enabled
Max Version Drift	Maximum version difference	1
Approval Required	Require confirmation for prod	Enabled
Notification	Send completion notifications	Enabled

6.6 Operation Timeouts

Operation	Default Timeout	Configurable
CDK Deploy	30 minutes	Yes
Health Check	5 minutes	Yes
Migration	10 minutes	Yes
Snapshot	5 minutes	Yes
Rollback	15 minutes	Yes

---

7. Multi-Region Deployments

7.1 Overview

Deploy RADIANT across multiple AWS regions for:

- **High Availability**: Survive regional outages
- **Low Latency**: Serve users from nearest region
- **Compliance**: Data residency requirements

7.2 Supported Regions

Region	Code	Primary Use
US East (N. Virginia)	`us-east-1`	Primary (required)
US West (Oregon)	`us-west-2`	West coast users
EU (Ireland)	`eu-west-1`	European users
EU (Frankfurt)	`eu-central-1`	GDPR compliance
Asia Pacific (Singapore)	`ap-southeast-1`	APAC users
Asia Pacific (Tokyo)	`ap-northeast-1`	Japanese users

7.3 Adding a Region

1. Navigate to **Multi-Region** tab
2. Click **"Add Region"**
3. Configure:
    - **Region**: Select from available regions
    - **Is Primary**: Set primary region flag
    - **Stack Prefix**: Region-specific prefix
    - **Endpoint**: Custom domain for region
4. Click **"Deploy to Region"**

7.4 Region Consistency

Monitor version consistency across regions:

```
Multi-Region Status                    Consistency:  100%


Region            Version    Status    Last Deploy

us-east-1 (P)     4.18.1      Healthy  2024-12-25 10:30
eu-west-1         4.18.1      Healthy  2024-12-25 10:35
ap-southeast-1    4.18.1      Healthy  2024-12-25 10:40

[Deploy All]  [Check Consistency]  [Sync Versions]
```

---

8. Snapshots & Rollbacks

8.1 Snapshot Types

Type	Description	Retention
Pre-Deploy	Automatic before each deployment	30 days
Manual	User-initiated backup	Until deleted
Scheduled	Periodic backups	Configurable

8.2 Creating a Snapshot

1. Navigate to **Snapshots** tab

2. Click **"Create Snapshot"**
3. Enter description (optional)
4. Select components to include:
    - Database state
    - Configuration
    - Lambda code
    - Infrastructure state
5. Click **"Create"**

8.3 Snapshot Contents

```
Snapshot: snap-20241225-103000
  metadata.json          # Snapshot info
  database/
     schema.sql          # Database schema
     data.sql            # Critical data
  config/
     ssm-parameters.json # SSM parameters
     secrets.json        # Secret references
  lambda/
     functions.zip       # Lambda code packages
  infrastructure/
     state.json          # CDK state
```

8.4 Restoring from Snapshot

1. Navigate to **Snapshots** tab
2. Select snapshot to restore
3. Click **"Restore"**
4. Confirm restoration scope:
    - Full restoration
    - Database only
    - Configuration only
5. Click **"Confirm Restore"**

8.5 Automatic Rollback

When enabled, failed deployments automatically:

1. Stop deployment process
2. Identify last known good state
3. Restore from pre-deploy snapshot
4. Verify system health
5. Send notification

---

9. AI Assistant

9.1 Overview

The Claude-powered AI Assistant provides:

- **Deployment Guidance**: Step-by-step help
- **Error Translation**: User-friendly error explanations
- **Troubleshooting**: Problem diagnosis

- **Best Practices**: Recommendations

9.2 Enabling AI Assistant

1. Navigate to **Settings → AI Assistant**
2. Enter Anthropic API key
3. Toggle **“Enable AI Assistant”**
4. Select response style

9.3 Using the Assistant

Access the assistant via:

- **Chat Panel**: Click AI icon in toolbar
- **Contextual Help**: Click “?” on any screen
- **Error Dialogs**: Click “Explain” on errors

9.4 Sample Interactions

```
AI Assistant


You: What does error "AccessDenied: User is not authorized
     to perform dynamodb:CreateTable" mean?

Assistant: This error indicates your AWS credentials don't
have permission to create DynamoDB tables. To fix this:

1. Check your IAM user/role has the required permissions
2. Add the "AmazonDynamoDBFullAccess" managed policy, or
3. Add specific permissions:
   - dynamodb:CreateTable
   - dynamodb:DescribeTable
   - dynamodb:UpdateTable

Would you like me to show the exact IAM policy needed?

[Yes, show policy]  [No, I'll figure it out]
```

9.5 Offline Mode

When offline, the assistant provides:

- Pre-cached common error explanations
- Local troubleshooting guides
- Fallback recommendations

---

10. Package Management

10.1 Package System

RADIANT uses atomic packages for deployment:

```
radiant-4.18.1.pkg
  manifest.json          # Package manifest
  checksums.sha256       # Component checksums
  radiant/               # Radiant components
     infrastructure/
     lambda/
     dashboard/
  thinktank/             # Think Tank components
     api/
     frontend/
```

10.2 Viewing Packages

Navigate to **Packages** tab to see:

- Installed packages
- Available updates
- Package history
- Component versions

10.3 Version Management

Version Type	Format	Example
Radiant	Major.Minor.Patch	4.18.1
Think Tank	Major.Minor.Patch	3.2.0
Package	Combined	4.18.1+3.2.0

10.4 Lock-Step Mode

When enabled:

- All components must have same minor version
- Maximum version drift configurable
- Automatic sync available

---

11. Monitoring & Health Checks

11.1 Health Dashboard

```
  System Health                          Overall:  Healthy


  Service               Status    Latency    Last Check

  API Gateway             Up      45ms        10s ago
  Lambda (Router)         Up      120ms       10s ago
  Aurora PostgreSQL      Up      12ms        10s ago
  DynamoDB               Up      8ms         10s ago
  Cognito                Up      85ms        10s ago
  S3 Storage             Up      35ms        10s ago
  CloudFront             Up      22ms        10s ago
  SageMaker (if T3+)     Up      250ms       10s ago

  [Refresh]  [Run Full Check]  [View History]
```

11.2 Health Check Types

Check	Frequency	Timeout
Quick	Every 60s	5s
Standard	Every 5m	30s
Deep	Manual/Deploy	2m

11.3 Alerts

Configure alerts for:

- Service degradation
- High latency
- Error rate spikes
- Failed deployments

---

12. Security Features

12.1 Credential Security

Feature	Implementation
Storage	macOS Keychain (encrypted)
Memory	Cleared after use
Transport	TLS 1.3 only
Validation	Format + connectivity check

12.2 Deployment Locks

Prevent concurrent deployments:

```
Deployment Lock: Active
  Acquired: 2024-12-25 10:30:00
  Owner: deployer@example.com
  Environment: production
  Expires: 2024-12-25 11:30:00
```

12.3 Audit Logging

All operations are logged:

```
{
  "timestamp": "2024-12-25T10:30:00Z",
  "operation": "deployment.start",
  "user": "admin@example.com",
  "environment": "production",
  "version": "4.18.1",
  "status": "success",
  "duration_ms": 272000
}
```

12.4 Secret Detection

Pre-commit checks scan for:

- AWS access keys
- API keys
- Passwords
- Private keys

---

13. Troubleshooting

13.1 Common Issues

Deployment Fails at CDK Synth

Symptoms: Deployment stops at synthesis phase

Solutions: 1. Check Node.js version: `node --version` (need 20.x) 2. Clear CDK cache: `rm -rf cdk.out` 3. Update CDK: `npm update -g aws-cdk` 4. Check TypeScript errors in `packages/infrastructure`

AWS Credentials Invalid

Symptoms: "Invalid credentials" error

Solutions: 1. Verify access key format (starts with AKIA) 2. Check secret key hasn't expired 3. Verify IAM user is active 4. Test with AWS CLI: `aws sts get-caller-identity`

Health Check Timeout

Symptoms: Services show unhealthy after deployment

Solutions: 1. Wait 2-3 minutes for cold start 2. Check CloudWatch logs for errors 3. Verify security group rules 4. Check VPC endpoint configuration

13.2 Log Locations

Log Type	Location
App Logs	`~/Library/Logs/RadiantDeployer/`
Deployment Logs	`~/Library/Application Support/RadiantDeployer/deployments/`
AWS Logs	CloudWatch Log Groups

13.3 Getting Help

1. **AI Assistant**: Built-in help
2. **Documentation**: This guide + online docs
3. **Support**: support@radiant.example.com

---

14. Reference

14.1 Keyboard Shortcuts

Shortcut	Action
`+ D`	Start deployment
`+ R`	Refresh status
`+ S`	Create snapshot
`+ ,`	Open settings
`+ ?`	Open AI assistant
`+ L`	View logs

14.2 CLI Commands

```
# Build and run from source
cd apps/swift-deployer
swift build -c release
swift run RadiantDeployer

# Run with specific config
swift run RadiantDeployer --environment prod --tier 3

# Headless deployment
swift run RadiantDeployer deploy --non-interactive
```

14.3 Environment Variables

Variable	Description
`RADIANT_ENV`	Override environment
`RADIANT_TIER`	Override tier
`RADIANT_DEBUG`	Enable debug logging
`RADIANT_AI_KEY`	Anthropic API key

14.4 File Locations

File	Location
Configuration	`~/Library/Application Support/RadiantDeployer/config.json`
Snapshots	`~/Library/Application Support/RadiantDeployer/snapshots/`
Logs	`~/Library/Logs/RadiantDeployer/`
Database	`~/Library/Application Support/RadiantDeployer/local.db`

Appendix A: IAM Policy Requirements

Minimum IAM permissions for deployment:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "s3:*",
        "lambda:*",
        "apigateway:*",
        "cognito-idp:*",
        "rds:*",
        "dynamodb:*",
        "sqs:*",
        "sns:*",
        "events:*",
        "logs:*",
        "iam:PassRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "ssm:*",
        "secretsmanager:*",
        "ecr:*",
        "ecs:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Appendix B: Glossary

Term	Definition
CDK	AWS Cloud Development Kit
Stack	CloudFormation stack deployed by CDK
Snapshot	Point-in-time backup of deployment
Lock-Step	Version consistency enforcement
Tier	Infrastructure sizing level (1-5)

Document Version: 4.18.1 Last Updated: December 2024

RADIANT Deployer Architecture & Deployment Packages

Technical Architecture Document

Version: 4.18.1 | Last Updated: December 2024

Overview

The RadiantDeployer Swift app operates in three distinct modes, each with different behaviors for parameter handling, package selection, and database operations.

Deployment Modes

Mode Definitions

```
                        DEPLOYER OPERATIONAL MODES



      INSTALL              UPDATE              ROLLBACK
     (Fresh)             (Upgrade)            (Revert)




    Use Default         Read Current        Read Target
    Parameters          From Instance       Snapshot




    Seed AI             Merge User          Restore
    Registry            Changes             Previous




    Create New          Apply Delta         Apply
    Instance            Changes             Snapshot
```

INSTALL Mode (Fresh Installation)

Trigger: No existing deployment detected for app/environment combination

Key Behaviors: 1. Uses DEFAULT parameters from tier configuration 2. Runs ALL database migrations (fresh) 3. SEEDS the AI Registry with providers and models 4. Creates initial admin user 5. Stores deployment metadata

Parameter Source: `InstallationParameters.defaults()`

```
// Parameters are initialized with tier-appropriate defaults
let parameters = InstallationParameters.defaults(
    appId: app.id,
    environment: environment,
    tier: .growth  // Based on selected tier
)
```

UPDATE Mode (Upgrade Existing)

Trigger: Existing deployment detected AND target version >= current version

Key Behaviors: 1. Fetches current parameters FROM the running instance 2. Creates pre-update snapshot for rollback 3. MERGES user changes with current parameters 4. Validates parameter changes are safe 5. Runs INCREMENTAL migrations only 6. **DOES NOT** seed AI Registry (preserves admin customizations)

Parameter Source: Running instance API + user modifications

```
// Parameters fetched from instance, then merged with user changes
let currentParameters = await fetchCurrentParameters(app, environment, credentials)
let updatedParameters = mergeParameters(current: currentParameters, changes: userChanges)
```

ROLLBACK Mode (Revert to Previous)

Trigger: User explicitly requests rollback OR target version < current version

Key Behaviors: 1. Loads target snapshot from S3 2. Creates safety snapshot of current state 3. Deploys with SNAPSHOT parameters (not current, not defaults) 4. Optionally restores database from RDS snapshot 5. Does not modify AI Registry

Parameter Source: Selected snapshot

Deployment Package Structure

Deployment packages are self-contained, versioned bundles containing everything needed to deploy a specific version of RADIANT.

```
radiant-4.18.0-abc123.radpkg
   manifest.json              # Package metadata & verification
   checksums.sha256           # File integrity verification

   infrastructure/            # CDK Stacks (compiled)
      cdk.out/               # Synthesized CloudFormation
      lib/                   # CDK TypeScript (compiled)
```

```
    cdk.json               # CDK configuration

migrations/                # Database migrations
    radiant/               # Core schema migrations
    thinktank/             # Think Tank specific
    seeds/                 # Seed data (AI Registry, etc.)

functions/                 # Lambda function code
    api/                   # API handlers
    admin/                 # Admin handlers
    billing/               # Billing handlers
    thermal/               # Thermal management

admin-dashboard/           # Next.js admin dashboard
    .next/                 # Compiled Next.js

config/                    # Default configurations
    defaults.json          # Default parameters per tier
    providers.json         # AI provider seed data
    models.json            # AI model seed data
```

Package Manifest

```json
{
  "packageFormat": "radpkg-v1",
  "version": "4.18.0",
  "buildId": "abc123def456",
  "buildTimestamp": "2024-12-24T10:30:00Z",

  "components": {
    "radiantPlatform": {
      "version": "4.18.0",
      "minUpgradeFrom": "4.15.0"
    },
    "thinkTank": {
      "version": "3.2.0",
      "minUpgradeFrom": "3.0.0"
    }
  },

  "compatibility": {
    "minimumDeployerVersion": "4.16.0",
    "supportedTiers": ["SEED", "STARTER", "GROWTH", "SCALE", "ENTERPRISE"],
    "supportedRegions": ["us-east-1", "us-west-2", "eu-west-1", "ap-southeast-1"]
  },

  "installBehavior": {
    "seedAIRegistry": true,
    "createInitialAdmin": true,
    "runFullMigrations": true
  },

  "updateBehavior": {
    "seedAIRegistry": false,
    "preserveAdminCustomizations": true,
    "runIncrementalMigrations": true,
    "createPreUpdateSnapshot": true
  }
}
```

Package Storage Locations

```
1. DEPLOYER APP CACHE (Local)
   ~/Library/Application Support/RadiantDeployer/packages/
     radiant-4.18.0-abc123.radpkg
     radiant-4.17.0-def456.radpkg
     index.json

2. S3 RELEASE BUCKET (Cloud - Official Releases)
   s3://radiant-releases-{region}/
     stable/
        radiant-4.18.0-abc123.radpkg
        latest.json
     beta/
        radiant-4.19.0-beta1-xyz789.radpkg
     archive/
        radiant-4.17.0-def456.radpkg

3. DEPLOYED INSTANCE (Cloud - Per Instance)
   s3://radiant-{appId}-{env}-deployments/
     current/
        radiant-4.18.0-abc123.radpkg
     snapshots/
        snapshot-2024-12-24T10-30-00Z/
           package.radpkg
           parameters.json
           db-snapshot-id.txt
        ...
```

Key Implementation Files

Swift Deployer

File	Purpose
`Models/InstallationParameters.swift`	DeploymentMode enum, TierLevel, InstallationParameters, InstanceParameters, ParameterChanges, DeploymentSnapshot
`Services/DeploymentService.swift`	Mode detection, executeInstall, executeUpdate, executeRollback, parameter fetching/merging
`Services/PackageService.swift`	Package discovery, download, verification, caching
`Views/Deployment/ParameterEditorView.swift`	UI for editing parameters based on mode

