(DVN-2025-12-19-TJM) Best CI-CD Pipeline Decision [ALL].md

Metadata

- **App**: [ALL]
- **Type**: [TASK]
- **Issue #**: [Launch]
- **Created**: [2025-12-19]
- **By**: [TJM and Cursor Composer]
- **Status**: Approved

Executive Summary

After evaluating AWS ECS vs Render.io for our CI/CD pipeline requirements, **Render.io has been selected** as the deployment platform. This decision is based on Render.io's built-in GitHub integration, automatic deployments that match our Git workflow exactly, lower initial costs ($57-58/month total vs ~$101/month for ECS), and minimal setup time (15 minutes vs 2-4 hours). Our existing GitHub Actions workflows will continue to handle CI (testing, linting), while Render.io handles CD (deployment) automatically.

Environment Strategy:

- **Production:** `main` branch → Starter Plus ($15/month) → S3 `/prod/` prefix
- **Beta:** `release/*` branches → Free tier ($0/month, < 24 concurrent users) → S3 `/beta/` prefix
- **Alpha:** `develop` branch → Free tier ($0/month) → S3 `/alpha/` prefix

Email Service: Mailgun Foundation plan ($35/month for 50,000 emails) handles all app-generated emails across all environments.

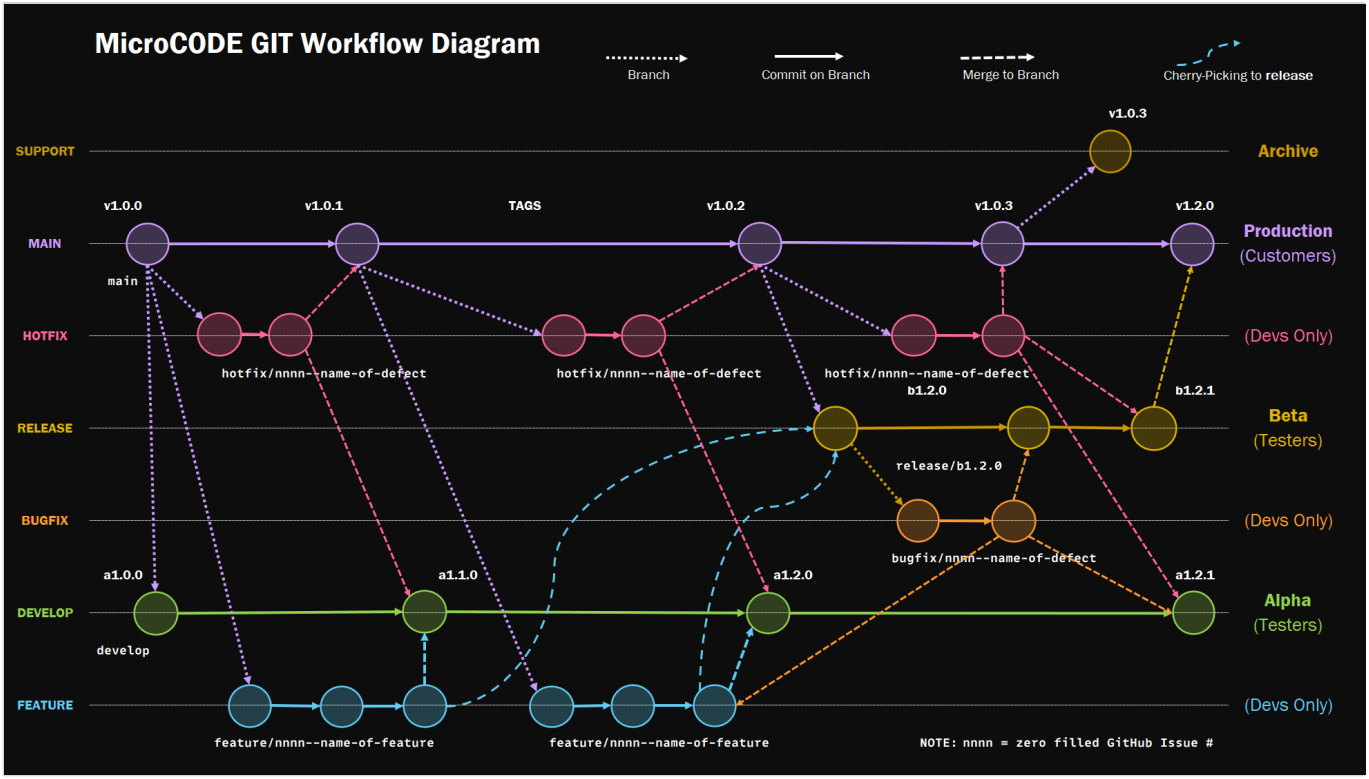
Git Workflow & Auto-Deployment Strategy

Critical Understanding: All branch types (`hotfix/*`, `bugfix/*`, `feature/*`) are developed and initially tested on the developer's local machine. When their Pull Request (PR) is merged, they automatically trigger deployments to their target environments.

