Contents

SECTION 9: ASSEMBLY & DEPLOYMENT GUIDE (v2.2.0)

â• â

> **Dependencies:** ALL previous sections **Provides:** Verification steps, testing procedures, troubleshooting

---

RADIANT v2.2.0 - Prompt 9: Assembly & Deployment Guide

> **Prompt 9 of 9** | Target Size: ~15KB | Version: 3.7.0 | December 2024

OVERVIEW

This is the final prompt in the 9-part RADIANT series. This prompt provides:

1. **Project Assembly** - How to combine all components from Prompts 1-8
2. **Deployment Checklist** - Pre-deployment, deployment, and post-deployment steps
3. **Testing Procedures** - Integration tests, smoke tests, compliance verification
4. **Success Criteria** - What constitutes a successful deployment
5. **Troubleshooting Guide** - Common issues and solutions
6. **Version History** - Changelog and upgrade notes

PROMPT SERIES RECAP

Prompt	Component	Est. Size	Key Deliverables
1	Foundation & Swift App	~50KB	Monorepo structure, Swift macOS deployment app
2	CDK Infrastructure	~45KB	VPC, Aurora, DynamoDB, S3, KMS, WAF
3	CDK AI & API Stacks	~50KB	Cognito, LiteLLM, API Gateway, AppSync
4	Lambda Core	~60KB	Router, Chat, Models, Providers, PHI
5	Lambda Admin & Billing	~55KB	Invitations, Approvals, Metering, Billing
6	Self-Hosted Models	~75KB	30+ SageMaker models, Thermal states, Mid-level services
7	External Providers & DB	~85KB	21 providers, PostgreSQL schema, DynamoDB tables
8	Admin Dashboard	~85KB	Next.js 14 dashboard, all management UIs
9	**Assembly & Deployment**	**~15KB**	**This guide**

Total Implementation: ~520KB of implementation prompts

PART 1: PROJECT ASSEMBLY

1.1 Directory Structure After All Prompts

After executing Prompts 1-8 in sequence, your project should have this structure:

```
radiant/
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ README.md
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ package.json
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ pnpm-workspace.yaml
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ tsconfig.base.json
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ .gitignore
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ .nvmrc
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ packages/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ shared/                          # From Prompt 1
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ package.json
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ tsconfig.json
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ src/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ types/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ constants/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ utils/
Ã¢â€ â€š   Ã¢â€ â€š
```

```
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ infrastructure/                  # From Prompts 2-3
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ package.json
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ tsconfig.json
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ cdk.json
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ bin/
Ã¢â€ â€š       Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ radiant.ts
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ lib/
Ã¢â€ â€š           Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ stacks/
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ foundation-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ networking-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ security-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ data-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ storage-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ auth-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ ai-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ api-stack.ts
Ã¢â€ â€š           Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ admin-stack.ts
Ã¢â€ â€š           Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ constructs/
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ functions/                           # From Prompts 4-5
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ router/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ chat/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ models/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ providers/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ phi-service/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ admin-invitations/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ admin-approvals/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ billing-metering/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ thermal-manager/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ model-sync/
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ config/                              # From Prompts 6-7
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ litellm/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ config.yaml
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ self-hosted-config.yaml
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ models/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ external-providers.json
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ self-hosted-models.json
```

```
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ sagemaker/
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ endpoint-configs/
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ migrations/                          # From Prompt 7
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 001_initial_schema.sql
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 002_rls_policies.sql
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 003_model_registry.sql
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 004_billing_tables.sql
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ 005_admin_tables.sql
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ apps/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ swift-deployer/                     # From Prompt 1
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ RadiantAdmin.xcodeproj
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ RadiantAdmin/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ App/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ Views/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ Services/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ Models/
Ã¢â€ â€š   Ã¢â€ â€š
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ admin-dashboard/                   # From Prompt 8
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ package.json
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ next.config.js
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ tailwind.config.js
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ app/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ components/
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ lib/
Ã¢â€ â€š
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ docs/
    Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ architecture.md
    Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ api-reference.md
    Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ deployment-guide.md
    Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ troubleshooting.md
```