Build Tools

File	Purpose
`tools/scripts/build-package.sh`	Build deployment packages from source
`tools/version-manager.ts`	Version bumping and synchronization

Data Flow Diagrams

Install Flow

```
User                    Deployer                    AWS



1. Select App
2. Select Env
3. Select Tier
4. Click Deploy
                       5. Check instance exists      (None found)
                       6. Mode = INSTALL
                       7. Load DEFAULT params
                       8. Download latest package      S3
                       9. Verify package integrity
                       10. Deploy CDK stacks       CloudFormation
                       11. Run ALL migrations       Aurora
                       12. SEED AI Registry        Aurora
                       13. Create initial admin      Cognito
                       14. Store deployment meta     S3 + DB
                       15. Report success
```

Update Flow

```
User                    Deployer                    AWS + Instance



1. Select App
2. Select Env
3. Change Params
4. Click Update
                       5. Check instance exists      (Found!)
                       6. Mode = UPDATE
                       7. Fetch CURRENT params       Radiant API
                       8. Create snapshot         S3
                       9. MERGE user changes
                       10. Validate changes
                       11. Download target package     S3
                       12. Deploy CDK stacks       CloudFormation
                       13. Run INCREMENTAL migrations   Aurora
                       14. SKIP AI Registry seeding
                       15. Update deployment meta     S3 + DB
                       16. Report success
```

Rollback Flow

```
User                    Deployer                     AWS



1. Select App
2. Select Env
3. Select Snapshot
4. Click Rollback
                       5. Mode = ROLLBACK
                       6. Load target snapshot       S3
                       7. Validate compatibility
                       8. Create safety snapshot     S3
                       9. Download snapshot package    S3
                       10. Deploy CDK stacks       CloudFormation
                       11. Optionally restore DB     RDS Snapshot
                       12. Update deployment meta     S3 + DB
                       13. Report success
```

Verification Checklist

Deployment Modes

- **On INSTALL**: Parameters come from defaults
- **On UPDATE**: Parameters come from running instance + user changes
- **On ROLLBACK**: Parameters come from selected snapshot

AI Registry Seeding

- AI Registry is seeded ONLY on fresh install
- On UPDATE, AI Registry is preserved (not touched)
- Admins can add/remove providers via Admin Dashboard

Deployment Packages

- Packages are created by build-package.sh script
- Package creation is triggered by code changes or version bumps
- Packages are stored in local cache, S3 release bucket, and instance bucket

Parameter Rules

- Region CANNOT be changed after install
- Tier CAN be changed on update (with feature validation)
- All parameter changes are tracked via snapshots

Related Documentation

- Deployer Admin Guide - User-facing documentation
- Deployment Guide - Deployment procedures
- API Reference - API documentation

Think Tank AI - User Guide

Your gateway to 100+ AI models in one place

Version: 3.2.0 (Platform: RADIANT 4.18.1) Last Updated: December 2024

Welcome to Think Tank

Think Tank is your all-in-one AI assistant platform. Access the world's best AI models—GPT-4, Claude, Gemini, and 100+ more—from a single, beautiful interface.

Table of Contents

---

1. Getting Started

1.1 Creating Your Account

1. Visit **thinktank.ai**
2. Click **Get Started Free**
3. Sign up with:
    - Email and password
    - Google account
    - Microsoft account
    - Apple ID

4. Verify your email
5. Complete your profile

1.2 Choosing a Plan

Plan	Price	Best For
Free	$0/month	Trying out Think Tank
Starter	$29/month	Individual creators
Pro	$99/month	Power users
Team	$49/user/month	Small teams
Business	$199/user/month	Organizations

1.3 Your First Chat

1. Click **New Chat** or press `Ctrl+N`
2. Type your question or request
3. Press `Enter` or click **Send**
4. Watch as AI responds in real-time

Try these starter prompts: - "Explain quantum computing like I'm 10 years old" - "Write a professional email declining a meeting" - "Help me debug this Python code: [paste code]" - "Create a meal plan for the week"

---

2. Your Dashboard

2.1 Interface Overview

```
Think Tank                          [Search]  [Credits: 450]


Chats                            Chat Area

Starred      Welcome! How can I help you today?
Today
 Chat 1
 Chat 2
Yesterday
 Chat 3



Personas    Type your message...              [Send]
Canvas
Settings
            Model: GPT-4 Turbo  |  Focus: General  |
```

2.2 Sidebar Navigation

Icon	Section	Description
	Chats	All your conversations
	Starred	Important chats
	Personas	Custom AI personalities
	Canvas	Visual workspace
	Usage	Credit usage stats
	Settings	Account settings

2.3 Quick Actions

Action	Shortcut
New Chat	`Ctrl+N`
Search	`Ctrl+K`
Toggle Sidebar	`Ctrl+B`
Settings	`Ctrl+,`

---

3. Chatting with AI

3.1 Sending Messages

Text Messages: - Type in the input box - Press `Enter` to send - Use `Shift+Enter` for new lines

Attachments: - Click to attach files - Drag & drop images, PDFs, code files - Paste images directly (`Ctrl+V`)

Voice Input: - Click microphone icon - Speak your message - Click again to stop

3.2 Message Actions

Hover over any message to see actions:

Icon	Action	Description
	Copy	Copy message text
	Regenerate	Get a new response
	Edit	Modify your message
/	Rate	Help improve AI
	Pin	Keep message visible
	Delete	Remove message

3.3 Streaming Responses

AI responses stream in real-time. You can: - **Stop**: Click to stop generation - **Continue**: Ask "continue" if response was cut off - **Regenerate**: Get a different response

3.4 Multi-Turn Conversations

Think Tank remembers your conversation context:

```
You: I'm planning a trip to Japan
AI:  That's exciting! When are you planning to visit...

You: What about the food?
AI:  Japanese cuisine is incredible! Based on your trip...
     [AI remembers you're going to Japan]
```

3.5 Code in Chats

Code is automatically syntax-highlighted:

```python
def hello_world():
    print("Hello, Think Tank!")
```

Click **Copy** to copy code blocks, or **Run** for supported languages.

---

4. Choosing Models

4.1 Model Selection

Click the model selector at the bottom of the chat:

```
 Select Model

   Favorites
    GPT-4 Turbo           $0.02/msg
    Claude 3 Opus         $0.03/msg

   Recommended
    GPT-4o                $0.01/msg
    Claude 3.5 Sonnet     $0.01/msg
    Gemini 1.5 Pro        $0.01/msg

   Writing
   Coding
   Analysis
   Creative
 [View All 100+ Models]
```

4.2 Model Categories

Category	Best For	Top Models
General	Everyday tasks	GPT-4o, Claude 3.5
Writing	Content creation	Claude 3 Opus, GPT-4
Coding	Programming help	GPT-4 Turbo, CodeLlama
Analysis	Data & research	Gemini 1.5, Claude 3
Creative	Art & ideas	GPT-4, Mistral Large
Vision	Image understanding	GPT-4V, LLaVA
Fast	Quick responses	GPT-3.5, Claude Instant

Choosing the Right Model

For everyday questions and tasks: Start with GPT-4o or Claude 3.5 Sonnet. These models offer the best balance of quality, speed, and cost. They handle most tasks excellently including writing, answering questions, brainstorming, and light coding.

For professional writing: Claude 3 Opus excels at long-form content, maintaining consistent tone, and nuanced writing. GPT-4 is also excellent for business documents and creative writing.

For coding and technical work: GPT-4 Turbo has strong coding abilities across many languages. For specialized tasks, consider CodeLlama (open source, good for common languages) or specialized models like DeepSeek Coder.

For data analysis: Gemini 1.5 Pro handles very long documents (up to 1 million tokens) making it ideal for analyzing large datasets or documents. Claude 3 is excellent for nuanced analytical reasoning.

For creative projects: GPT-4 and Mistral Large are both creative and can help with brainstorming, storytelling, and idea generation. They're less constrained in creative contexts.

For image understanding: GPT-4V (Vision) and Claude 3 Vision can analyze images, read text from photos, describe scenes, and answer questions about visual content.

For quick, simple tasks: GPT-3.5 Turbo and Claude Instant are much faster and cheaper. Use them for simple questions, formatting, or when you need instant responses.

4.3 Auto Mode

Let Think Tank choose the best model:

1. Enable **Auto Mode** in settings
2. Our Brain Router analyzes your request
3. Automatically selects optimal model
4. Balances quality, speed, and cost

How Auto Mode Works

When you enable Auto Mode, Think Tank's Brain Router analyzes each message you send and selects the best model based on:

- **Task complexity**: Simple questions go to fast models; complex tasks go to powerful models
- **Content type**: Coding questions route to code-specialized models; creative requests to creative models
- **Your history**: Learns your preferences over time and adjusts recommendations
- **Cost efficiency**: Avoids using expensive models when cheaper ones would work equally well
- **Current availability**: Routes around any models experiencing slowdowns

When to use Auto Mode: - You're not sure which model to use - You want to optimize cost without sacrificing quality - You have varied tasks throughout the day - You're new to Think Tank

When to choose manually: - You need a specific model's unique capabilities - You're doing specialized work (e.g., always want Claude for writing) - You're comparing models intentionally - You have strong preferences for certain models

4.4 Model Comparison

Split-screen to compare models:

1. Click **Compare** button
2. Select 2-4 models
3. Send message to all simultaneously
4. See responses side-by-side

---

5. Focus Modes & Personas

5.1 Focus Modes

Pre-configured modes for specific tasks:

Mode	Optimized For
Professional	Business writing, emails
Developer	Code, debugging, architecture
Research	Analysis, citations, accuracy
Creative	Stories, brainstorming
Learning	Explanations, tutoring
Concise	Brief, direct answers

To switch modes: 1. Click the Focus selector 2. Choose your mode 3. AI adapts its style

Focus Mode Details

Professional Mode: The AI adopts a business-appropriate tone. Responses are polished, formal, and suitable for workplace communication. Great for drafting emails, reports, presentations, and client communications. Avoids casual language and ensures professional formatting.

Developer Mode: Optimized for technical work. The AI provides code with proper syntax highlighting, explains technical concepts clearly, suggests best practices, and can help debug issues. Responses include code comments and consider edge cases.

Research Mode: Emphasizes accuracy and thoroughness. The AI cites sources when possible, acknowledges uncertainty, presents multiple perspectives, and structures information logically. Ideal for academic work, fact-checking, and deep analysis.

Creative Mode: Removes constraints on creativity. The AI is more willing to explore unusual ideas, use vivid language, and think outside the box. Perfect for brainstorming, creative writing, storytelling, and generating innovative solutions.

Learning Mode: The AI becomes a patient tutor. Explanations start from basics and build up, concepts are broken into digestible pieces, and the AI checks understanding before moving on. Great for studying new topics.

Concise Mode: Responses are brief and to the point. The AI avoids lengthy explanations and gets straight to the answer. Useful when you need quick facts or are in a hurry.

5.2 Custom Personas

Create your own AI personalities:

1. Go to **Personas** → **Create New**
2. Configure:
   - **Name**: “Marketing Expert”
   - **Personality**: Professional, enthusiastic
   - **Expertise**: Digital marketing, SEO
   - **Style**: Uses bullet points, data-driven
3. Click **Save**

Example Persona:

```
Name: Code Reviewer
Personality: Thorough, constructive
Instructions: Review code for bugs, security issues,
              and best practices. Always suggest
              improvements with examples.
```

5.3 Sharing Personas

- **Public**: Share with all Think Tank users
- **Team**: Share within your organization
- **Private**: Only you can use

---

6. Canvas & Artifacts

6.1 What is Canvas?

Canvas is your visual workspace for complex outputs: - Code files with syntax highlighting - Diagrams and flowcharts - Documents and reports - Data tables - Mind maps

6.2 Creating Artifacts

When AI generates complex content, it appears as an artifact:

```
   business_plan.md

 # Business Plan

 ## Executive Summary
 ...


 [Copy] [Download] [Edit] [Version History]
```

6.3 Artifact Actions

Action	Description
Copy	Copy content to clipboard
Download	Save as file
Edit	Modify directly
Versions	View previous versions
Share	Generate share link
To Canvas	Open in full Canvas view

6.4 Full Canvas Mode

For larger projects:

1. Click **Canvas** in sidebar
2. Create new canvas or open existing
3. Add multiple artifacts

4. Arrange spatially
5. Connect related items

---

7. Collaboration Features

7.1 Sharing Chats

Share any conversation:

1. Click **Share** ( ) on a chat
2. Choose visibility:
    - **Link**: Anyone with link
    - **Team**: Your organization
    - **Private**: Specific people
3. Copy and share the link

7.2 Real-Time Collaboration

Work together on the same chat:

1. Share chat with **Edit** access
2. Multiple users can:
    - Send messages simultaneously
    - See each other's cursors
    - React to messages
3. Changes sync in real-time

7.3 Team Workspaces

For Team and Business plans:

- **Shared Chats**: Team-visible conversations
- **Shared Personas**: Team AI configurations
- **Shared Canvas**: Collaborative workspaces
- **Usage Dashboard**: Team analytics

7.4 Comments & Annotations

Add notes to any message:

1. Hover over message
2. Click **Comment** ( )
3. Add your note
4. Tag teammates with @mention

---

8. Managing Your Account

8.1 Profile Settings

Access via **Settings** → **Profile**:

Setting	Description
Display Name	Your visible name
Email	Login email
Avatar	Profile picture
Language	Interface language
Timezone	For scheduled features

8.2 Preferences

Customize your experience:

Preference	Options
Theme	Light, Dark, System
Font Size	Small, Medium, Large
Default Model	Your preferred model
Auto Mode	Enable/disable
Sound Effects	On/Off
Notifications	Email, Push, None

8.3 Data & Privacy

Control your data:

- **Export Data**: Download all your chats
- **Delete History**: Remove chat history
- **Training Opt-Out**: Exclude from AI training
- **Data Retention**: Set auto-delete period

8.4 Connected Apps

Manage integrations:

- Google Drive
- Dropbox
- Notion
- Slack
- GitHub

---

9. Credits & Billing

9.1 Understanding Credits

Credits are Think Tank's universal currency:

Credit Value	Equivalent
1 credit	$0.01
100 credits	$1.00

9.2 Credit Usage

Different models cost different amounts:

Model	~Cost per Message
GPT-3.5 Turbo	0.5 credits
GPT-4o	1-2 credits
GPT-4 Turbo	2-3 credits
Claude 3.5 Sonnet	1-2 credits
Claude 3 Opus	3-5 credits

Actual cost depends on message length

9.3 Viewing Usage

Check your usage in **Settings** → **Usage**:

```
Credit Usage - December 2024

Balance:        450 credits
Used this month: 1,550 credits
Included:       2,000 credits

[           ] 77% used

By Model:
  GPT-4 Turbo     800 credits (52%)
  Claude 3        500 credits (32%)
  Other           250 credits (16%)
```

9.4 Purchasing Credits

Need more credits?