This is an overview of MicroCODE's Workflow and App Versioning...

`xM.F.H`

- 'x' = Branch: "v" = Production (main branch), "b" = Beta (release/*), "a" = Alpha (develop)
- `M` = Major Release (architecture/design/API change)
- `F` = Feature Release (bundle of enhancements delivered together)
- `H` = Hotfix Patch (single production fix)



Branch Workflow:

1. **Hotfix (`hotfix/*`):**
   - **Purpose:** Fix to Production (`main` branch)
   - **Development:** Developed and initially tested on Dev Machine
   - **PR Target:** Merges to `main` branch
   - **Auto-Deploy:** ☑ Automatically deploys to **Production** (via Render.io `main` branch service)
   - **Use Case:** Critical production fixes that need immediate deployment

2. **Bugfix (`bugfix/*`):**
   - **Purpose:** Fix to Beta (`release/*` branch)
   - **Development:** Developed and initially tested on Dev Machine
   - **PR Target:** Merges to `release/bM.F.0` branch (current release branch)
   - **Auto-Deploy:** ☑ Automatically deploys to **Beta** (via Render.io `release/*` branch service)
   - **Use Case:** Fixes bugs found during beta testing before production release

3. **Feature (`feature/*`):**
   - **Purpose:** Addition to Alpha (`develop` branch)
   - **Development:** Developed and initially tested on Dev Machine
   - **PR Target:** Merges to `develop` branch
   - **Auto-Deploy:** ☑ Automatically deploys to **Alpha** (via Render.io `develop` branch service)
   - **Use Case:** New features for alpha testing before beta release

Deployment Flow:

```
Developer Machine (Local Testing)
    ↓
Create Branch: hotfix/* | bugfix/* | feature/*
```

```
    ↓
Commit & Push → GitHub Actions (CI: test, lint)
    ↓
Create Pull Request
    ↓
Review & Merge PR
    ↓
Auto-Deploy via Render.io:
    hotfix/*  → main      → Production ☑
    bugfix/*  → release/* → Beta ☑
    feature/* → develop   → Alpha ☑
```

Key Points:

- **No manual deployment steps** - All deployments are automatic after PR merge
- **CI runs before merge** - GitHub Actions ensures code quality
- **CD runs after merge** - Render.io automatically deploys based on target branch
- **Environment isolation** - Each branch type deploys to its specific environment
- **S3 prefix alignment** - Each environment uses its corresponding S3 prefix (`/prod/`, `/beta/`, `/alpha/`)

Who:

Primary Audience:

- Development team (all developers working on MicroCODE apps)
- DevOps/Infrastructure team
- Product managers coordinating releases

Affected Teams:

- All teams deploying Regatta-RC, LADDERS, and future MicroCODE apps
- Teams managing beta testing and production releases

What:

Decision: Use **Render.io** as the deployment platform for all MicroCODE web applications, integrated with our existing GitHub Actions CI workflows.

Key Requirements Addressed:

1. ☑ SSH to GitHub repositories (already working)
2. ☑ Work on ISSUEs (HOTFIX, FEATURE, BUGFIX) with existing Git workflow
3. ☑ Commit, Auto Test (GitHub Actions), and generate PR
4. ☑ Review and accept PR, auto-deploy triggers:
    - **HOTFIX** (`hotfix/*` → `main`) → Auto-deploys to **Production**
    - **BUGFIX** (`bugfix/*` → `release/*`) → Auto-deploys to **Beta**
    - **FEATURE** (`feature/*` → `develop`) → Auto-deploys to **Alpha**
    - **RELEASE** (create `release/bM.F.0` branch) → Auto-deploys to **Beta**
    - **PROMOTE** (merge `release/*` → `main`) → Auto-deploys to **Production**