1.2 Assembly Verification

Run these commands to verify all components are in place:

```bash
# Navigate to project root
cd radiant

# Verify structure
echo "Checking project structure..."
ls -la packages/shared/src/types/
ls -la packages/infrastructure/lib/stacks/
ls -la functions/
ls -la apps/swift-deployer/
ls -la apps/admin-dashboard/app/
ls -la migrations/
```

```bash
# Install dependencies
pnpm install

# Build shared package
cd packages/shared && pnpm build && cd ../..

# Verify CDK synthesizes
cd packages/infrastructure
npx cdk synth --context environment=dev --context tier=1
cd ../..

# Build admin dashboard
cd apps/admin-dashboard && pnpm build && cd ../..

echo "Assembly verification complete!"
```

PART 2: DEPLOYMENT CHECKLIST

2.1 Pre-Deployment Requirements

AWS Account Setup

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

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`
Swift	5.9+	`swift --version`

AWS Service Quotas Verify these quotas before deployment (request increases if needed):

Service	Quota	Required Minimum
VPC	VPCs per region	5
Aurora	DB clusters	3 (per environment)
Lambda	Concurrent executions	1,000
API Gateway	APIs per region	10
SageMaker	Endpoint instances	20 (Tier 3+)
Secrets Manager	Secrets	100
KMS	CMKs	50

2.2 Deployment Phases

Phase 1: Bootstrap & Foundation (15-25 minutes)

```
# 1. Bootstrap CDK (one-time per account/region)
cd packages/infrastructure
cdk bootstrap aws://ACCOUNT_ID/us-east-1 --qualifier radiant

# 2. Deploy Foundation Stack
cdk deploy Radiant-dev-Foundation \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 3. Deploy Networking Stack
cdk deploy Radiant-dev-Networking \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] S3 deployment bucket created - [ ] VPC created with correct CIDR - [ ] Subnets in 3 AZs - [ ] NAT Gateway(s) active - [ ] VPC endpoints created

Phase 2: Security & Data (15-25 minutes)

```
# 4. Deploy Security Stack
cdk deploy Radiant-dev-Security \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 5. Deploy Data Stack
cdk deploy Radiant-dev-Data \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

```
# 6. Deploy Storage Stack
cdk deploy Radiant-dev-Storage \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] KMS CMK created - [ ] Secrets Manager secrets created - [ ] Aurora cluster available - [ ] DynamoDB tables created - [ ] S3 buckets created with encryption

Phase 3: Auth & AI (20-30 minutes)

```
# 7. Deploy Auth Stack
cdk deploy Radiant-dev-Auth \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 8. Deploy AI Stack
cdk deploy Radiant-dev-AI \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] Cognito User Pool created - [ ] Cognito Admin Pool created - [ ] LiteLLM ECS service running - [ ] SageMaker endpoints active (Tier 3+)

Phase 4: API & Admin (15-20 minutes)

```
# 9. Deploy API Stack
cdk deploy Radiant-dev-API \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 10. Deploy Admin Stack
cdk deploy Radiant-dev-Admin \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] API Gateway deployed - [ ] AppSync API created - [ ] Lambda functions deployed - [ ] CloudFront distribution active - [ ] Admin dashboard accessible

Phase 5: Database Migrations (5-10 minutes)