1. Go to **Settings** → **Billing**
2. Click **Buy Credits**
3. Select amount:
    - 500 credits - $5
    - 1,000 credits - $9 (10% bonus)
    - 5,000 credits - $40 (20% bonus)
4. Complete payment

9.5 Subscription Management

Manage your plan:

- **Upgrade**: Get more features and credits
- **Downgrade**: Switch to lower tier (end of period)
- **Cancel**: Cancel subscription (keep access until end)
- **Invoices**: Download billing history

---

10. Tips & Best Practices

10.1 Writing Better Prompts

Be Specific:

```
"Write about dogs"
"Write a 200-word blog post about training golden
   retriever puppies, focusing on positive reinforcement"
```

Provide Context:

```
"Fix this code"
"Fix this Python code that should sort a list but
   throws an IndexError: [paste code]"
```

Set the Format:

```
"Give me ideas"
"Give me 5 blog post ideas about sustainable living,
   formatted as bullet points with a brief description"
```

10.2 Getting Better Results

Technique	Example
Chain of thought	"Think step by step..."
Role assignment	"Act as a senior developer..."
Examples	"Here's an example of what I want..."
Constraints	"In 100 words or less..."
Iteration	"Good, but make it more formal"

10.3 Saving Credits

- Use **Auto Mode** for optimal model selection
- Use **GPT-3.5** for simple tasks
- Be concise in your prompts
- Avoid regenerating unnecessarily
- Use **Focus Modes** for specialized tasks

10.4 Organizing Chats

- **Star** important conversations
- **Folders**: Group related chats
- **Tags**: Add searchable labels
- **Search**: Find any past conversation

---

11. Keyboard Shortcuts

11.1 General

Shortcut	Action
`Ctrl+N`	New chat
`Ctrl+K`	Search

Shortcut	Action
`Ctrl+B`	Toggle sidebar
`Ctrl+,`	Settings
`Ctrl+/`	Show shortcuts
`Escape`	Close modal

11.2 Chat

Shortcut	Action
`Enter`	Send message
`Shift+Enter`	New line
`Ctrl+↑`	Edit last message
`Ctrl+Shift+C`	Copy last response
`Ctrl+Shift+R`	Regenerate
`Ctrl+.`	Stop generation

11.3 Navigation

Shortcut	Action
`Alt+↑/↓`	Previous/next chat
`Ctrl+1-9`	Switch to chat 1-9
`Ctrl+Tab`	Cycle tabs

---

12. FAQ

Getting Started

Q: Is Think Tank free? A: Yes! The Free plan includes 50 credits/month. Upgrade for more credits and features.

Q: Which AI model should I use? A: Enable Auto Mode and let us choose, or: - General tasks → GPT-4o or Claude 3.5 Sonnet - Complex analysis → GPT-4 Turbo or Claude 3 Opus - Quick answers → GPT-3.5 Turbo

Q: Can I use Think Tank on mobile? A: Yes! Visit thinktank.ai on any mobile browser, or download our iOS/Android apps.

Credits & Billing

Q: What happens when I run out of credits? A: You can purchase more credits or wait for your monthly refresh (paid plans).

Q: Do unused credits roll over? A: Monthly included credits expire. Purchased credits never expire.

Q: Can I get a refund? A: Contact support within 14 days for subscription refunds.

Privacy & Security

Q: Is my data used to train AI? A: By default, no. You can verify in Settings → Privacy.

Q: Who can see my chats? A: Only you, unless you explicitly share them.

Q: Is my data encrypted? A: Yes, with AES-256 encryption at rest and TLS 1.3 in transit.

Troubleshooting

Q: Why is the AI response slow? A: Complex queries or busy times may cause delays. Try a faster model.

Q: Why did my response get cut off? A: Models have output limits. Type "continue" to get the rest.

Q: I found a bug. How do I report it? A: Click **Help** → **Report Issue** or email support@thinktank.ai.

Need Help?

- **Help Center**: help.thinktank.ai
- **Live Chat**: Click the chat bubble
- **Email**: support@thinktank.ai
- **Twitter**: @ThinkTankAI
- **Discord**: discord.gg/thinktank

Version History

Version	Date	Changes
3.2.0	December 2024	Time Machine, enhanced collaboration, A/B experiments, **Delight System**
3.1.0	November 2024	Canvas improvements, new models
3.0.0	October 2024	Initial release

---

13. Delight System

Think Tank includes a personality system called "Delight" that makes your AI experience more engaging.

13.1 What is Delight?

Delight adds contextual, friendly messages during your conversations:

- **Domain Loading**: "Consulting the fundamental forces…" when you ask physics questions
- **Time Awareness**: "Burning the midnight tokens" during late-night sessions
- **Model Dynamics**: "Consensus forming…" when multiple models agree
- **Wellbeing Nudges**: "You've been thinking hard. Time for a break?"

13.2 Personality Modes

Choose your preferred personality style in **Settings → Delight**:

Mode	Description
Professional	Minimal, business-appropriate feedback
Subtle	Light touches of personality
Expressive	Full personality with humor
Playful	Maximum fun, includes easter eggs

13.3 Achievements

Earn achievements as you use Think Tank:

Achievement	How to Unlock
Domain Explorer	Explore 10+ knowledge domains
Week Warrior	Use Think Tank 7 days in a row
Renaissance Mind	Explore 50+ domains
Monthly Mind	30-day streak

View your achievements in **Settings → Achievements**.

13.4 Easter Eggs

Think Tank has hidden surprises! Try: - Typing special phrases - Using keyboard shortcuts - Exploring during special times

Discover them yourself—that's half the fun!

13.5 Sound Effects

Enable audio feedback in **Settings → Delight → Sounds**:

Theme	Style
Default	Pleasant chimes
Mission Control	NASA-inspired beeps
Library	Page turns, book sounds
Workshop	Tool clicks
Emissions	Tesla-style… fun

13.6 Customizing Delight

Toggle individual features: - Domain messages - Model personality - Time awareness - Achievements - Wellbeing nudges - Easter eggs - Sound effects

Set intensity level (1-10) to control how often messages appear.

Thank you for using Think Tank! We're constantly improving based on your feedback.

RADIANT API Reference

Complete API documentation for the RADIANT AI Platform.

Base URL: `https://api.radiant.example.com/v2`

Authentication: Bearer token (API key or JWT)

`Authorization: Bearer rad_your_api_key`

Chat Completions

Create Chat Completion

`POST /v2/chat/completions`

Create a chat completion with any supported model.

Request Body:

Field	Type	Required	Description
`model`	string		Model ID (e.g., `gpt-4o`, `claude-3-sonnet`)
`messages`	array		Array of message objects
`max_tokens`	integer		Maximum tokens to generate
`temperature`	number		Sampling temperature (0-2)
`top_p`	number		Nucleus sampling (0-1)
`stream`	boolean		Stream response tokens
`stop`	string/array		Stop sequences
`functions`	array		Function definitions
`function_call`	string/object		Function calling mode

Message Object:

Field	Type	Required	Description
`role`	string		`system`, `user`, `assistant`, `function`
`content`	string		Message content
`name`	string		Function name (for function messages)

Example Request:

```
{
  "model": "gpt-4o",
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello!"}
  ],
  "max_tokens": 1000,
  "temperature": 0.7
}
```

Response:

```
{
  "id": "chatcmpl_abc123",
  "object": "chat.completion",
  "created": 1703980800,
  "model": "gpt-4o",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I help you today?"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 25,
    "completion_tokens": 10,
    "total_tokens": 35
  }
}
```

Models

List Models

`GET /v2/models`

List all available models.

Query Parameters:

Parameter	Type	Description
`category`	string	Filter by category (`chat`, `embedding`, `image`)
`provider`	string	Filter by provider (`openai`, `anthropic`, `google`)

Response:

```
{
  "object": "list",
  "data": [
    {
      "id": "gpt-4o",
      "object": "model",
      "created": 1703980800,
      "owned_by": "openai",
      "display_name": "GPT-4o",
      "category": "chat",
      "context_window": 128000,
      "input_cost_per_1k": 0.005,
      "output_cost_per_1k": 0.015,
      "capabilities": ["chat", "vision", "function_calling"]
    }
  ]
}
```

Get Model

`GET /v2/models/{model_id}`

Get details for a specific model.

Embeddings

Create Embeddings

`POST /v2/embeddings`

Generate embeddings for text.

Request Body:

Field	Type	Required	Description
`model`	string		Embedding model ID
`input`	string/array		Text to embed
`encoding_format`	string		`float` or `base64`

Response:

```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [0.0023, -0.0091, ...]
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 8,
    "total_tokens": 8
  }
}
```

Billing

Get Credit Balance

`GET /v2/billing/credits`

Get current credit balance.

Response:

```json
{
  "data": {
    "available": 150.50,
    "reserved": 10.00,
    "currency": "USD",
    "updated_at": "2024-01-15T10:30:00Z"
  }
}
```

Get Usage

`GET /v2/billing/usage`

Get usage data for a period.

Query Parameters:

Parameter	Type	Description
`start_date`	string	Start date (YYYY-MM-DD)
`end_date`	string	End date (YYYY-MM-DD)
`group_by`	string	`day`, `model`, `endpoint`

Webhooks

List Webhooks

`GET /v2/webhooks`

List configured webhooks.

Create Webhook

`POST /v2/webhooks`

Request Body:

```json
{
  "url": "https://your-server.com/webhook",
  "event_types": ["billing.low_balance", "usage.quota_reached"],
  "description": "Billing alerts"
}
```

Response:

```json
{
  "data": {
    "id": "wh_abc123",
    "url": "https://your-server.com/webhook",
    "secret": "whsec_xyz789...",
    "event_types": ["billing.low_balance", "usage.quota_reached"],
    "is_active": true,
    "created_at": "2024-01-15T10:30:00Z"
  }
}
```

Test Webhook

`POST /v2/webhooks/{webhook_id}/test`

Send a test event to the webhook.

Batch Processing

Create Batch Job

`POST /v2/batch/jobs`

Create a batch processing job.

Request Body:

```json
{
  "type": "completions",
  "model": "gpt-4o",
  "input_file": "batch-input.jsonl",
  "options": {
    "system_prompt": "You are a helpful assistant.",
    "max_tokens": 500
  }
}
```

Get Batch Job

`GET /v2/batch/jobs/{job_id}`

Get batch job status and results.

List Batch Jobs

`GET /v2/batch/jobs`

List all batch jobs.

Error Codes

RADIANT uses standardized error codes across all endpoints. See Error Codes Reference for the complete list.

Error Categories

Category	Code Range	Description
Authentication	`RADIANT_AUTH_1xxx`	Token, API key, session errors
Authorization	`RADIANT_AUTHZ_2xxx`	Permission, role, tenant errors
Validation	`RADIANT_VAL_3xxx`	Input validation errors
Resource	`RADIANT_RES_4xxx`	Not found, conflict, quota errors
Rate Limiting	`RADIANT_RATE_5xxx`	Throttling and rate limit errors
AI/Model	`RADIANT_AI_6xxx`	Model, provider, inference errors
Billing	`RADIANT_BILL_7xxx`	Credits, subscription errors
Storage	`RADIANT_STOR_8xxx`	File upload, storage errors
Internal	`RADIANT_INT_9xxx`	Server, database, timeout errors

Common Error Codes

Code	HTTP	Retryable	Description
`RADIANT_AUTH_1001`	401		Invalid authentication token
`RADIANT_AUTH_1004`	401		Invalid API key
`RADIANT_VAL_3001`	400		Required field missing
`RADIANT_RES_4001`	404		Resource not found
`RADIANT_RATE_5001`	429		Rate limit exceeded
`RADIANT_BILL_7001`	402		Insufficient credits
`RADIANT_AI_6004`	502		AI provider error
`RADIANT_INT_9001`	500		Internal server error

Error Response Format:

```
{
  "error": {
    "code": "RADIANT_RATE_5001",
    "message": "Too many requests. Please slow down.",
    "category": "rate_limit",
    "retryable": true,
    "timestamp": "2024-12-25T10:30:00.000Z"
  }
}
```

Retry-After Header: Retryable errors include `Retry-After` header with seconds to wait.

Rate Limits

Tier	Requests/min	Tokens/min
Free	10	10,000
Starter	50	50,000
Professional	100	200,000
Business	500	1,000,000
Enterprise	2,000	Unlimited

Rate limit headers:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1703980860
```

SDKs

TypeScript/JavaScript

```
npm install @radiant/sdk
```

```
import { RadiantClient } from '@radiant/sdk';

const client = new RadiantClient({ apiKey: 'your-key' });
const response = await client.chat.create({
  model: 'gpt-4o',
  messages: [{ role: 'user', content: 'Hello!' }],
});
```

Python

```
pip install radiant-sdk
```

```
from radiant import RadiantClient

client = RadiantClient(api_key="your-key")
response = client.chat.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": "Hello!"}],
)
```

CLI

```
npm install -g @radiant/cli
radiant auth login
radiant chat send "Hello!"
```

Changelog

See CHANGELOG.md for version history.

Support

- **Email:** support@radiant.example.com
- **Documentation:** https://docs.radiant.example.com
- **Status:** https://status.radiant.example.com

RADIANT API Versioning Guide

Overview

This document describes the API versioning strategy for the RADIANT platform.

Versioning Strategy

URL Path Versioning

RADIANT uses URL path versioning as the primary versioning mechanism:

```
https://api.radiant.example.com/v2/models
https://api.radiant.example.com/v2/chat/completions
```

Version Lifecycle

Version	Status	Support End	Deprecation
v1	Deprecated	2024-06-01	Sunset
v2	Current	-	-
v3	Planned	-	Q3 2025

Support Policy

- **Current version**: Full support, all new features
- **Previous version**: Security fixes only, 12 months after new version
- **Deprecated**: No fixes, 6-month sunset warning

Breaking vs Non-Breaking Changes

Non-Breaking Changes (No Version Bump)

These changes can be made to the current version:

- Adding new endpoints
- Adding new optional request parameters
- Adding new response fields
- Adding new enum values (with graceful handling)
- Performance improvements
- Bug fixes that don't change behavior

Breaking Changes (Require New Version)

These changes require a new API version:

- Removing endpoints
- Removing request/response fields
- Changing field types
- Changing validation rules
- Changing authentication methods
- Changing error response formats
- Removing enum values
- Changing default values

Version Header Support

Request Headers

```
# Specify API version via header (optional override)
X-API-Version: 2024-12-01

# Request specific features
X-API-Features: beta-orchestration,streaming-v2
```

Response Headers

```
# Current API version
X-API-Version: 2
X-API-Version-Date: 2024-12-01

# Deprecation warning
Deprecation: true
Sunset: Sat, 01 Jun 2024 00:00:00 GMT
Link: <https://api.radiant.example.com/v2>; rel="successor-version"
```

Deprecation Process

Timeline

```
Day 0:    Announce deprecation
          Add Deprecation header
          Update documentation

Month 3:  Send reminder emails
          Log deprecation warnings

Month 6:  Begin returning 299 status for deprecated endpoints
          Increase warning frequency

Month 12: Sunset - Return 410 Gone
          Redirect to new version docs
```

Deprecation Headers

```
// Add deprecation headers to old endpoints
function addDeprecationHeaders(res: Response, sunset: Date): void {
  res.setHeader('Deprecation', 'true');
  res.setHeader('Sunset', sunset.toUTCString());
  res.setHeader('Link', '<https://api.radiant.example.com/v3>; rel="successor-version"');
}
```

Deprecation Warnings

```
// Log deprecation usage for migration tracking
async function logDeprecatedUsage(req: Request): Promise<void> {
  await analytics.track({
    event: 'deprecated_api_usage',
    properties: {
      endpoint: req.path,
      version: extractVersion(req),
      apiKey: extractKeyId(req),
      tenant: extractTenantId(req),
    },
  });
}
```

Migration Guide Template

v1 to v2 Migration

```
# Migrating from API v1 to v2

## Breaking Changes

### 1. Authentication
- v1: API key in query string (`?api_key=xxx`)
- v2: API key in header (`Authorization: Bearer xxx`)

### 2. Response Format
- v1: Flat response (`{ models: [...] }`)
- v2: Wrapped response (`{ data: [...], meta: {...} }`)

### 3. Error Format
- v1: `{ error: "message" }`
- v2: `{ error: { code: "...", message: "...", details: [...] } }`

## Migration Steps

1. Update authentication headers
2. Update response parsing
3. Update error handling
4. Test all endpoints
5. Switch base URL from /v1 to /v2
```

Feature Flags

Beta Features

```
// Enable beta features via header
const betaFeatures = {
  'beta-orchestration': true,
  'beta-streaming-v2': true,
  'beta-function-calling': true,
};

function isBetaEnabled(req: Request, feature: string): boolean {
  const features = req.headers['x-api-features']?.split(',') || [];
  return features.includes(feature) && betaFeatures[feature];
}
```

Graduated Features

```
// Track feature graduation
const featureGraduation = {
  'function-calling': {
    beta: '2024-06-01',
    stable: '2024-09-01',
    version: 'v2',
  },
  'streaming-v2': {
    beta: '2024-09-01',
    stable: null, // Still in beta
    version: 'v2',
  },
};
```

SDK Versioning

SDK Version Matrix

SDK	Latest	Min API Version	Max API Version
JavaScript	2.5.0	v2	v2
Python	2.3.0	v2	v2
Go	1.2.0	v2	v2
Ruby	1.1.0	v2	v2

SDK Version Headers