Architecture:

```
GitHub Repository
    ↓
GitHub Actions (CI) ← Existing workflows (lint, test, build)
    ↓ (runs tests, linting)
    ↓
Render.io (CD) ← Auto-deploys on merge
    ↓
Production/Beta/Alpha Services
    ↓
AWS S3 (Storage) ← Uses environment-specific prefixes
Mailgun (Email) ← Handles all app emails
```

Environment Flow:

```
main branch          → Production Service → S3  /prod/ prefix
release/bM.F.0       → Beta Service       → S3  /beta/ prefix
develop branch       → Alpha Service      → S3  /alpha/ prefix (free
tier)
```

When:

Implementation Timeline:

- **Immediate:** Set up Render.io services for beta launch (January 2026)
- **Phase 1:** Configure Production, Beta, and Alpha services (Week 1)
- **Phase 2:** Configure Mailgun (Foundation plan) and test email delivery (Week 1)
- **Phase 3:** Test auto-deployment workflow across all environments (Week 2)
- **Phase 4:** Full rollout for Regatta-RC and LADDERS (Week 3)

Target: Complete setup before beta launch in January 2026.

Service Configuration Summary:

- **Production:** `main` branch → Starter Plus ($15/month) → S3 `/prod/`
- **Beta:** `release/*` branches → Free tier ($0/month, < 24 concurrent users) → S3 `/beta/`
- **Alpha:** `develop` branch → Free tier ($0/month) → S3 `/alpha/`
- **Email:** Mailgun Foundation ($35/month, 50,000 emails) → All environments

Where:

Documentation:

- This document: `.github/docs/DNVs/(DVN-2025-12-19) Best CI-CD Pipeline Decision [ALL].md`
- Related: `.github/docs/DNVs/(DVN-2025-12-19-TJM) Creation of AWS S3 + RENDER for all Apps [ALL].md`
- Git Workflow: `.github/docs/_DEV/DEVELOPER-GIT.md`
- GitHub Actions: `.github/.github/workflows/`

Services:

- Render.io Dashboard: https://dashboard.render.com
- GitHub Actions: https://github.com/MicroCODE-App-Template/.github/actions

Why:

Decision Criteria

Required Workflow Support:

1. **HOTFIX → Production:** Must auto-deploy `main` branch to Production service
2. **BUGFIX → Beta:** Must auto-deploy `release/bM.F.0` branches to Beta service
3. **RELEASE → Beta:** Must auto-deploy `release/bM.F.0` branches to Beta service
4. **FEATURE → Alpha:** Must auto-deploy `develop` branch to Alpha service (free tier)
5. **PROMOTE Beta → Production:** Must auto-deploy when `release/*` merges to `main`

Comparison: Render.io vs AWS ECS

Criteria	Render.io	AWS ECS	Winner
GitHub Integration	☑ Built-in, zero config	❌ Requires GitHub Actions setup	**Render.io**
Auto-Deploy	☑ Automatic on push	❌ Must write workflows (~100 lines YAML)	**Render.io**
Preview Deployments	☑ Auto for PRs/feature branches	❌ Must configure manually	**Render.io**
Setup Time	☑ 15 minutes	❌ 2-4 hours	**Render.io**
Maintenance	☑ Minimal	❌ Ongoing workflow updates	**Render.io**
Cost (Beta)	☑ $15/month	❌ ~$66/month	**Render.io**
AWS Integration	⚠ Via access keys	☑ Native IAM roles	**ECS**
Control	⚠ Limited	☑ Full control	**ECS**
Scalability	☑ Good	☑ Excellent	**Tie**
Cost at Scale	⚠ Higher	☑ Lower	**ECS**

Why Render.io Wins

1. Perfect Workflow Match:

- Render.io's branch-based deployments map exactly to our Git workflow
- `main` → Production, `release/*` → Beta (automatic)
- No custom workflow code needed

2. Zero CI/CD Setup:

- Our existing GitHub Actions handle CI (testing, linting)
- Render.io handles CD (deployment) automatically

- No need to write deployment workflows

3. Lower Initial Cost:

- Render.io: $15/month (Production $15 + Beta $0 + Alpha $0)
- ECS: ~$66/month (Fargate $40 + ALB $16 + ECR $1 + CloudWatch $2 + other $7)
- Mailgun: $35/month (Foundation plan - 50,000 emails/month)
- AWS S3/KMS: ~$7-8/month
- **Total: $57-58/month** (Render.io + Mailgun + AWS) vs ~$101/month (ECS + Mailgun + AWS)
- **Savings: $43/month** (43% cheaper for beta phase)