```
# Run migrations
cd ../../
npm run db:migrate -- --environment=dev
```

```
# Verify schema
npm run db:verify -- --environment=dev
```

Verification: - [ ] All migrations applied - [ ] RLS policies active - [ ] Indexes created - [ ] Seed data loaded

2.3 Post-Deployment Configuration

Create First Super Admin

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

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

Configure AI Providers

1. Navigate to Admin Dashboard Ã¢â€ â€™ Providers
2. Add API keys for each external provider:
    - OpenAI
    - Anthropic
    - Google AI
    - xAI (Grok)
    - DeepSeek
    - Others as needed
3. Verify provider connectivity with test requests

Set Pricing Configuration

1. Navigate to Admin Dashboard Ã¢â€ â€™ Billing Ã¢â€ â€™ Pricing
2. Review default markup:
    - External providers: 40%
    - Self-hosted models: 75%
3. Adjust per-model pricing if needed
4. Configure billing thresholds and alerts

PART 3: TESTING PROCEDURES

3.1 Smoke Tests

Run immediately after deployment:

```
# API Health Check
curl https://YOUR_API_ENDPOINT/health
# Expected: {"status":"healthy","version":"2.2.0"}

# GraphQL Introspection
curl -X POST https://YOUR_GRAPHQL_ENDPOINT/graphql \
  -H "Content-Type: application/json" \
  -d '{"query":"{ __schema { types { name } } }"}'

# LiteLLM Health
curl https://YOUR_LITELLM_ENDPOINT/health
# Expected: {"status":"healthy"}

# Admin Dashboard
curl -I https://YOUR_ADMIN_DOMAIN
# Expected: HTTP/2 200
```

3.2 Integration Tests

```
# Run full integration test suite
npm run test:integration -- --environment=dev