```
# SDKs include version info
User-Agent: radiant-js/2.5.0 node/20.10.0
X-Radiant-SDK: js
X-Radiant-SDK-Version: 2.5.0
```

OpenAPI Specification

Versioned Specs

```
/docs/openapi/v2.yaml     # Current version
/docs/openapi/v2-beta.yaml # With beta features
/docs/openapi/v1.yaml     # Deprecated version
```

Schema Versioning

```yaml
# openapi.yaml
openapi: 3.1.0
info:
  title: RADIANT API
  version: 2.0.0
  x-api-version: v2
  x-version-date: '2024-12-01'
  x-deprecation-date: null
```

Testing Versions

Version Compatibility Tests

```
describe('API Version Compatibility', () => {
  it('should support v2 endpoints', async () => {
    const res = await fetch('/v2/health');
    expect(res.status).toBe(200);
  });

  it('should return 410 for sunset v1 endpoints', async () => {
    const res = await fetch('/v1/health');
    expect(res.status).toBe(410);
    expect(res.headers.get('Link')).toContain('/v2');
  });

  it('should include deprecation headers for deprecated endpoints', async () => {
    const res = await fetch('/v2/deprecated-endpoint');
    expect(res.headers.get('Deprecation')).toBe('true');
    expect(res.headers.get('Sunset')).toBeDefined();
  });
});
```

Client Communication

Changelog

Maintain a public changelog:

```markdown
# API Changelog

## 2024-12-24 (v2.5)
- Added: Orchestration patterns endpoint
- Added: Workflow proposals endpoint
- Changed: Increased rate limits for Professional tier

## 2024-12-01 (v2.4)
- Added: AI translation for localization
- Deprecated: Legacy /translate endpoint (sunset 2025-06-01)
```

Email Notifications

```
// Notify developers of breaking changes
async function notifyApiChanges(change: ApiChange): Promise<void> {
  const affectedKeys = await getApiKeysUsingEndpoint(change.endpoint);

  for (const key of affectedKeys) {
    await sendEmail({
      to: key.ownerEmail,
      subject: `RADIANT API: ${change.type} - ${change.endpoint}`,
      template: 'api-change-notification',
      data: {
        change,
        migrationGuide: change.migrationGuideUrl,
        deadline: change.sunsetDate,
      },
    });
  }
}
```

Best Practices

For API Developers

1. **Plan for change**: Design APIs to be extensible
2. **Use optional fields**: Make new fields optional with defaults
3. **Version from day one**: Include version in all endpoints
4. **Document everything**: Keep OpenAPI specs updated
5. **Communicate early**: 12-month deprecation notice minimum

For API Consumers

1. **Pin versions**: Don't use unversioned endpoints
2. **Handle unknown fields**: Ignore unexpected response fields
3. **Monitor deprecation headers**: Set up alerts
4. **Test regularly**: Run integration tests against current version
5. **Subscribe to updates**: Follow changelog and email updates

Contact

Role	Contact	Purpose
API Support	api-support@radiant.example.com	Usage questions
Developer Relations	devrel@radiant.example.com	SDKs, docs
Engineering	engineering@radiant.example.com	Bug reports

RADIANT Error Codes Reference

Standardized error codes for consistent API responses across all RADIANT services.

Overview

All RADIANT errors follow a consistent format:

```json
{
  "error": {
    "code": "RADIANT_AUTH_1001",
    "message": "Invalid authentication token. Please sign in again.",
    "category": "authentication",
    "retryable": false,
    "timestamp": "2024-12-25T10:30:00.000Z"
  }
}
```

Error Code Format

```
RADIANT_<CATEGORY>_<NUMBER>
```

- **RADIANT** - Prefix for all error codes
- **CATEGORY** - Short category identifier (AUTH, VAL, RES, etc.)
- **NUMBER** - Unique 4-digit number within category

Authentication Errors (1xxx)

Errors related to authentication and identity.

Code	HTTP	Retryable	Description
`RADIANT_AUTH_1001`	401		Invalid authentication token
`RADIANT_AUTH_1002`	401		Token has expired
`RADIANT_AUTH_1003`	401		Missing authentication token
`RADIANT_AUTH_1004`	401		Invalid API key
`RADIANT_AUTH_1005`	401		API key has expired
`RADIANT_AUTH_1006`	401		API key has been revoked
`RADIANT_AUTH_1007`	403		Insufficient API key scope
`RADIANT_AUTH_1008`	401		Multi-factor authentication required
`RADIANT_AUTH_1009`	401		Session has expired

Authorization Errors (2xxx)

Errors related to permissions and access control.

Code	HTTP	Retryable	Description
`RADIANT_AUTHZ_2001`	403		Forbidden - access denied
`RADIANT_AUTHZ_2002`	403		Tenant ID mismatch
`RADIANT_AUTHZ_2003`	403		Required role not assigned
`RADIANT_AUTHZ_2004`	403		Permission denied
`RADIANT_AUTHZ_2005`	403		Resource access denied
`RADIANT_AUTHZ_2006`	403		Subscription tier insufficient

Validation Errors (3xxx)

Errors related to input validation.

Code	HTTP	Retryable	Description
`RADIANT_VAL_3001`	400		Required field is missing
`RADIANT_VAL_3002`	400		Invalid field format
`RADIANT_VAL_3003`	400		Value out of allowed range
`RADIANT_VAL_3004`	400		Invalid data type
`RADIANT_VAL_3005`	400		Constraint violation
`RADIANT_VAL_3006`	400		Schema mismatch
`RADIANT_VAL_3007`	400		Invalid JSON in request body
`RADIANT_VAL_3008`	400		Maximum length exceeded
`RADIANT_VAL_3009`	400		Minimum length required

Resource Errors (4xxx)

Errors related to resources and entities.

Code	HTTP	Retryable	Description
`RADIANT_RES_4001`	404		Resource not found
`RADIANT_RES_4002`	409		Resource already exists
`RADIANT_RES_4003`	410		Resource has been deleted
`RADIANT_RES_4004`	423		Resource is locked
`RADIANT_RES_4005`	409		Resource conflict
`RADIANT_RES_4006`	429		Resource quota exceeded

Rate Limiting Errors (5xxx)

Errors related to rate limiting and throttling.

Code	HTTP	Retryable	Description
`RADIANT_RATE_5001`	429		Rate limit exceeded
`RADIANT_RATE_5002`	429		Tenant rate limit exceeded
`RADIANT_RATE_5003`	429		User rate limit exceeded
`RADIANT_RATE_5004`	429		API key rate limit exceeded
`RADIANT_RATE_5005`	429		Model rate limit exceeded
`RADIANT_RATE_5006`	429		Burst limit exceeded

Retry-After Header: Rate limit errors include `Retry-After` header with seconds to wait.

AI/Model Errors (6xxx)

Errors related to AI models and inference.

Code	HTTP	Retryable	Description
`RADIANT_AI_6001`	404		Model not found
`RADIANT_AI_6002`	503		Model temporarily unavailable
`RADIANT_AI_6003`	503		Model overloaded
`RADIANT_AI_6004`	502		AI provider error
`RADIANT_AI_6005`	400		Context length exceeded
`RADIANT_AI_6006`	400		Content filtered by safety
`RADIANT_AI_6007`	400		Invalid AI request
`RADIANT_AI_6008`	500		Streaming error
`RADIANT_AI_6009`	504		AI request timeout
`RADIANT_AI_6010`	503		Model is cold (warming up)

Billing Errors (7xxx)

Errors related to billing, credits, and subscriptions.

Code	HTTP	Retryable	Description
`RADIANT_BILL_7001`	402		Insufficient credits
`RADIANT_BILL_7002`	402		Payment required
`RADIANT_BILL_7003`	402		Payment failed
`RADIANT_BILL_7004`	402		Subscription expired
`RADIANT_BILL_7005`	402		Subscription cancelled
`RADIANT_BILL_7006`	400		Invalid coupon code
`RADIANT_BILL_7007`	429		Usage quota exceeded

Storage Errors (8xxx)

Errors related to file storage.

Code	HTTP	Retryable	Description
`RADIANT_STOR_8001`	413		Storage quota exceeded
`RADIANT_STOR_8002`	413		File too large
`RADIANT_STOR_8003`	415		Invalid file type
`RADIANT_STOR_8004`	500		Upload failed
`RADIANT_STOR_8005`	404		File not found

Internal Errors (9xxx)

Internal server errors and system failures.

Code	HTTP	Retryable	Description
`RADIANT_INT_9001`	500		Internal server error
`RADIANT_INT_9002`	500		Database error
`RADIANT_INT_9003`	500		Cache error
`RADIANT_INT_9004`	500		Queue processing error
`RADIANT_INT_9005`	503		Service unavailable
`RADIANT_INT_9006`	502		Dependency failure
`RADIANT_INT_9007`	500		Configuration error
`RADIANT_INT_9008`	504		Request timeout

Usage in Code

TypeScript/JavaScript

```
import {
  ErrorCodes,
  RadiantError,
  createNotFoundError,
  createValidationError,
  isRetryableError
} from '@radiant/shared';

// Using factory functions (recommended)
throw createNotFoundError('User', userId);
throw createValidationError('Email is required', 'email');

// Direct construction
throw new RadiantError(ErrorCodes.AUTH_INVALID_TOKEN, 'Custom message', {
  details: { tokenPrefix: 'rad_...' },
  requestId: context.awsRequestId,
});
```

```
// Check if retryable
if (isRetryableError(error.code)) {
  // Implement retry logic
}
```

Response Format

```
// RadiantError automatically formats responses
const error = new RadiantError(ErrorCodes.RESOURCE_NOT_FOUND);
return error.toResponse();

// Returns:
// {
//   statusCode: 404,
//   headers: { 'Content-Type': 'application/json' },
//   body: '{"error":{"code":"RADIANT_RES_4001","message":"Resource not found.","category":"resource","
// }
```

Client Handling

Retry Logic

```
async function callWithRetry(fn: () => Promise<Response>, maxRetries = 3) {
  for (let i = 0; i < maxRetries; i++) {
    try {
      const response = await fn();
      if (response.ok) return response;

      const error = await response.json();
      if (!error.error.retryable) throw error;

      const retryAfter = response.headers.get('Retry-After') || '5';
      await sleep(parseInt(retryAfter) * 1000);
    } catch (e) {
      if (i === maxRetries - 1) throw e;
    }
  }
}
```

Error Display

```
function getUserMessage(error: RadiantError): string {
  // Error codes include user-friendly messages
  return error.message;
}

function shouldShowRetryButton(error: RadiantError): boolean {
  return error.retryable;
}
```

Adding New Error Codes

1. Add the code to `packages/shared/src/errors/codes.ts`:

```
export const ErrorCodes = {
  // ... existing codes
  MY_NEW_ERROR: 'RADIANT_CAT_NNNN',
} as const;
```

2. Add metadata:

```
export const ErrorCodeMetadata: Record<ErrorCode, {...}> = {
  // ... existing metadata
  [ErrorCodes.MY_NEW_ERROR]: {
    httpStatus: 400,
    category: 'category',
    retryable: false,
    userMessage: 'User-friendly error message.',
  },
};
```

3. Update this documentation.

See Also

- API Reference
- Contributing Guide
- Troubleshooting

RADIANT Testing Guide

Comprehensive guide for testing RADIANT components.

Overview

RADIANT uses a multi-layered testing strategy:

Layer	Tool	Location	Purpose
Unit Tests	Vitest	`**/__tests__/*.test.ts`	Test individual functions/components
Integration Tests	Vitest	`**/__tests__/*.integration.test.ts`	Test service interactions
E2E Tests	Playwright	`apps/admin-dashboard/e2e/`	Test user workflows
Swift Tests	XCTest	`apps/swift-deployer/Tests/`	Test Swift services

Quick Start

```
# Run all tests
pnpm test

# Run with coverage
pnpm test:coverage

# Run E2E tests
cd apps/admin-dashboard && pnpm test:e2e

# Run Swift tests
cd apps/swift-deployer && swift test
```

Unit Testing

Lambda Handler Tests

Tests for Lambda handlers are located in `__tests__/` directories:

```
packages/infrastructure/lambda/
  admin/
     __tests__/
```

```
          handler.test.ts
   billing/
       __tests__/
           handler.test.ts
   shared/
       __tests__/
           auth.test.ts
           errors.test.ts
           services.test.ts
```

Running Lambda Tests

```bash
cd packages/infrastructure

# Run all Lambda tests
pnpm test

# Run specific handler
pnpm test -- admin
pnpm test -- billing

# Run with coverage
pnpm test:coverage

# Watch mode
pnpm test:watch
```

Writing Lambda Tests

```typescript
import { describe, it, expect, vi, beforeEach } from 'vitest';
import type { APIGatewayProxyEvent, Context } from 'aws-lambda';

// Mock dependencies
vi.mock('../../shared/db', () => ({
  listTenants: vi.fn(),
  getTenantById: vi.fn(),
}));

import { handler } from '../handler';
import { listTenants, getTenantById } from '../../shared/db';

// Create mock context
const mockContext = {
  awsRequestId: 'test-request-id',
  functionName: 'test-handler',
  // ... other required fields
} as Context;

// Create mock event helper
function createMockEvent(overrides = {}): APIGatewayProxyEvent {
  return {
    httpMethod: 'GET',
    path: '/test',
    headers: { Authorization: 'Bearer test-token' },
    // ... default values
    ...overrides,
  };
}

describe('Handler', () => {
  beforeEach(() => {
    vi.clearAllMocks();
  });

  it('should return 200 for valid request', async () => {
    (listTenants as ReturnType<typeof vi.fn>).mockResolvedValue([]);

    const event = createMockEvent({ path: '/admin/tenants' });
    const result = await handler(event, mockContext);

    expect(result.statusCode).toBe(200);
  });
});
```

Shared Module Tests

Test shared utilities and services:

```
// packages/infrastructure/lambda/shared/__tests__/errors.test.ts
import { describe, it, expect } from 'vitest';
import {
  ValidationError,
  NotFoundError,
  isOperationalError,
  toAppError,
} from '../errors';

describe('Error Classes', () => {
  it('should create ValidationError with 400 status', () => {
    const error = new ValidationError('Invalid input');

    expect(error.statusCode).toBe(400);
    expect(error.code).toBe('VALIDATION_ERROR');
  });
});
```

E2E Testing (Admin Dashboard)

Setup

```
cd apps/admin-dashboard

# Install Playwright browsers
npx playwright install

# Run E2E tests
pnpm test:e2e
```

```
# Run with UI
pnpm test:e2e:ui

# Run specific test file
pnpm test:e2e -- dashboard.spec.ts
```

Test Structure

```
apps/admin-dashboard/e2e/
  dashboard.spec.ts      # Dashboard navigation tests
  deployment.spec.ts     # Deployment workflow tests
  fixtures/
      test-data.json     # Test fixtures
```

Writing E2E Tests

```
// apps/admin-dashboard/e2e/dashboard.spec.ts
import { test, expect } from '@playwright/test';

test.describe('Dashboard', () => {
  test.beforeEach(async ({ page }) => {
    // Mock authentication
    await page.addInitScript(() => {
      localStorage.setItem('auth_token', 'test-token');
      localStorage.setItem('user', JSON.stringify({
        id: 'test-user',
        email: 'test@example.com',
        role: 'admin',
      }));
    });
  });

  test('should display dashboard home', async ({ page }) => {
    await page.goto('/');
    await expect(page.getByRole('heading', { level: 1 }))
      .toContainText('Dashboard');
  });

  test('should navigate to models page', async ({ page }) => {
    await page.goto('/');
    await page.getByRole('link', { name: 'Models' }).click();
    await expect(page).toHaveURL('/models');
  });
});
```

Swift Testing

Test Structure