4. Faster Time to Market:

- Setup: 15 minutes vs 2-4 hours
- Less DevOps overhead
- Focus on features, not infrastructure

5. Better Developer Experience:

- Preview deployments for feature branches (automatic)
- Simple environment variable management
- Clear deployment status in dashboard

Why Not ECS (For Now)

ECS is Better For:

- Advanced AWS features (VPC, custom networking)
- Teams already heavily invested in AWS tooling
- Organizations with dedicated DevOps capacity
- Applications requiring fine-grained infrastructure control
- Cost optimization at very large scale (>$200/month)

Our Situation:

- Small team, need to move fast
- Beta launch deadline (January 2026)
- Limited DevOps capacity
- Cost-sensitive for beta phase
- → **Render.io is the better fit**

How:

Implementation Plan

Phase 1: Render.io Service Setup

1. Create Production Service:

```
Service Name: regattarc-production
Branch: main
```

```
Auto-deploy: ☑ Enabled
Environment: production
Plan: Starter Plus ($15/month)
Environment Variables:
  - NODE_ENV=production
  - S3_ENVIRONMENT=prod
  - AWS_ACCESS_KEY_ID=... (from IAM user)
  - AWS_SECRET_ACCESS_KEY=... (from IAM user)
  - AWS_REGION=us-east-2
  - [Other production config]
```

2. Create Beta Service:

```
Service Name: regattarc-beta
Branch: release/* (pattern matching)
Auto-deploy: ☑ Enabled
Environment: beta
Plan: Free tier ($0/month) - Low concurrent usage expected (< 24 users)
Environment Variables:
  - NODE_ENV=beta
  - S3_ENVIRONMENT=beta
  - AWS_ACCESS_KEY_ID=... (from IAM user)
  - AWS_SECRET_ACCESS_KEY=... (from IAM user)
  - AWS_REGION=us-east-2
  - [Other beta config]
```

Note: Beta uses Render.io's free tier due to low expected concurrent usage (< 24 users). Service sleeps after 15 minutes of inactivity but auto-wakes on first request. If beta usage increases or cold starts become problematic, upgrade to Starter Plus ($15/month).

3. Alpha Service (Required):

```
Service Name: regattarc-alpha
Branch: develop
Auto-deploy: ☑ Enabled
Environment: alpha
Plan: Free tier (for alpha testing)
Environment Variables:
  - NODE_ENV=alpha
  - S3_ENVIRONMENT=alpha
  - AWS_ACCESS_KEY_ID=... (from IAM user)
  - AWS_SECRET_ACCESS_KEY=... (from IAM user)
  - AWS_REGION=us-east-2
  - [Other alpha config]
```

Note: Alpha service uses Render.io's free tier. Services sleep after 15 minutes of inactivity but auto-wake on first request.

How Auto-Deployment Works with Branch Types:

- **Hotfix (`hotfix/*`):** Developer creates `hotfix/fix-name` branch locally, commits, pushes, creates PR targeting `main`. When PR is merged to `main`, Render.io automatically detects the change and deploys to Production ☑
- **Bugfix (`bugfix/*`):** Developer creates `bugfix/fix-name` branch locally, commits, pushes, creates PR targeting `release/bM.F.0`. When PR is merged to `release/*`, Render.io automatically detects the

change and deploys to Beta ☑

- **Feature (`feature/*`):** Developer creates `feature/feature-name` branch locally, commits, pushes, creates PR targeting `develop`. When PR is merged to `develop`, Render.io automatically detects the change and deploys to Alpha ☑

Key Point: Render.io watches the target branches (`main`, `release/*`, `develop`), not the source branches (`hotfix/*`, `bugfix/*`, `feature/*`). When a PR merges to a watched branch, deployment is automatic.