# Individual test suites
npm run test:auth        # Authentication flows
npm run test:models      # Model registry CRUD
npm run test:chat        # Chat completions
npm run test:billing     # Metering and billing
npm run test:admin       # Admin workflows
```

3.3 End-to-End Test Scenarios

Scenario 1: User Registration Ã¢â€ â€™ Chat Completion

1. Register new user via Cognito
2. Verify email and set MFA
3. Login and get access token
4. Create chat session
5. Send message with model selection
6. Verify response received
7. Verify usage metered

Scenario 2: Admin Invitation Ã¢â€ â€™ Two-Person Approval

1. Super admin creates invitation
2. Invitee receives email
3. Invitee accepts and creates account
4. Invitee logs in to admin dashboard
5. Invitee requests production action
6. First admin approves

7. Second admin approves
8. Action executes

Scenario 3: Self-Hosted Model Warm-Up (Tier 3+)

1. Verify model in COLD state
2. Send warm-up request
3. Verify transition to WARM state
4. Monitor endpoint scaling
5. Verify inference works
6. Test auto-cooling after inactivity

3.4 Compliance Verification

HIPAA Technical Safeguards

Control	Test	Pass Criteria
Access Control	Verify MFA required	All production users have MFA
Audit Controls	Check CloudTrail	All API calls logged
Integrity	Verify log integrity	File validation enabled
Transmission	Test TLS	TLS 1.3 only
Encryption	Check KMS	Per-tenant CMKs active
PHI Handling	Test sanitization	PHI redacted correctly

SOC 2 Controls

Criterion	Test	Pass Criteria
CC1	Policy review	Policies documented
CC2	Communication	Alert channels configured
CC3	Risk assessment	Findings documented
CC4	Monitoring	Dashboards active
CC5	Control activities	Approvals working
CC6	Logical access	RLS verified
CC7	System operations	Health checks passing
CC8	Change management	CI/CD with approvals
CC9	Risk mitigation	Backups verified

PART 4: SUCCESS CRITERIA

4.1 Deployment Success Criteria

All of the following must be TRUE for a successful deployment:

Infrastructure

- ☐ All CDK stacks deployed without errors
- ☐ VPC with correct CIDR and subnets
- ☐ Aurora cluster status: available
- ☐ All DynamoDB tables active
- ☐ S3 buckets with encryption enabled
- ☐ KMS CMKs with rotation enabled

Authentication

- ☐ Cognito User Pool created
- ☐ Cognito Admin Pool with MFA required (production)
- ☐ At least one super admin created
- ☐ Admin can log in successfully

AI Services

- ☐ LiteLLM ECS service: RUNNING
- ☐ At least one external provider configured
- ☐ Test completion returns valid response
- ☐ (Tier 3+) SageMaker endpoints: InService

API Layer

- ☐ API Gateway: deployed
- ☐ AppSync API: active
- ☐ /health returns 200
- ☐ GraphQL introspection works

Admin Dashboard

- ☐ CloudFront distribution: Deployed
- ☐ Dashboard loads without errors
- ☐ Navigation works
- ☐ API calls succeed

Database

- ☐ All migrations applied
- ☐ RLS policies active
- ☐ Test queries succeed
- ☐ No orphaned data

Monitoring

- ☐ CloudWatch alarms configured
- ☐ CloudTrail logging active
- ☐ GuardDuty enabled
- ☐ Security Hub enabled

4.2 Performance Benchmarks

Metric	Target	Acceptable
API Gateway latency (p50)	< 50ms	< 100ms
API Gateway latency (p99)	< 200ms	< 500ms
Chat completion (streaming start)	< 500ms	< 1s
Admin dashboard load	< 2s	< 3s
Model warm-up time	< 3 min	< 5 min
Aurora query latency	< 10ms	< 50ms

PART 5: TROUBLESHOOTING GUIDE

5.1 Common Issues

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
Resource limit	Service quota exceeded	Request quota increase
IAM permission denied	Insufficient permissions	Verify AdministratorAccess
Circular dependency	Stack references	Check stack dependencies

Aurora Connection Issues

Issue	Likely Cause	Solution
Connection refused	Security group	Verify Lambda SG can reach Aurora SG
Authentication failed	Wrong credentials	Check Secrets Manager
Timeout	VPC endpoints missing	Add RDS VPC endpoint
Too many connections	Connection pooling	Use RDS Proxy

Lambda Errors

Issue	Likely Cause	Solution
Cold start > 10s	VPC configuration	Use provisioned concurrency
Timeout	Slow downstream	Increase timeout, check dependencies
Out of memory	Large payloads	Increase memory allocation
Permission denied	IAM role	Check Lambda execution role

LiteLLM Issues

Issue	Likely Cause	Solution
503 Service Unavailable	ECS not healthy	Check ECS service events
Provider timeout	API key invalid	Verify provider secrets
Rate limited	Too many requests	Implement retry with backoff
Wrong model response	Model misconfigured	Check litellm config.yaml

SageMaker Issues (Tier 3+)

Issue	Likely Cause	Solution
Endpoint failed	Out of memory	Use larger instance type
Slow cold start	Model too large	Use warm pool or smaller model
InvocationError	Model bug	Check CloudWatch logs
Capacity error	Insufficient instances	Request quota increase

5.2 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	/aws/rds/cluster/radiant-{env}/postgresql
CloudFront	/aws/cloudfront/radiant-{env}-admin

5.3 Health Check Endpoints

```
# Platform health
curl https://api.YOUR_DOMAIN.com/health

# LiteLLM health
curl https://api.YOUR_DOMAIN.com/v2/litellm/health

# Admin API health
curl https://admin-api.YOUR_DOMAIN.com/health

# Model registry status
curl https://api.YOUR_DOMAIN.com/v2/models/status
```

5.4 Emergency Procedures

Database Restore

```
# List available snapshots
aws rds describe-db-cluster-snapshots \
  --db-cluster-identifier radiant-prod-cluster

# Restore from snapshot
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier radiant-prod-restored \
  --snapshot-identifier your-snapshot-id \
  --engine aurora-postgresql

# Or use 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"
```

Rollback Deployment

```
# Rollback to previous CDK deployment
cdk deploy Radiant-prod-API \
  --context environment=prod \
  --context tier=3 \
  --rollback
```

Disable Problematic Model