```
apps/swift-deployer/Tests/
  RadiantDeployerTests.swift          # Basic unit tests
  RadiantDeployerTests/
```

```
    E2ETests/
        DeploymentE2ETests.swift     # E2E workflow tests
    ServiceTests/
        LocalStorageManagerTests.swift
        CredentialServiceTests.swift
```

Running Swift Tests

```
cd apps/swift-deployer

# Run all tests
swift test

# Run specific test class
swift test --filter LocalStorageManagerTests

# Run with verbose output
swift test -v

# Generate coverage (requires llvm-cov)
swift test --enable-code-coverage
```

Writing Swift Tests

```swift
import XCTest
@testable import RadiantDeployer

final class LocalStorageManagerTests: XCTestCase {
    var storageManager: LocalStorageManager!

    override func setUp() {
        super.setUp()
        storageManager = LocalStorageManager.shared
    }

    override func tearDown() {
        // Cleanup
        super.tearDown()
    }

    func testSaveAndLoadConfiguration() async throws {
        // Given
        let key = "test_config"
        let config = TestConfiguration(name: "Test", value: 42)

        // When
        try await storageManager.save(config, forKey: key)
        let loaded: TestConfiguration? = try await storageManager.load(forKey: key)

        // Then
        XCTAssertNotNil(loaded)
        XCTAssertEqual(loaded?.name, "Test")
        XCTAssertEqual(loaded?.value, 42)
    }
}
```

Test Utilities

Mock Factories

Use the shared testing utilities:

```
import {
  createMockTenant,
  createMockUser,
  createMockApiKey,
  createMockChatRequest,
  createMockChatResponse,
  createMockApiGatewayEvent,
  createMockLambdaContext,
} from '@radiant/shared/testing';

// Create mock data
const tenant = createMockTenant({ name: 'Test Corp' });
const user = createMockUser({ tenantId: tenant.id });
const event = createMockApiGatewayEvent({
  httpMethod: 'POST',
  body: JSON.stringify({ model: 'gpt-4o' }),
});
```

Assertion Helpers

```
import {
  assertDefined,
  assertEqual,
  assertMatch,
  assertContains,
  assertThrows,
  waitFor,
  sleep,
} from '@radiant/shared/testing';

// Custom assertions
assertDefined(result, 'Result should not be null');
assertMatch(response.id, /^chatcmpl_/, 'Invalid response ID format');

// Wait for async condition
await waitFor(() => service.isReady(), { timeout: 5000 });
```

Mocking Guidelines

Database Mocking

```
vi.mock('../db/client', () => ({
  executeStatement: vi.fn(),
}));

import { executeStatement } from '../db/client';

// Mock return value
(executeStatement as ReturnType<typeof vi.fn>).mockResolvedValue({
  rows: [{ id: '123', name: 'Test' }],
});
```

AWS SDK Mocking

```
vi.mock('@aws-sdk/client-s3', () => ({
  S3Client: vi.fn().mockImplementation(() => ({
    send: vi.fn(),
  })),
  PutObjectCommand: vi.fn(),
}));
```

External API Mocking

```
vi.mock('node-fetch', () => ({
  default: vi.fn(),
}));

import fetch from 'node-fetch';

(fetch as ReturnType<typeof vi.fn>).mockResolvedValue({
  ok: true,
  json: () => Promise.resolve({ data: 'mocked' }),
});
```

CI/CD Integration

Tests run automatically in GitHub Actions:

```
# .github/workflows/ci.yml
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Setup Node.js
        uses: actions/setup-node@v4
      - name: Install dependencies
        run: pnpm install
      - name: Run tests
        run: pnpm test
        env:
          DATABASE_URL: postgres://test@localhost:5432/test
```

Coverage Requirements

- Minimum 80% coverage for new code

- Critical paths require 90%+ coverage
- All error handling paths must be tested

Best Practices

Do's

- Test behavior, not implementation
- Use descriptive test names
- Mock external dependencies
- Test error cases and edge cases
- Keep tests fast and isolated
- Use factory functions for test data

Don'ts

- Don't test private methods directly
- Don't share state between tests
- Don't test framework code
- Don't ignore flaky tests
- Don't hardcode test data

Test Naming Convention

```
describe('ServiceName', () => {
  describe('methodName', () => {
    it('should return X when given Y', () => {});
    it('should throw error when invalid input', () => {});
    it('should handle empty array gracefully', () => {});
  });
});
```

Debugging Tests

Vitest

```
# Run with verbose output
pnpm test -- --reporter=verbose

# Run single test
pnpm test -- -t "should return 200"

# Debug mode
node --inspect-brk node_modules/.bin/vitest run
```

Playwright

```
# Debug mode with browser
pnpm test:e2e -- --debug

# Generate trace on failure
```

```
pnpm test:e2e -- --trace on

# View trace
npx playwright show-trace trace.zip
```

Swift

```
# Run with verbose output
swift test -v

# Run single test
swift test --filter "testSaveAndLoadConfiguration"
```

See Also

- Contributing Guide
- Error Codes Reference
- API Reference

RADIANT Compliance Guide

Overview

This document outlines RADIANT's compliance posture for SOC 2, HIPAA, and GDPR requirements.

Compliance Matrix

Framework	Tier Required	Status
SOC 2 Type II	All tiers	Controls implemented
HIPAA	Tier 3+ (GROWTH)	BAA available
GDPR	All tiers (EU data)	DPA available
PCI DSS	N/A	Not applicable (no card data)

SOC 2 Controls

Trust Service Criteria

Security (Common Criteria)

Control	Implementation
CC1.1 - Board oversight	Documented security policies
CC2.1 - Communication	Security awareness training
CC3.1 - Risk assessment	Annual risk assessments
CC4.1 - Monitoring	CloudWatch, GuardDuty
CC5.1 - Logical access	IAM, Cognito, RLS
CC6.1 - System operations	Runbooks, on-call
CC7.1 - Change management	CI/CD, PR reviews
CC8.1 - Risk mitigation	WAF, rate limiting
CC9.1 - Entity risk	Vendor assessments

Availability

Control	Implementation
A1.1 - Capacity planning	Auto-scaling, monitoring
A1.2 - Environmental protection	Multi-AZ, DR procedures
A1.3 - Recovery	Backups, PITR, runbooks

Confidentiality

Control	Implementation
C1.1 - Data classification	PII tagging, encryption
C1.2 - Data disposal	Lifecycle policies

Evidence Collection

```
// Automated evidence collection
const auditLogs = {
  // All admin actions logged
  source: 'audit_logs table',
  retention: '7 years',

  // Access logs
  accessLogs: 'CloudWatch Logs',

  // Configuration changes
  configChanges: 'AWS Config',

  // Security events
  securityEvents: 'GuardDuty findings',
};
```

Annual Audit Checklist

- ☐ Access review completed
- ☐ Penetration test completed
- ☐ Vulnerability scan completed
- ☐ Security training completed
- ☐ Incident response test completed
- ☐ DR test completed
- ☐ Vendor assessments updated
- ☐ Policies reviewed and updated

HIPAA Compliance

Applicability

HIPAA compliance is available for Tier 3 (GROWTH) and above, which includes: - Encryption at rest (AES-256) - Encryption in transit (TLS 1.3) - Audit logging - Access controls - BAA with AWS

Technical Safeguards

Requirement	Implementation
Access Control (§164.312(a))	Cognito MFA, RLS, RBAC
Audit Controls (§164.312(b))	CloudTrail, audit_logs table
Integrity Controls (§164.312(c))	Checksums, versioning
Transmission Security (§164.312(e))	TLS 1.3, VPC endpoints

Administrative Safeguards

Requirement	Implementation
Security Officer	Designated in org
Workforce Training	Annual security training
Access Management	Quarterly access reviews
Incident Response	Documented procedures

Physical Safeguards

Handled by AWS: - Data center security - Device controls - Facility access

PHI Data Handling

```
-- PHI fields are encrypted at column level
CREATE TABLE patient_data (
  id UUID PRIMARY KEY,
  tenant_id UUID NOT NULL,
  -- PHI fields use additional encryption
  encrypted_data BYTEA NOT NULL,
  encryption_key_id VARCHAR(255) NOT NULL,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Enable RLS for tenant isolation
ALTER TABLE patient_data ENABLE ROW LEVEL SECURITY;
```

BAA Requirements

Before processing PHI: 1. Sign BAA with RADIANT 2. Enable HIPAA-eligible services only 3. Configure CloudTrail logging 4. Enable AWS Config 5. Review shared responsibility model

GDPR Compliance

Data Subject Rights

Right	Implementation
Right to Access	Data export API
Right to Rectification	Self-service + API
Right to Erasure	Deletion API + cascade
Right to Restrict	Processing flags
Right to Portability	JSON/CSV export
Right to Object	Consent management

Data Export (Right to Access)

```
// API endpoint for data export
// GET /api/v2/gdpr/export
async function exportUserData(userId: string): Promise<UserDataExport> {
  return {
    personalData: await getPersonalData(userId),
    activityLogs: await getActivityLogs(userId),
    preferences: await getPreferences(userId),
    exportedAt: new Date().toISOString(),
    format: 'JSON',
  };
}
```

Data Deletion (Right to Erasure)

```
// API endpoint for data deletion
// DELETE /api/v2/gdpr/delete
async function deleteUserData(userId: string): Promise<DeletionResult> {
  // Cascade delete all user data
  await deletePersonalData(userId);
  await deleteActivityLogs(userId);
  await deletePreferences(userId);
  await deleteApiKeys(userId);

  // Anonymize audit logs (retain for compliance)
  await anonymizeAuditLogs(userId);

  return {
    deletedAt: new Date().toISOString(),
    confirmation: generateDeletionCertificate(userId),
  };
}
```

Data Processing Agreement

DPA includes: - Nature and purpose of processing - Types of personal data - Categories of data subjects - Sub-processor list - Technical measures - Audit rights

Data Residency

Region	Data Location	Backup Location
EU	eu-west-1 (Ireland)	eu-central-1 (Frankfurt)
US	us-east-1 (Virginia)	us-west-2 (Oregon)
APAC	ap-northeast-1 (Tokyo)	ap-southeast-1 (Singapore)

```
// Enforce data residency
const dataResidency = {
  EU: ['eu-west-1', 'eu-central-1'],
  US: ['us-east-1', 'us-west-2'],
  APAC: ['ap-northeast-1', 'ap-southeast-1'],
};

// Route requests to appropriate region
function routeByResidency(tenantRegion: string): string {
  return dataResidency[tenantRegion][0];
}
```

Consent Management

```sql
-- Consent tracking table
CREATE TABLE consent_records (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID NOT NULL REFERENCES users(id),
  consent_type VARCHAR(50) NOT NULL,
  granted BOOLEAN NOT NULL,
  granted_at TIMESTAMPTZ,
  withdrawn_at TIMESTAMPTZ,
  ip_address INET,
  user_agent TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Consent types: marketing, analytics, essential, third_party
```

Data Classification

Classification Levels

Level	Description	Examples	Controls
Public	No restrictions	Marketing content	None
Internal	Business use	Metrics, configs	Access control
Confidential	Sensitive business	API keys, billing	Encryption, audit
Restricted	Highly sensitive	PHI, PII, credentials	Full controls

PII Fields

```typescript
// Fields classified as PII
const piiFields = [
  'email',
  'display_name',
  'phone_number',
  'ip_address',
  'user_agent',
  'billing_address',
  'payment_method',
];

// Automatic PII detection and tagging
function tagPiiFields(data: Record<string, unknown>): void {
  for (const field of piiFields) {
    if (data[field]) {
      // Tag for audit and retention policies
      data[`${field}_pii`] = true;
    }
  }
}
```

Encryption

At Rest

Data Type	Encryption	Key Management
Database	AES-256	AWS KMS
S3	AES-256	AWS KMS
Secrets	AES-256	Secrets Manager
Backups	AES-256	AWS KMS

In Transit

Connection	Protocol	Minimum Version
API	TLS	1.2 (1.3 preferred)
Database	TLS	1.2
Internal	TLS	1.2

Key Rotation

```
// Automatic key rotation
const kmsKey = new kms.Key(this, 'Key', {
  enableKeyRotation: true,  // Annual rotation
  rotationPeriod: cdk.Duration.days(365),
});
```

Audit Logging

What We Log

Event Type	Retention	Purpose
Authentication	2 years	Security
Authorization	2 years	Security
Data access	7 years	Compliance
Admin actions	7 years	Compliance
Configuration changes	7 years	Compliance
API requests	90 days	Operations

Log Format

```
{
  "timestamp": "2024-12-24T10:30:00Z",
  "event_type": "data_access",
  "actor": {
    "id": "user-123",
    "type": "admin",
    "ip": "192.168.1.100"
  },
  "resource": {
    "type": "model",
    "id": "model-456"
  },
  "action": "read",
  "outcome": "success",
  "metadata": {}
}
```

Log Protection

- Logs are immutable (write-once)
- Logs are encrypted at rest
- Access requires special IAM role
- Log deletion requires dual approval

Incident Response

Classification

Severity	Response Time	Examples
Critical	1 hour	Data breach, service down
High	4 hours	Attempted breach, partial outage
Medium	24 hours	Policy violation
Low	72 hours	Minor security event

Breach Notification

Jurisdiction	Requirement	Timeline
GDPR	DPA + affected users	72 hours
HIPAA	HHS + affected individuals	60 days
State laws	Varies by state	Varies

Vendor Management

Approved Sub-Processors

Vendor	Purpose	Location	DPA
AWS	Infrastructure	Global	Yes
OpenAI	AI provider	US	Yes
Anthropic	AI provider	US	Yes
Google Cloud	AI provider	Global	Yes

Vendor Assessment

Annual assessment includes: - Security questionnaire - SOC 2 report review - Penetration test results - Insurance verification

Contact

Role	Contact
Data Protection Officer	dpo@radiant.example.com
Security Team	security@radiant.example.com
Compliance Team	compliance@radiant.example.com

RADIANT Cost Optimization Guide

Overview

This guide provides strategies for optimizing AWS costs for the RADIANT platform while maintaining performance and reliability.

Current Architecture Costs

Estimated Monthly Costs by Tier

Tier	Infrastructure	Est. Monthly Cost
SEED (Dev)	Minimal	$50-150
STARTUP	Small production	$200-400
GROWTH	Self-hosted models	$1,000-2,500
SCALE	Multi-region	$4,000-8,000
ENTERPRISE	Global, full HA	$15,000-35,000

Cost Breakdown by Service

Service	% of Total	Optimization Potential
Aurora	30-40%	High
Lambda	15-25%	Medium
API Gateway	5-10%	Low
S3	5-10%	Medium
CloudFront	5-10%	Low
ElastiCache	10-15%	Medium
Other	10-15%	Varies

Optimization Strategies

1. Database Optimization

Aurora Serverless v2

```
// Use Serverless v2 for variable workloads
const cluster = new rds.DatabaseCluster(this, 'Database', {
  serverlessV2MinCapacity: 0.5,   // Scale to near-zero
  serverlessV2MaxCapacity: 16,    // Scale up when needed
});
```

Savings: 40-60% vs. provisioned instances for variable workloads

Reserved Instances (Steady Workloads)

```
# Purchase reserved capacity for predictable workloads
aws rds purchase-reserved-db-instances-offering \
  --reserved-db-instances-offering-id xxx \
  --db-instance-count 1
```

Savings: 30-60% for 1-3 year terms

Read Replicas Strategy

```
// Use read replicas only when needed
// Scale readers with traffic
readers: [
  rds.ClusterInstance.serverlessV2('reader', {
    scaleWithWriter: true,  // Auto-scale with primary
  }),
],
```

2. Lambda Optimization

Right-Size Memory

```
// Test different memory sizes to find optimal cost/performance
const memoryOptions = [256, 512, 1024, 2048];

// Use AWS Lambda Power Tuning tool
// https://github.com/alexcasalboni/aws-lambda-power-tuning
```

Function Type	Recommended Memory	Reason
Simple CRUD	256-512 MB	Light compute
API Router	512-1024 MB	Balanced
AI Processing	1024-2048 MB	Heavy compute

Provisioned Concurrency (Strategic)