Phase 2: Workflow Integration

Existing GitHub Actions (No Changes Needed):

- ☑ `.github/.github/workflows/ci.yml` - Runs lint, test, build
- ☑ `.github/.github/workflows/release.yml` - Creates releases and tags
- ☑ Branch validation workflows

Render.io Auto-Deployment:

- Automatically triggers on push to configured branches
- No additional GitHub Actions needed
- Works seamlessly with existing CI workflows

Phase 3: Workflow Execution Examples

HOTFIX → Production:

```
# 1. Developer creates hotfix branch locally (Dev Machine)
git checkout main
git checkout -b hotfix/0067--connection-retry
# ... fix code, test locally ...
git commit -m "fix: connection retry logic"
git push origin hotfix/0067--connection-retry

# 2. Create PR targeting main branch
# → GitHub Actions runs CI (test, lint) ☑
# → Review PR
# → Merge PR to main ☑

# 3. Render.io automatically detects main branch change
# → Auto-deploys to Production ☑
# → Uses S3 /prod/ prefix
```

BUGFIX → Beta:

```
# 1. Developer creates bugfix branch locally (Dev Machine)
git checkout release/b2.0.0
git checkout -b bugfix/0068--ui-fix
# ... fix code, test locally ...
git commit -m "fix: UI layout issue"
git push origin bugfix/0068--ui-fix

# 2. Create PR targeting release/b2.0.0 branch
# → GitHub Actions runs CI (test, lint) ☑
# → Review PR
```

```
# → Merge PR to release/b2.0.0 ☑

# 3. Render.io automatically detects release/* branch change
# → Auto-deploys to Beta ☑
# → Uses S3 /beta/ prefix
```

FEATURE → Alpha:

```
# 1. Developer creates feature branch locally (Dev Machine)
git checkout develop
git checkout -b feature/0100--new-feature
# ... add feature, test locally ...
git commit -m "feat: add new feature"
git push origin feature/0100--new-feature

# 2. Create PR targeting develop branch
# → GitHub Actions runs CI (test, lint) ☑
# → Review PR
# → Merge PR to develop ☑

# 3. Render.io automatically detects develop branch change
# → Auto-deploys to Alpha ☑
# → Uses S3 /alpha/ prefix
```

RELEASE → Beta:

```
# 1. Create release branch from main (or develop)
git checkout main
git checkout -b release/b2.1.0
git cherry-pick feature/0100--new-feature
# ... test release candidate ...
git push origin release/b2.1.0

# 2. Render.io automatically detects release/* branch creation
# → Auto-deploys to Beta ☑
# → Uses S3 /beta/ prefix
```

PROMOTE Beta → Production:

```
git checkout main
git merge release/b2.1.0
git tag v2.1.0
git push --tags
# Render.io auto-deploys main to Production ☑
```

Git Branch to Render.io Environment Mapping

Complete Branch Strategy:

Git Branch	PR Merge Target	Code Environment	Render.io Service	S3 Prefix	Render.io Plan	Auto-Deploy

Git Branch	PR Merge Target	Code Environment	Render.io Service	S3 Prefix	Render.io Plan	Auto-Deploy
`main`	N/A (target)	Production	`{app}-production`	`/prod/` (blank)	Starter Plus ($15/month)	☑ Yes
`release/bM.F.0`	N/A (target)	Beta	`{app}-beta`	`/beta/`	Free tier ($0/month)	☑ Yes
`develop`	N/A (target)	Alpha	`{app}-alpha`	`/alpha/`	Free tier ($0/month)	☑ Yes
`hotfix/*`	→ `main`	Production	`{app}-production`	`/prod/` (blank)	Starter Plus ($15/month)	☑ Yes*
`bugfix/*`	→ `release/*`	Beta	`{app}-beta`	`/beta/`	Free tier ($0/month)	☑ Yes*
`feature/*`	→ `develop`	Alpha	`{app}-alpha`	`/alpha/`	Free tier ($0/month)	☑ Yes*

Notes:

- **Production (`main`):** Live production environment, always running (Starter Plus - $15/month)
  - **Hotfix Flow:** `hotfix/*` branches are developed locally, PR merges to `main`, auto-deploys to Production ☑
- **Beta (`release/*`):** Beta testing environment for release candidates (Free tier - $0/month, sleeps after 15 min inactivity)
  - **Bugfix Flow:** `bugfix/*` branches are developed locally, PR merges to `release/*`, auto-deploys to Beta ☑
- **Alpha (`develop`):** Alpha testing environment, uses free tier (sleeps after 15 min inactivity)
  - **Feature Flow:** `feature/*` branches are developed locally, PR merges to `develop`, auto-deploys to Alpha ☑
- **Auto-Deploy Note:** Branches marked with `*` auto-deploy when their PR is merged to their target branch (which triggers the Render.io service for that branch)

Beta Free Tier Rationale:

- Expected concurrent usage: < 24 users
- Low traffic volume justifies free tier
- Cold starts acceptable for beta testing (30-60 second delay after 15 min inactivity)
- Can upgrade to Starter Plus ($15/month) if usage increases or cold starts become problematic