```
-- In Aurora
UPDATE models
SET status = 'disabled',
    disabled_reason = 'Emergency disable due to errors'
WHERE model_id = 'problematic-model-id';
```

PART 6: VERSION HISTORY

v2.2.0 (December 2024) - Current

New Features: - Dynamic AI Model Registry with database-driven discovery - 21 external AI provider integrations - 30+ self-hosted models across 9 categories - 5 mid-level orchestration services - Thermal state management (OFF/COLD/WARM/HOT/AUTOMATIC) - Administrator invitation system with email notifications - Two-person approval workflow for production deployments - Configurable PHI sanitization with HIPAA compliance - Multi-region deployment (US/EU/APAC) - Comprehensive billing and metering system

Architecture: - Unified Swift macOS deployment application - Next.js 14 admin dashboard - AWS CDK infrastructure as code - 9-prompt implementation structure

Breaking Changes: - PHI configuration now requires explicit setup - Admin pool separate from user pool - New database schema with RLS

v2.1.0 (December 2024)

- Added admin dashboard web application
- Implemented two-person approval workflow
- Enhanced billing service with invoicing

v2.0.0 (December 2024)

- Major architectural refactor
- Added PHI sanitization service
- Implemented compliance testing framework
- Multi-app support from single deployment

v1.1.0 (December 2024)

- Added media handling (images, video, audio, 3D)
- Implemented 20+ external provider integrations
- Enhanced metering and billing

v1.0.0 (December 2024)

- Initial release
- Basic Swift deployment app
- Core CDK infrastructure
- LiteLLM integration

DEPLOYMENT TIMELINE

Phase	Duration	Activities
Pre-deployment	1-2 hours	Account setup, quotas, certificates
Bootstrap	2-5 min	CDK bootstrap, S3 buckets
Foundation/Networking	10-15 min	VPC, subnets, NAT gateways
Security/Data	15-20 min	KMS, Secrets, Aurora, DynamoDB
Storage	5-10 min	S3 buckets, lifecycle rules
Auth	10-15 min	Cognito pools, groups
AI	15-30 min	LiteLLM, SageMaker (if Tier 3+)
API	10-15 min	API Gateway, AppSync, Lambda
Admin	10-15 min	CloudFront, dashboard sync
Migrations	5-10 min	Schema, RLS, seed data
Configuration	15-30 min	First admin, providers, pricing
Testing	30-60 min	Smoke tests, integration tests
Total	**2-4 hours**	**Complete deployment**

ESTIMATED COSTS BY TIER

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Foundation	$45	$120	$350	$1,200	$4,500
AI/API	$40	$135	$625	$2,250	$9,000
Self-Hosted	$0	$0	$500	$2,000	$8,000
Total/Month	**~$85**	**~$255**	**~$1,475**	**~$5,450**	**~$21,500**

Costs are estimates and vary based on usage, region, and actual resource consumption.

FINAL NOTES

Support Resources

- **AWS Documentation**: https://docs.aws.amazon.com
- **CDK Documentation**: https://docs.aws.amazon.com/cdk
- **LiteLLM Documentation**: https://docs.litellm.ai
- **Next.js Documentation**: https://nextjs.org/docs

Recommended Monitoring Setup

1. **CloudWatch Dashboards** - Create dashboards for each component
2. **Alarms** - Set up alarms for error rates, latency, costs
3. **X-Ray** - Enable distributed tracing
4. **Cost Explorer** - Set up daily cost reports and anomaly detection

Security Best Practices

1. Rotate AWS access keys every 90 days
2. Enable MFA on root account
3. Use separate AWS accounts for prod/staging/dev
4. Regular security audits with AWS Inspector
5. Quarterly penetration testing