```
// Only use for latency-critical functions
new lambda.Alias(this, 'LiveAlias', {
  aliasName: 'live',
  version: fn.currentVersion,
  provisionedConcurrentExecutions: 5,  // Keep 5 warm
});
```

Cost: ~$0.015/hour per provisioned instance **Use when:** P99 latency requirements < 200ms

ARM64 (Graviton2)

```
// 20% cheaper, often faster
const fn = new lambda.Function(this, 'Function', {
  architecture: lambda.Architecture.ARM_64,
  runtime: lambda.Runtime.NODEJS_20_X,
});
```

Savings: 20% on compute costs

3. S3 Optimization

Intelligent Tiering

```
const bucket = new s3.Bucket(this, 'Storage', {
  intelligentTieringConfigurations: [{
    name: 'auto-tier',
    archiveAccessTierTime: cdk.Duration.days(90),
    deepArchiveAccessTierTime: cdk.Duration.days(180),
  }],
});
```

Savings: Up to 95% for infrequently accessed data

Lifecycle Rules

```
const bucket = new s3.Bucket(this, 'Storage', {
  lifecycleRules: [
    // Move old versions to cheaper storage
    {
      noncurrentVersionTransitions: [
        {
          storageClass: s3.StorageClass.INFREQUENT_ACCESS,
          transitionAfter: cdk.Duration.days(30),
        },
        {
          storageClass: s3.StorageClass.GLACIER,
          transitionAfter: cdk.Duration.days(90),
        },
      ],
    },
    // Delete old logs
    {
      prefix: 'logs/',
      expiration: cdk.Duration.days(90),
    },
  ],
});
```

4. API Gateway Optimization

HTTP API vs REST API

```
// HTTP API is 70% cheaper than REST API
// Use when you don't need REST API features

// HTTP API: $1.00/million requests
// REST API: $3.50/million requests
```

Feature	REST API	HTTP API
Cost	\$3.50/M	\$1.00/M
Lambda integration	Yes	Yes
Request validation	Yes	No
API keys/usage plans	Yes	No
Caching	Yes	No

Caching

```
// Enable caching for GET endpoints
const method = resource.addMethod('GET', integration, {
  cacheKeyParameters: ['method.request.querystring.id'],
});

// Cache stage setting
stage.cacheClusterEnabled = true;
stage.cacheClusterSize = '0.5';  // 0.5 GB minimum
```

Note: Cache costs $0.02/hour (0.5 GB). Calculate break-even point.

5. CloudWatch Optimization

Log Retention

```
// Don't keep logs forever
new logs.LogGroup(this, 'LogGroup', {
  retention: logs.RetentionDays.ONE_MONTH,  // Adjust per environment
});
```

Environment	Retention	Reason
Development	7 days	Quick debugging
Staging	14 days	Testing cycles
Production	90 days	Compliance needs

Metric Filters vs. Logs Insights

```
// Use metric filters for known patterns
// Cheaper than running Logs Insights queries repeatedly

new logs.MetricFilter(this, 'ErrorMetric', {
  logGroup,
  metricNamespace: 'Radiant',
  metricName: 'Errors',
  filterPattern: logs.FilterPattern.literal('ERROR'),
});
```

6. ElastiCache Optimization

Reserved Nodes

```
# Purchase reserved nodes for production
aws elasticache purchase-reserved-cache-nodes-offering \
  --reserved-cache-nodes-offering-id xxx
```

Savings: 30-55% for 1-3 year terms

Right-Size Nodes

Use Case	Recommended	Memory
Development	cache.t3.micro	0.5 GB
Small Prod	cache.t3.small	1.4 GB
Medium Prod	cache.r6g.large	13 GB

Use Case	Recommended	Memory
Large Prod	cache.r6g.xlarge	26 GB

7. Data Transfer Optimization

Use VPC Endpoints

```
// Avoid NAT Gateway costs for AWS services
vpc.addInterfaceEndpoint('S3Endpoint', {
  service: ec2.InterfaceVpcEndpointAwsService.S3,
});

vpc.addInterfaceEndpoint('SecretsManagerEndpoint', {
  service: ec2.InterfaceVpcEndpointAwsService.SECRETS_MANAGER,
});
```

Savings: $0.045/GB saved vs. NAT Gateway

CloudFront for S3

```
// Serve S3 content through CloudFront
// Cheaper data transfer + better performance
const distribution = new cloudfront.Distribution(this, 'CDN', {
  defaultBehavior: {
    origin: new origins.S3Origin(bucket),
  },
});
```

Cost Monitoring

AWS Cost Explorer

```
# Get cost breakdown by service
aws ce get-cost-and-usage \
  --time-period Start=2024-12-01,End=2024-12-31 \
  --granularity MONTHLY \
  --metrics BlendedCost \
  --group-by Type=DIMENSION,Key=SERVICE
```

CloudWatch Billing Alerts

```
// Alert before surprise bills
new cloudwatch.Alarm(this, 'BillingAlarm', {
  metric: new cloudwatch.Metric({
    namespace: 'AWS/Billing',
    metricName: 'EstimatedCharges',
    dimensionsMap: { Currency: 'USD' },
    statistic: 'Maximum',
    period: cdk.Duration.hours(6),
  }),
  threshold: 1000,  // $1000 threshold
  evaluationPeriods: 1,
});
```

Cost Allocation Tags

```
// Tag all resources for cost tracking
cdk.Tags.of(this).add('Project', 'radiant');
cdk.Tags.of(this).add('Environment', environment);
cdk.Tags.of(this).add('CostCenter', 'platform');
```

Environment-Specific Recommendations

Development

- Use Aurora Serverless v2 (scales to zero)
- Minimal Lambda memory
- No provisioned concurrency
- Short log retention
- Single-AZ deployments

Target: < $100/month

Staging

- Aurora Serverless v2
- Moderate Lambda memory
- No provisioned concurrency
- 14-day log retention
- Single-AZ acceptable

Target: < $300/month

Production

- Aurora Serverless v2 or Reserved (if predictable)
- Right-sized Lambda memory
- Provisioned concurrency for critical paths
- 90-day log retention
- Multi-AZ required

Target: Optimize for reliability, then cost

Monthly Cost Review Checklist

- ☐ Review AWS Cost Explorer for anomalies
- ☐ Check for unused resources (idle RDS, orphan EBS)
- ☐ Review Lambda right-sizing opportunities
- ☐ Check S3 storage class distribution
- ☐ Review data transfer costs
- ☐ Validate reserved capacity utilization
- ☐ Update cost allocation tags
- ☐ Project next month's costs

Tools

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Compute Optimizer

- Lambda Power Tuning
- Infracost - Cost estimation for IaC

RADIANT Data Retention Policy

Overview

This document defines data retention periods and deletion procedures for all data stored in the RADIANT platform.

Retention Schedule

User Data

Data Type	Active Retention	Archive	Total Retention	Deletion
Account info	Active + 30 days	N/A	Account lifetime + 30 days	Automatic
Usage history	2 years	5 years	7 years	Automatic
Chat history	90 days	1 year	1 year	Automatic
Uploaded files	Active	30 days post-delete	Active + 30 days	On request
API keys	Active	N/A	Revoked + 90 days	Automatic

System Data

Data Type	Retention	Purpose	Deletion
Audit logs	7 years	Compliance	Automatic
Access logs	2 years	Security	Automatic
Error logs	90 days	Debugging	Automatic
Metrics	15 months	CloudWatch default	Automatic
Backups	35 days	Recovery	Automatic

Billing Data

Data Type	Retention	Purpose	Legal Basis
Invoices	7 years	Tax compliance	Legal requirement
Transactions	7 years	Financial audit	Legal requirement
Payment methods	Active	Processing	Contract
Receipts	7 years	Tax compliance	Legal requirement

Implementation

Database Retention

```
-- Automatic data cleanup job (runs daily)
CREATE OR REPLACE FUNCTION cleanup_expired_data()
RETURNS void AS $$
BEGIN
  -- Delete expired chat messages (90 days)
  DELETE FROM chat_messages
  WHERE created_at < NOW() - INTERVAL '90 days'
  AND archived = false;

  -- Archive chat messages older than 90 days
  UPDATE chat_messages
  SET archived = true, archived_at = NOW()
  WHERE created_at < NOW() - INTERVAL '90 days'
  AND archived = false;

  -- Delete archived messages older than 1 year
  DELETE FROM chat_messages
  WHERE archived = true
  AND archived_at < NOW() - INTERVAL '1 year';

  -- Delete revoked API keys (90 days after revocation)
  DELETE FROM api_keys
  WHERE revoked_at < NOW() - INTERVAL '90 days';

  -- Delete expired sessions
  DELETE FROM user_sessions
  WHERE expires_at < NOW();

  -- Log cleanup
  INSERT INTO system_jobs (job_name, completed_at, records_affected)
  VALUES ('cleanup_expired_data', NOW(),
    (SELECT count(*) FROM pg_stat_user_tables WHERE relname IN
      ('chat_messages', 'api_keys', 'user_sessions')));
END;
$$ LANGUAGE plpgsql;

-- Schedule daily at 3 AM UTC
SELECT cron.schedule('data-cleanup', '0 3 * * *', 'SELECT cleanup_expired_data()');
```

S3 Lifecycle Policies

```
const bucket = new s3.Bucket(this, 'Storage', {
  lifecycleRules: [
    // User uploads - delete 30 days after object deletion marker
    {
      id: 'delete-old-versions',
      noncurrentVersionExpiration: cdk.Duration.days(30),
    },

    // Temp files - delete after 7 days
    {
      id: 'cleanup-temp',
      prefix: 'temp/',
      expiration: cdk.Duration.days(7),
    },

    // Logs - transition to Glacier after 90 days, delete after 2 years
    {
      id: 'archive-logs',
      prefix: 'logs/',
      transitions: [
        {
          storageClass: s3.StorageClass.GLACIER,
          transitionAfter: cdk.Duration.days(90),
        },
      ],
      expiration: cdk.Duration.days(730), // 2 years
    },

    // Backups - delete after 35 days
    {
      id: 'cleanup-backups',
      prefix: 'backups/',
      expiration: cdk.Duration.days(35),
    },
  ],
});
```

CloudWatch Log Retention

```
// Set retention for all log groups
const logRetention: Record<string, logs.RetentionDays> = {
  // Application logs
  '/aws/lambda/radiant-*': logs.RetentionDays.THREE_MONTHS,

  // API Gateway logs
  '/aws/apigateway/radiant-*': logs.RetentionDays.THREE_MONTHS,

  // Database logs (longer for compliance)
  '/aws/rds/cluster/radiant-*': logs.RetentionDays.TWO_YEARS,

  // Audit logs (longest retention)
  '/radiant/audit/*': logs.RetentionDays.TEN_YEARS,
};
```

Data Deletion

User-Initiated Deletion

Account Deletion Flow

```
async function deleteUserAccount(userId: string): Promise<void> {
  // 1. Verify identity (MFA required)
  await verifyIdentity(userId);
  // 2. Cancel active subscriptions
  await cancelSubscriptions(userId);

  // 3. Export data (optional, user-requested)
  const exportUrl = await exportUserData(userId);

  // 4. Mark account for deletion (30-day grace period)
  await markForDeletion(userId, {
    scheduledAt: addDays(new Date(), 30),
    reason: 'user_requested',
  });

  // 5. Send confirmation email
  await sendDeletionConfirmation(userId, exportUrl);
}

// Actual deletion after grace period
async function executeAccountDeletion(userId: string): Promise<void> {
  // Delete in order (respect foreign keys)
  await deleteApiKeys(userId);
  await deleteChatHistory(userId);
  await deleteFiles(userId);
  await deletePreferences(userId);
  await deleteBillingHistory(userId); // Anonymize, don't delete
  await deleteAccount(userId);

  // Anonymize audit logs
  await anonymizeAuditLogs(userId);

  // Log deletion for compliance
  await logAccountDeletion(userId);
}
```

Data Categories Deleted

Category	Action	Timing
Profile	Delete	Immediate
Preferences	Delete	Immediate
Chat history	Delete	Immediate
Files	Delete	Immediate
API keys	Revoke + Delete	Immediate
Billing history	Anonymize	Immediate
Audit logs	Anonymize	Immediate
Backups	Excluded	Expires naturally

Administrative Deletion

```
// Bulk deletion for compliance (e.g., GDPR request)
async function adminBulkDelete(
  tenantId: string,
  options: {
    dataTypes: string[];
    olderThan: Date;
```

```typescript
    reason: string;
    approvedBy: string[];
  }
): Promise<DeletionReport> {
  // Require dual admin approval
  if (options.approvedBy.length < 2) {
    throw new Error('Dual admin approval required');
  }

  // Log the deletion request
  await logAdminAction({
    action: 'bulk_delete',
    tenantId,
    options,
  });

  // Execute deletion
  const results = await Promise.all(
    options.dataTypes.map(type =>
      deleteDataByType(tenantId, type, options.olderThan)
    )
  );

  return {
    requestId: generateRequestId(),
    deletedAt: new Date(),
    recordsDeleted: results.reduce((a, b) => a + b, 0),
    dataTypes: options.dataTypes,
  };
}
```

Tenant Offboarding

```typescript
async function offboardTenant(tenantId: string): Promise<void> {
  // 1. Export all data (required for compliance)
  const exportUrl = await exportTenantData(tenantId);

  // 2. Notify all users
  await notifyTenantUsers(tenantId, 'account_closing');

  // 3. Wait for grace period (30 days default)
  await scheduleTenantDeletion(tenantId, {
    gracePeriod: 30,
    exportUrl,
  });

  // 4. After grace period, delete all data
  // (Handled by scheduled job)
}
```

Legal Holds

Implementing a Legal Hold

```sql
-- Legal hold table
CREATE TABLE legal_holds (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID REFERENCES tenants(id),
  user_id UUID REFERENCES users(id),
  hold_type VARCHAR(50) NOT NULL, -- 'litigation', 'investigation', 'regulatory'
  description TEXT,
  started_at TIMESTAMPTZ DEFAULT NOW(),
  expires_at TIMESTAMPTZ,
  created_by UUID REFERENCES administrators(id),
  CONSTRAINT legal_holds_target CHECK (tenant_id IS NOT NULL OR user_id IS NOT NULL)
);

-- Prevent deletion of held data
CREATE OR REPLACE FUNCTION check_legal_hold()
RETURNS TRIGGER AS $$
BEGIN
  IF EXISTS (
    SELECT 1 FROM legal_holds
    WHERE (tenant_id = OLD.tenant_id OR user_id = OLD.user_id)
    AND (expires_at IS NULL OR expires_at > NOW())
  ) THEN
    RAISE EXCEPTION 'Cannot delete data under legal hold';
  END IF;
  RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Suspending Retention Policies

```typescript
// Suspend automatic deletion during legal hold
async function applyLegalHold(params: {
  holdId: string;
  scope: 'tenant' | 'user';
  targetId: string;
}): Promise<void> {
  // Update retention flags
  await updateRetentionPolicy(params.targetId, {
    suspended: true,
    holdId: params.holdId,
  });

  // Exclude from cleanup jobs
  await excludeFromCleanup(params.targetId);

  // Notify compliance team
  await notifyCompliance('legal_hold_applied', params);
}
```

Audit Trail

Retention Actions Log

```sql
-- Log all retention-related actions
CREATE TABLE retention_actions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  action_type VARCHAR(50) NOT NULL, -- 'delete', 'archive', 'export', 'hold'
  target_type VARCHAR(50) NOT NULL, -- 'user', 'tenant', 'data_type'
  target_id VARCHAR(255) NOT NULL,
  records_affected INTEGER,
  performed_by UUID,
  reason TEXT,
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Index for compliance queries
CREATE INDEX idx_retention_actions_date ON retention_actions(created_at);
CREATE INDEX idx_retention_actions_target ON retention_actions(target_type, target_id);
```

Compliance Reporting

```typescript
// Generate retention compliance report
async function generateRetentionReport(
  startDate: Date,
  endDate: Date
): Promise<RetentionReport> {
  return {
    period: { start: startDate, end: endDate },

    // Data deleted by type
    deletions: await getRetentionActions('delete', startDate, endDate),

    // Data archived
    archives: await getRetentionActions('archive', startDate, endDate),

    // Active legal holds
    legalHolds: await getActiveLegalHolds(),

    // Policy violations (data past retention not deleted)
    violations: await getRetentionViolations(),

    // User deletion requests
    userRequests: await getUserDeletionRequests(startDate, endDate),
  };
}
```

Verification

Monthly Retention Audit

- ☐ Verify cleanup jobs running successfully
- ☐ Check for retention policy violations

- ☐ Review legal holds status
- ☐ Verify S3 lifecycle policies active
- ☐ Confirm CloudWatch log retention settings
- ☐ Review user deletion requests processed
- ☐ Update retention schedule if needed

Compliance Queries

```
-- Find data past retention period
SELECT
  'chat_messages' as table_name,
  COUNT(*) as records,
  MIN(created_at) as oldest_record
FROM chat_messages
WHERE created_at < NOW() - INTERVAL '90 days'
AND archived = false

UNION ALL

SELECT
  'api_keys' as table_name,
  COUNT(*) as records,
  MIN(revoked_at) as oldest_record
FROM api_keys
WHERE revoked_at < NOW() - INTERVAL '90 days';
```

Contact

Role	Contact	Purpose
Data Protection Officer	dpo@radiant.example.com	GDPR requests
Legal	legal@radiant.example.com	Legal holds
Compliance	compliance@radiant.example.com	Audit questions

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

RADIANT Performance Guide

Overview

This guide covers performance optimization, caching strategies, and scalability considerations for the RADIANT platform.

Architecture Performance

Request Flow

```
Client → CloudFront → WAF → API Gateway → Lambda → Aurora
                                            ↓
                                         Redis Cache
```

Latency Targets

Component	Target	Max Acceptable
CloudFront edge	< 50ms	100ms
WAF processing	< 5ms	20ms
API Gateway	< 20ms	50ms
Lambda cold start	< 500ms	1000ms
Lambda execution	< 200ms	500ms
Database query	< 50ms	200ms
Total P95	**< 500ms**	**2000ms**

Caching Strategy

Multi-Layer Caching

```
                   CloudFront CDN
  TTL: 5m for static, 1m for API (with stale-while-revalidate)

                          ↓

                  API Gateway Cache
         TTL: 60s for GET endpoints

                          ↓

                    Redis Cache
```

```
      Session: 24h, Config: 5m, Translations: 1h
                             ↓
                     Aurora Database
            Query cache, Connection pooling
```

Cache Keys

```
// Session cache
`session:${tenantId}:${userId}` → TTL: 24h

// Configuration cache
`config:${tenantId}:${key}` → TTL: 5m
`config:global:${key}` → TTL: 5m

// Translation cache
`i18n:${language}:bundle` → TTL: 1h
`i18n:${language}:${key}` → TTL: 1h

// Model cache
`models:${tenantId}:list` → TTL: 5m
`models:${tenantId}:${modelId}` → TTL: 5m

// Rate limit cache
`ratelimit:${tenantId}:${endpoint}` → TTL: 1m
`ratelimit:ip:${ip}` → TTL: 5m
```

Cache Invalidation

```
// Pattern-based invalidation
await redis.del(`config:${tenantId}:*`);

// Event-driven invalidation
eventBridge.putEvents({
  Entries: [{
    Source: 'radiant.config',
    DetailType: 'ConfigUpdated',
    Detail: JSON.stringify({ tenantId, key }),
  }],
});
```

Database Optimization

Connection Pooling

```
// RDS Proxy configuration
const pool = {
  min: 2,
  max: 10,
  idleTimeoutMillis: 30000,
  connectionTimeoutMillis: 5000,
};
```

Query Optimization

```sql
-- Always use indexes
CREATE INDEX idx_models_tenant_status ON ai_models(tenant_id, status);
CREATE INDEX idx_transactions_tenant_date ON credit_transactions(tenant_id, created_at DESC);

-- Use covering indexes for common queries
CREATE INDEX idx_models_list ON ai_models(tenant_id, status, is_enabled)
  INCLUDE (display_name, category, input_cost_per_1k);

-- Partition large tables by date
CREATE TABLE audit_logs_2024_01 PARTITION OF audit_logs
  FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

RLS Performance

```sql
-- Set tenant context once per request
SET app.current_tenant_id = 'tenant-123';

-- All subsequent queries automatically filtered
SELECT * FROM models;  -- Implicitly filtered by RLS
```

Lambda Optimization

Cold Start Reduction

```typescript
// 1. Minimize dependencies
// 2. Use Lambda layers for shared code
// 3. Enable provisioned concurrency for critical functions

// Provisioned concurrency config
new lambda.Function(this, 'Router', {
  // ... config
  provisionedConcurrentExecutions: 10, // Keep 10 warm
});
```

Memory Optimization

```typescript
// Memory vs CPU tradeoff
// More memory = more CPU = faster execution

// Recommended settings by function type:
const memoryConfig = {
  router: 1024,      // Main API - balanced
  billing: 512,      // Light compute
  aiProxy: 2048,     // Heavy compute for AI
  migration: 256,    // Infrequent, light
};
```

Bundling

```typescript
// esbuild configuration for minimal bundle size
{
  bundle: true,
  minify: true,
  treeShaking: true,
  external: ['aws-sdk'], // Use Lambda runtime SDK
  target: 'node20',
}
```

Rate Limiting

Tier Limits

Tier	RPS	Burst	Daily
Free	10	20	1,000
Starter	50	100	10,000
Professional	100	200	50,000
Business	500	1,000	250,000
Enterprise	2,000	5,000	Unlimited

Implementation

```
// Token bucket algorithm in Redis
const rateLimiter = {
  async checkLimit(tenantId: string, limit: number): Promise<boolean> {
    const key = `ratelimit:${tenantId}`;
    const current = await redis.incr(key);

    if (current === 1) {
      await redis.expire(key, 1); // 1 second window
    }

    return current <= limit;
  }
};
```

Load Testing

Running Tests

```
# Install k6
brew install k6

# Run smoke test
k6 run --env BASE_URL=https://api-dev.radiant.example.com tests/load/k6-config.js

# Run with specific scenario
k6 run --env BASE_URL=https://api-dev.radiant.example.com \
  -e SCENARIO=load tests/load/k6-config.js
```

Performance Baselines

Metric	Baseline	Target
Throughput	500 RPS	2000 RPS
P50 Latency	100ms	50ms

Metric	Baseline	Target
P95 Latency	500ms	200ms
P99 Latency	1000ms	500ms
Error Rate	< 1%	< 0.1%

Scaling

Horizontal Scaling

Component	Scaling Method
Lambda	Automatic (up to account limit)
Aurora	Read replicas, Serverless v2
Redis	ElastiCache cluster mode
API Gateway	Automatic

Vertical Scaling

```
// Aurora Serverless v2 scaling
const database = new rds.DatabaseCluster(this, 'Database', {
  serverlessV2MinCapacity: 0.5,  // Minimum ACUs
  serverlessV2MaxCapacity: 16,   // Maximum ACUs
});

// Lambda memory scaling
const lambda = new lambda.Function(this, 'Function', {
  memorySize: 2048,  // More memory = more CPU
});
```

Monitoring

Key Metrics

```
// CloudWatch metrics to monitor
const metrics = {
  // Latency
  'AWS/ApiGateway/Latency': 'p95 < 500ms',
  'AWS/Lambda/Duration': 'p95 < 200ms',
  'AWS/RDS/ReadLatency': 'avg < 50ms',

  // Throughput
  'AWS/ApiGateway/Count': 'track trends',
  'AWS/Lambda/Invocations': 'track trends',

  // Errors
  'AWS/ApiGateway/5XXError': 'rate < 1%',
  'AWS/Lambda/Errors': 'rate < 1%',

  // Resources
  'AWS/Lambda/ConcurrentExecutions': '< 80% of limit',
  'AWS/RDS/CPUUtilization': '< 80%',
  'AWS/RDS/DatabaseConnections': '< 80% of max',
};
```

Alerting Thresholds

Metric	Warning	Critical
API P95 Latency	> 1s	> 3s
Error Rate	> 1%	> 5%
Lambda Concurrent	> 500	> 800
DB CPU	> 70%	> 85%
DB Connections	> 60%	> 80%

Best Practices

Do's

- Cache aggressively with proper invalidation
- Use connection pooling
- Minimize cold starts with provisioned concurrency
- Use read replicas for read-heavy workloads
- Implement circuit breakers for external services
- Use async processing for non-critical paths

Don'ts

- Don't make synchronous calls to external APIs in hot paths
- Don't use Lambda for long-running tasks (> 15 min)
- Don't store large objects in Redis
- Don't rely on API Gateway caching for dynamic data
- Don't skip database indexes

Troubleshooting

High Latency

1. Check Lambda cold starts (enable provisioned concurrency)
2. Check database query times (add indexes)
3. Check external API latency (add caching/circuit breaker)
4. Check connection pool exhaustion

High Error Rate

1. Check Lambda errors in CloudWatch Logs
2. Check database connection errors
3. Check rate limiting (429 errors)
4. Check WAF blocked requests

Scaling Issues

1. Check Lambda concurrent execution limit
2. Check database connection limit
3. Check API Gateway throttling
4. Check Redis memory usage

RADIANT AI Registry Seed Data System

Technical Documentation

Version: 4.18.1 | Last Updated: December 2024

Overview

The RADIANT Seed Data System manages versioned AI provider and model configurations that are used to populate the AI Registry during fresh installations. Seed data is stored separately from packages, can be versioned independently, and is selectable when building deployment packages.

Architecture

```
                        SEED DATA ARCHITECTURE


  config/seeds/
     registry.json          # Index of all seed versions
     v1/                    # Seed data version 1.0.0
        manifest.json      # Version metadata and stats
        providers.json     # 21 external providers
        external-models.json  # 50+ external models
        self-hosted-models.json  # 38 self-hosted models
        services.json      # 5 orchestration services
     v2/                    # Future seed versions...

  Build Time:

    build-package.sh    Select seed version   Include in package
    --seed-version 1


  Deploy Time (INSTALL only):

    DeploymentService   Read seeds from package   INSERT to database
    .executeInstall()
```

Critical Rules

Rule 1: NO HARDCODING IN DEPLOYER APP

The Swift Deployer app **MUST NOT** contain hardcoded lists of providers or models:

```swift
//  WRONG - Never do this
let providers = ["openai", "anthropic", "google", ...]

//  CORRECT - Fetch from Radiant API after deployment
let providers = try await radiantAPI.fetchProviders()
```

Rule 2: INSTALLER SEEDS, UPDATER PRESERVES

Mode	Seed Behavior
INSTALL	Seeds database with complete provider/model list
UPDATE	NEVER touches AI Registry - preserves admin customizations
ROLLBACK	Restores from snapshot - does not re-seed

Rule 3: ADMIN CONTROLS ALL

Everything in seed data is **editable by the administrator** post-deployment: - Enable/disable providers and models - Change pricing markup - Add new providers/models - Delete providers/models

---

Seed Data Structure

manifest.json

```json
{
  "version": "1.0.0",
  "name": "RADIANT AI Registry Seed Data",
  "description": "Complete provider and model seed data for fresh installations",
  "createdAt": "2024-12-25T00:00:00Z",
  "updatedAt": "2024-12-25T00:00:00Z",
  "compatibility": {
    "minRadiantVersion": "4.16.0",
    "maxRadiantVersion": "5.0.0"
  },
  "files": {
    "providers": "providers.json",
    "externalModels": "external-models.json",
    "selfHostedModels": "self-hosted-models.json",
    "services": "services.json"
  },
  "stats": {
    "externalProviders": 21,
    "externalModels": 50,
    "selfHostedModels": 38,
    "services": 5
  },
  "pricing": {
    "externalMarkup": 1.40,
    "selfHostedMarkup": 1.75
  }
}
```

providers.json

Each provider includes:

Field	Description
`id`	Unique identifier
`name`	Internal name
`displayName`	Human-readable name
`category`	Provider category (text_generation, image_generation, etc.)
`apiBaseUrl`	API endpoint
`authType`	Authentication type (bearer, api_key, iam)
`secretName`	AWS Secrets Manager path for API key
`features`	Supported features (streaming, vision, etc.)
`compliance`	Compliance certifications (SOC2, GDPR, HIPAA)
`rateLimit`	Rate limiting configuration

external-models.json

Each model includes:

Field	Description
`id`	Unique model identifier
`providerId`	Reference to provider
`modelId`	Provider's model ID
`litellmId`	LiteLLM routing ID
`category`	Model category
`capabilities`	Model capabilities
`contextWindow`	Max input tokens
`maxOutput`	Max output tokens
`pricing`	Cost per 1K tokens + markup
`minTier`	Minimum tier required

self-hosted-models.json

Each self-hosted model includes:

Field	Description
`id`	Unique model identifier
`instanceType`	SageMaker instance type
`thermal`	Thermal management config (COLD/WARM/HOT)
`license`	Open-source license
`pricing`	Hourly rate + per-unit pricing
`minTier`	Minimum tier required (typically 3+)

Building Packages with Seed Data

List Available Seed Versions

```
./tools/scripts/build-package.sh --list-seeds
```

Output:

```
Available Seed Data Versions:
  v1.0.0 - 21 providers, 50 external models, 38 self-hosted models
```

Build with Specific Seed Version

```
# Use default (latest) seed version
./tools/scripts/build-package.sh

# Use specific seed version
./tools/scripts/build-package.sh --seed-version 1
```

Package Manifest with Seed Data

The generated package manifest includes seed data information:

```
{
  "schemaVersion": "2.1",
  "package": {
    "version": "4.18.1"
  },
  "seedData": {
    "version": "1.0.0",
    "hash": "abc123...",
    "externalProviders": 21,
    "externalModels": 50,
    "selfHostedModels": 38,
    "services": 5
  },
  "installBehavior": {
    "seedAIRegistry": true
  },
  "updateBehavior": {
    "seedAIRegistry": false
  }
}
```

Seed Data Categories

External Providers (21)

Category	Providers
Text Generation	OpenAI, Anthropic, Google, xAI, DeepSeek, Mistral, Cohere
Image Generation	OpenAI Images, Stability AI, FLUX
Video Generation	Runway, Luma AI

Category	Providers
Audio	ElevenLabs, OpenAI Audio
Embeddings	OpenAI Embeddings, Voyage AI
Search	Perplexity
3D Generation	Meshy
Self-Hosted	SageMaker (internal)

External Models (50+)

Category	Example Models
Text	GPT-4o, Claude Sonnet 4, Gemini 2.0, Grok 3, DeepSeek R1
Reasoning	O1, O3 Mini, DeepSeek Reasoner
Code	Codestral
Image	DALL-E 3, Stable Diffusion 3, FLUX Pro
Video	Gen-3 Alpha, Ray 2
Audio	Whisper, TTS-1, Multilingual V2

Self-Hosted Models (38)

Category	Models
Vision Classification	EfficientNet, Swin Transformer, CLIP
Object Detection	YOLOv8 (Nano/Small/Medium/XLarge), Grounding DINO
Segmentation	SAM, SAM 2, MobileSAM
Speech	Whisper Large V3, Parakeet TDT
Scientific	AlphaFold 2, ESM-2
Medical	nnU-Net, MedSAM
Geospatial	Prithvi 100M/600M
3D	Nerfstudio
LLM	Mistral 7B, Llama 3 70B

Pricing Structure

External Providers

Default markup: **40% (1.40x)**

Example: GPT-4o - Provider cost: $0.0025/1K input, $0.01/1K output - Tenant cost: $0.0035/1K input, $0.014/1K output

Self-Hosted Models

Default markup: **75% (1.75x)**

Example: YOLOv8 Medium - Infrastructure cost: ~$2.47/hour + $0.005/image - Tenant cost: ~$4.32/hour + $0.00875/image

---

Creating New Seed Versions

1. Create Version Directory