End of Prompt 9: Assembly & Deployment Guide RADIANT v2.2.0 - December 2024

CONGRATULATIONS! Ã°Å¸Å½â€°

You have completed the full RADIANT v2.2.0 implementation series.

What you've built: - Production-grade multi-tenant SaaS platform - 21 external AI provider integrations - 30+ self-hosted AI models - HIPAA and SOC 2 compliant infrastructure - Multi-region global deployment - Complete admin management system - Comprehensive billing and metering

Next Steps: 1. Deploy to staging and run full test suite 2. Configure all production providers 3. Complete compliance verification 4. Deploy to production with two-person approval 5. Monitor and iterate

Thank you for building with RADIANT!

â• â

END OF SECTION 9

â• â

â• â

APPENDIX: IMPLEMENTATION VERIFICATION CHECKLIST

â• â

Pre-Implementation

- ☐ Node.js 20+ installed (`node --version`)
- ☐ pnpm 8+ installed (`pnpm --version`)
- ☐ AWS CLI configured (`aws sts get-caller-identity`)
- ☐ Xcode 15+ installed (for Swift app)
- ☐ AWS CDK CLI installed (`cdk --version`)

Section 0: Shared Types

- ☐ `packages/shared/` directory created
- ☐ All type files implemented
- ☐ All constant files implemented
- ☐ All utility files implemented
- ☐ `pnpm build` succeeds in packages/shared
- ☐ `dist/` directory contains .js and .d.ts files

Section 1: Foundation & Swift App

- ☐ Root package.json configured
- ☐ Swift app compiles in Xcode
- ☐ Credentials can be saved/loaded
- ☐ Bundle scripts execute successfully

Section 2: CDK Infrastructure

- ☐ Infrastructure package imports from @radiant/shared
- ☐ NO duplicate tier/region configs created
- ☐ `cdk synth` succeeds for all stacks
- ☐ Foundation stack deploys
- ☐ Networking stack deploys
- ☐ Security stack deploys
- ☐ Data stack deploys

- ☐ Storage stack deploys

Section 3: CDK AI & API

- ☐ Auth stack deploys
- ☐ AI stack deploys
- ☐ API stack deploys
- ☐ Admin stack deploys
- ☐ Cognito user pool created
- ☐ API Gateway accessible

Section 4: Lambda Core

- ☐ All core Lambda functions implemented
- ☐ Functions import types from @radiant/shared
- ☐ Health endpoint returns 200

Section 5: Lambda Admin & Billing

- ☐ Admin Lambda functions implemented
- ☐ Uses canonical table names (administrators, invitations)
- ☐ Billing Lambda functions implemented

Section 6: Self-Hosted Models

- ☐ SageMaker endpoint configs created
- ☐ Thermal state management working
- ☐ Mid-level services defined

Section 7: Database

- ☐ All migrations in `migrations/` directory
- ☐ Migrations use canonical table names
- ☐ Migrations run successfully
- ☐ RLS policies applied

Section 8: Admin Dashboard

- ☐ Dashboard imports types from @radiant/shared
- ☐ `pnpm build` succeeds
- ☐ Dashboard accessible via CloudFront
- ☐ All pages render correctly

Section 9: Verification

- ☐ All stacks deployed
- ☐ Health checks pass
- ☐ First admin can be created
- ☐ API calls succeed
- ☐ Dashboard fully functional

Quick Domain Replacement

After cloning/creating the project, run:

```
# macOS
find . -type f \( -name "*.ts" -o -name "*.tsx" -o -name "*.json" -o -name "*.swift" -o -name
  -not -path "./node_modules/*" \
  -exec sed -i '' 's/YOUR_DOMAIN\.com/zynapses.com/g' {} \;

# Linux
find . -type f \( -name "*.ts" -o -name "*.tsx" -o -name "*.json" -o -name "*.swift" -o -name
  -not -path "./node_modules/*" \
  -exec sed -i 's/YOUR_DOMAIN\.com/zynapses.com/g' {} \;
```

Replace `YOUR_DOMAIN.com` with your actual domain.