```
mkdir config/seeds/v2
```

2. Create Required Files

- `manifest.json` - Version metadata
- `providers.json` - Provider definitions
- `external-models.json` - External model definitions
- `self-hosted-models.json` - Self-hosted model definitions
- `services.json` - Service definitions

3. Update Registry

Add new version to `config/seeds/registry.json`:

```
{
  "versions": [
    {
      "version": "2.0.0",
      "directory": "v2",
      "releaseDate": "2025-01-15",
      "status": "stable",
      "changelog": "Added new providers and models..."
    },
    // ... existing versions
  ]
}
```

4. Test Build

```
./tools/scripts/build-package.sh --seed-version 2
```

Database Seeding

During fresh installation, the DeploymentService generates SQL migrations from seed data:

```
-- Only runs if providers table is empty
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM providers LIMIT 1) THEN
        -- Insert providers
        INSERT INTO providers (...) VALUES (...);

        -- Insert external models
        INSERT INTO models (...) VALUES (...);

        -- Insert self-hosted models
        INSERT INTO self_hosted_models (...) VALUES (...);
    END IF;
END $$;
```

Key behaviors: - Uses `ON CONFLICT DO NOTHING` to preserve admin changes - Only runs on fresh install (empty database) - Logs completion with model counts

---

Swift Service API

SeedDataService

```
actor SeedDataService {
    /// List available seed versions
    func listAvailableSeedVersions() async throws -> [SeedDataInfo]

    /// Load complete seed data for a version
    func loadSeedData(version: String) async throws -> SeedData

    /// Generate SQL migration from seed data
    func generateSeedMigration(seedData: SeedData) -> String
}
```

Usage in DeploymentService

```
func executeInstall(...) async throws -> DeploymentExecutionResult {
    // Load seed data from package
    let seedData = try await seedDataService.loadSeedData(
        version: package.manifest.seedData?.version ?? "1.0.0"
    )

    // Generate and run seed migration
    let seedSQL = seedDataService.generateSeedMigration(seedData: seedData)
    try await runMigration(sql: seedSQL)
}
```

Related Documentation

- Deployer Architecture - Deployment modes and package management
- Deployer Admin Guide - User-facing deployment documentation
- API Reference - Provider and model API endpoints

RADIANT v4.17.0 - Troubleshooting Guide

Common Issues and Solutions

CDK Deployment Failures

Issue	Likely Cause	Solution
Bootstrap failed	Wrong account/region	Verify `aws sts get-caller-identity`
Stack timeout	Slow resource creation	Check CloudFormation events in AWS Console
Resource limit	Service quota exceeded	Request quota increase via AWS Service Quotas
IAM permission denied	Insufficient permissions	Ensure IAM user has AdministratorAccess
Circular dependency	Stack references	Check stack dependencies in CDK code
Asset upload failed	S3 bucket permissions	Verify CDK bootstrap bucket exists

Aurora Database Issues

Issue	Likely Cause	Solution
Connection refused	Security group rules	Verify Lambda SG can reach Aurora SG on port 5432
Authentication failed	Wrong credentials	Check Secrets Manager for correct credentials
Connection timeout	Missing VPC endpoints	Add RDS VPC endpoint to private subnets
Too many connections	Connection exhaustion	Use RDS Proxy or increase max_connections
Slow queries	Missing indexes	Run EXPLAIN ANALYZE and add appropriate indexes
RLS blocking access	Tenant ID not set	Ensure `app.current_tenant_id` is set in session

Lambda Function Errors

Issue	Likely Cause	Solution
Cold start > 10s	VPC attachment	Use provisioned concurrency for critical functions
Timeout	Slow downstream services	Increase timeout, check DB/API latency
Out of memory	Large payloads/responses	Increase memory allocation (also increases CPU)
Permission denied	IAM role misconfigured	Check Lambda execution role policies
Module not found	Missing dependency	Verify all dependencies in package.json
Handler not found	Incorrect handler path	Check function configuration in CDK

LiteLLM / ECS Issues

Issue	Likely Cause	Solution
503 Service Unavailable	ECS task unhealthy	Check ECS service events and task logs
Provider timeout	Invalid API key	Verify provider secrets in Secrets Manager
Rate limited	Too many requests	Implement exponential backoff retry
Wrong model response	Model misconfigured	Check config.yaml model mappings
Container crashes	Memory exhaustion	Increase task memory in CDK
No healthy targets	Health check failing	Verify health check endpoint returns 200

SageMaker Issues (Tier 3+)

Issue	Likely Cause	Solution
Endpoint failed to create	Insufficient capacity	Try different instance type or region
InvocationError	Model loading failed	Check CloudWatch logs for model errors
Slow cold start	Large model size	Use warm pools or smaller model variant
Capacity error	Instance quota reached	Request SageMaker quota increase
Timeout	Long inference time	Increase endpoint timeout or optimize model

Cognito Authentication Issues

Issue	Likely Cause	Solution
Invalid grant	Expired refresh token	Re-authenticate user

Issue	Likely Cause	Solution
User not confirmed	Email not verified	Check email or manually confirm user
MFA required	MFA not set up	Complete MFA setup flow
Invalid client	Wrong client ID	Verify app client ID in configuration
Callback URL mismatch	URL not whitelisted	Add URL to allowed callbacks in Cognito

Admin Dashboard Issues

Issue	Likely Cause	Solution
403 Forbidden	CloudFront OAC issue	Verify S3 bucket policy allows CloudFront
API calls fail	CORS configuration	Check API Gateway CORS settings
Login redirect loop	Cookie domain mismatch	Verify cookie domain matches site domain
Blank page	Build error	Check `next build` output for errors
Slow load	Large bundle size	Enable code splitting and lazy loading

---

Log Locations

Component	CloudWatch Log Group
API Gateway	`/aws/api-gateway/radiant-{env}-api`
Lambda Functions	`/aws/lambda/Radiant-{env}-*`
LiteLLM (ECS)	`/ecs/radiant-{env}-litellm`
SageMaker Endpoints	`/aws/sagemaker/Endpoints/radiant-*`
Aurora PostgreSQL	`/aws/rds/cluster/radiant-{env}/postgresql`
CloudFront	Standard CloudFront logs in S3

Viewing Logs

```
# Tail Lambda logs in real-time
aws logs tail /aws/lambda/Radiant-dev-router --follow

# View ECS logs
aws logs tail /ecs/radiant-dev-litellm --follow

# Search logs for errors
aws logs filter-log-events \
  --log-group-name /aws/lambda/Radiant-dev-router \
  --filter-pattern "ERROR" \
  --start-time $(date -d '1 hour ago' +%s)000
```

Health Check Endpoints

```
# Platform API health
curl https://api.YOUR_DOMAIN/health
# Expected: {"status":"healthy","version":"4.17.0"}

# LiteLLM health
curl https://api.YOUR_DOMAIN/v2/litellm/health
# Expected: {"status":"healthy"}

# Admin API health
curl https://admin-api.YOUR_DOMAIN/health

# Model registry status
curl https://api.YOUR_DOMAIN/v2/models/status
```

Emergency Procedures

Database Restore from Snapshot

```
# List available snapshots
aws rds describe-db-cluster-snapshots \
  --db-cluster-identifier radiant-prod-cluster \
  --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotCreateTime]' \
  --output table

# Restore from snapshot
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier radiant-prod-restored \
  --snapshot-identifier your-snapshot-id \
  --engine aurora-postgresql \
  --vpc-security-group-ids sg-xxx \
  --db-subnet-group-name radiant-prod-db-subnet
```

Point-in-Time Recovery

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-prod-cluster \
  --db-cluster-identifier radiant-prod-recovered \
  --restore-to-time "2024-12-20T10:00:00Z" \
  --vpc-security-group-ids sg-xxx \
  --db-subnet-group-name radiant-prod-db-subnet
```

Rollback CDK Deployment

```
# Rollback to previous deployment
cd packages/infrastructure
npx cdk deploy Radiant-prod-API \
  --context environment=prod \
  --context tier=3 \
  --rollback
```

Disable Problematic Model

```sql
-- Connect to Aurora and run:
UPDATE models
SET status = 'disabled',
    disabled_reason = 'Emergency disable due to errors',
    updated_at = NOW()
WHERE model_id = 'problematic-model-id';
```

Force Scale Down SageMaker

```bash
# Scale endpoint to 0 instances
aws sagemaker update-endpoint-weights-and-capacities \
  --endpoint-name radiant-prod-model-endpoint \
  --desired-weights-and-capacities '[{"VariantName":"AllTraffic","DesiredInstanceCount":0}]'
```

Performance Benchmarks

Metric	Target	Acceptable	Action if Exceeded
API Gateway p50 latency	< 50ms	< 100ms	Check Lambda cold starts
API Gateway p99 latency	< 200ms	< 500ms	Enable provisioned concurrency
Chat streaming start	< 500ms	< 1s	Check LiteLLM/provider latency
Admin dashboard load	< 2s	< 3s	Optimize bundle, enable CDN caching
Model warm-up time	< 3 min	< 5 min	Use larger instance or warm pools
Aurora query latency	< 10ms	< 50ms	Add indexes, optimize queries

---

Support Checklist

When reporting issues, include:

1. **Environment**: dev/staging/prod
2. **Tier**: 1-5
3. **Error message**: Full error text
4. **Request ID**: From response headers
5. **Timestamp**: When the error occurred
6. **Steps to reproduce**: What actions led to the error
7. **Relevant logs**: CloudWatch log excerpts

Useful AWS CLI Commands

```bash
# Check stack status
aws cloudformation describe-stacks --stack-name Radiant-dev-API \
  --query 'Stacks[0].StackStatus'

# List recent CloudFormation events
aws cloudformation describe-stack-events --stack-name Radiant-dev-API \
  --query 'StackEvents[0:10].[Timestamp,ResourceStatus,ResourceType,LogicalResourceId]' \
  --output table

# Check Lambda function configuration
aws lambda get-function-configuration --function-name Radiant-dev-router

# List ECS services
aws ecs list-services --cluster radiant-dev-cluster

# Describe ECS service
aws ecs describe-services --cluster radiant-dev-cluster \
  --services radiant-dev-litellm

# Check Secrets Manager secret
aws secretsmanager get-secret-value --secret-id radiant/dev/db-credentials \
  --query 'SecretString' --output text | jq .
```

RADIANT CDK Stack Dependencies

Technical Reference

Version: 4.18.1 | Last Updated: December 2024

This document defines the explicit dependency graph for RADIANT CDK stacks to ensure correct deployment ordering.

Stack Dependency Graph

```
                              NetworkStack
                               (Foundation)



                 DatabaseStack           SecurityStack            StorageStack



                                     AuthStack
                                     (Cognito)



                   APIStack              AdminStack            BillingStack



                                    AIStack
```

```
                    (Models/LLM)




      ThermalStack    PerceptionSvc    SageMaker
        (Scaling)        (Vision)      (Self-Host)
```

Stack Definitions

Layer 1: Foundation

Stack	Purpose	Exports
NetworkStack	VPC, Subnets, NAT Gateways	VPC ID, Subnet IDs, Security Group IDs

Layer 2: Core Infrastructure

Stack	Purpose	Dependencies	Exports
DatabaseStack	Aurora PostgreSQL, RDS Proxy	NetworkStack	Cluster ARN, Secret ARN, Proxy Endpoint
SecurityStack	KMS Keys, WAF, GuardDuty	NetworkStack	Key ARNs, WAF ACL ARN
StorageStack	S3 Buckets, CloudFront	NetworkStack	Bucket ARNs, Distribution ID

Layer 3: Authentication

Stack	Purpose	Dependencies	Exports
AuthStack	Cognito User/Identity Pools	Database, Security, Storage	Pool IDs, Client IDs

Layer 4: Application Services

Stack	Purpose	Dependencies	Exports
APIStack	API Gateway, Lambda handlers	Auth, Database, Security	API Endpoint, Lambda ARNs
AdminStack	Admin API, Dashboard hosting	Auth, Database, Security	Admin API Endpoint
BillingStack	Stripe integration, Usage tracking	Auth, Database	Billing API Endpoint

Layer 5: AI Services

Stack	Purpose	Dependencies	Exports
AIStack	LiteLLM, Model routing	API, Database, Security	AI Gateway Endpoint

Layer 6: Specialized Services

Stack	Purpose	Dependencies	Exports
ThermalStack	Model scaling, State management	AI, Database	Thermal API Endpoint
PerceptionStack	Computer vision pipeline	AI, Storage	Perception API Endpoint
SageMakerStack	Self-hosted model endpoints	AI, Network	Endpoint ARNs

---

CDK Implementation

Explicit Dependencies

```typescript
// packages/infrastructure/lib/main.ts

import { App } from 'aws-cdk-lib';

const app = new App();

// Layer 1
const networkStack = new NetworkStack(app, 'Network', { env });

// Layer 2
const databaseStack = new DatabaseStack(app, 'Database', {
  env,
  vpc: networkStack.vpc,
});
databaseStack.addDependency(networkStack);

const securityStack = new SecurityStack(app, 'Security', {
  env,
  vpc: networkStack.vpc,
});
securityStack.addDependency(networkStack);

const storageStack = new StorageStack(app, 'Storage', {
  env,
  vpc: networkStack.vpc,
});
storageStack.addDependency(networkStack);

// Layer 3
```

```
const authStack = new AuthStack(app, 'Auth', {
  env,
  database: databaseStack,
  security: securityStack,
  storage: storageStack,
});
authStack.addDependency(databaseStack);
authStack.addDependency(securityStack);
authStack.addDependency(storageStack);

// Layer 4
const apiStack = new APIStack(app, 'API', {
  env,
  auth: authStack,
  database: databaseStack,
  security: securityStack,
});
apiStack.addDependency(authStack);

// ... continue for remaining stacks
```

Cross-Stack References

```
// Example: APIStack referencing DatabaseStack exports

export class APIStack extends Stack {
  constructor(scope: Construct, id: string, props: APIStackProps) {
    super(scope, id, props);

    // Use exports from DatabaseStack
    const clusterArn = props.database.clusterArn;
    const secretArn = props.database.secretArn;

    // Create Lambda with database access
    const handler = new Function(this, 'ApiHandler', {
      environment: {
        AURORA_CLUSTER_ARN: clusterArn,
        AURORA_SECRET_ARN: secretArn,
      },
    });
  }
}
```

Deployment Order

Fresh Install

```
# Deploy in dependency order
cdk deploy NetworkStack
cdk deploy DatabaseStack SecurityStack StorageStack --parallel
cdk deploy AuthStack
cdk deploy APIStack AdminStack BillingStack --parallel
cdk deploy AIStack
cdk deploy ThermalStack PerceptionStack SageMakerStack --parallel
```

Update (with dependencies)

```
# CDK handles ordering automatically when using addDependency
cdk deploy --all
```

Selective Deployment

```
# Deploy specific stack and its dependencies
cdk deploy AIStack --require-approval never
```

Rollback Considerations

Stack	Rollback Safe	Notes
NetworkStack	Caution	May affect all dependent stacks
DatabaseStack	Caution	Requires DB snapshot for data preservation
SecurityStack	Safe	KMS keys have deletion protection
StorageStack	Caution	S3 buckets may have data
AuthStack	Safe	Cognito pools preserved
APIStack	Safe	Stateless Lambda functions
AdminStack	Safe	Stateless
BillingStack	Caution	May have pending transactions
AIStack	Safe	Stateless routing
ThermalStack	Safe	State in database
SageMakerStack	Caution	May have running endpoints

Validation Script

```
#!/bin/bash
# tools/scripts/validate-stack-deps.sh

echo "Validating CDK stack dependencies..."

# Check for circular dependencies
cdk synth --quiet 2>&1 | grep -i "circular" && {
    echo "  Circular dependency detected!"
    exit 1
}
```

```
# Verify deployment order
cdk diff --all 2>&1 | head -50

echo "  Stack dependencies validated"
```

Related Documentation

- Deployment Guide - Full deployment procedures
- Deployer Architecture - Package and deployment flow