S3 Prefix Strategy:

- Production uses blank prefix (root level): `regattarc-private-assets/users/123/`
- Beta uses `/beta/` prefix: `regattarc-private-assets/beta/users/123/`
- Alpha uses `/alpha/` prefix: `regattarc-private-assets/alpha/users/123/`

Cost Breakdown

Complete Infrastructure Costs (Beta Phase):

```
Render.io Services:
  Production Service:  $15/month (Starter Plus - 1 GB RAM, 1 CPU)
  Beta Service:        $15/month (Starter Plus - 1 GB RAM, 1 CPU)
  Alpha Service:       $0/month (Free tier - sleeps after 15 min)
  ──────────────────────────────────────────────────────────────
  Render.io Total:     $30/month

AWS Services:
  S3 Storage:          $4-5/month
  CloudFront:          $0/month (free tier)
  KMS Encryption:      $3/month
  ──────────────────────────────────────────────────────────────
  AWS Total:           $7-8/month

Email Service:
  Mailgun Foundation:  $35/month (50,000 emails/month included)
  ──────────────────────────────────────────────────────────────
  Email Total:         $35/month

──────────────────────────────────────────────────────────────
TOTAL INFRASTRUCTURE:  $57-58/month
```

Cost Savings: Using free tier for Beta saves $15/month, bringing total infrastructure cost down to **$57-58/month** (43% under $100/month target).

Well under $100/month target ☑

Mailgun Details:

- **Plan:** Foundation ($35/month)
- **Included:** 50,000 emails per month
- **Overage:** $1.30 per 1,000 emails
- **Usage:** All app-generated emails (notifications, password resets, reports, etc.)
- **Environments:** Used by Production, Beta, and Alpha (same account, different sender addresses)

Beta & Alpha Environment Ramifications (Free Tier)

Free Tier Limitations (Apply to Both Beta & Alpha):

1. **Cold Starts:**
    - Service sleeps after 15 minutes of inactivity
    - First request after sleep: 30-60 second delay
    - **Impact:** Not suitable for production workloads or high-traffic beta testing
    - **Beta Mitigation:** Acceptable for low-traffic beta testing (< 24 concurrent users)
    - **Alpha Mitigation:** Acceptable for alpha testing/integration
2. **Performance:**
    - Limited CPU/RAM compared to paid plans
    - May timeout on long-running operations
    - **Impact:** Database migrations may fail
    - **Mitigation:** Run migrations in Production environment (or upgrade Beta to paid tier if needed)

3. **Reliability:**
    - Free tier has lower SLA than paid plans
    - May experience more downtime
    - **Beta Impact:** Beta testing may be interrupted (acceptable for low-traffic beta)
    - **Alpha Impact:** Alpha testing may be interrupted (acceptable for development/integration)
4. **S3 Prefix Usage:**
    - Beta operations use `/beta/` prefix
    - Alpha operations use `/alpha/` prefix
    - Keeps data isolated from Production
    - **Benefit:** Safe to test without affecting production
    - **Consideration:** Need to clean up `/beta/` and `/alpha/` data periodically

Best Practices for Beta (Free Tier):

- Use for low-traffic beta testing (< 24 concurrent users)
- Set expectations with beta testers about cold starts (30-60 second delay after 15 min inactivity)
- Monitor usage - upgrade to Starter Plus ($15/month) if:
    - Concurrent users regularly exceed 24
    - Cold starts become problematic for testers
    - Performance issues arise
    - Need better reliability for beta testing
- Don't use for performance testing (use Production for that)
- Don't use for critical migrations (use Production)
- Clean up `/beta/` S3 data monthly

Best Practices for Alpha (Free Tier):

- Use for feature integration testing
- Use for early alpha user testing
- Don't use for performance testing
- Don't use for critical migrations (use Production)
- Monitor cold start times and set expectations
- Clean up `/alpha/` S3 data monthly

Migration Path (Future Consideration)

When to Consider ECS:

- Costs exceed $200/month consistently
- Need advanced AWS features (VPC, custom networking)
- Require fine-grained infrastructure control
- Have dedicated DevOps capacity
- Scale requires cost optimization
- Beta or Alpha environments need better performance (upgrade to paid tier)

Migration Strategy:

- Keep architecture portable (Express/Node.js)
- Use environment variables for all config
- Abstract deployment details in code
- Can migrate to ECS without code changes
- S3 prefix strategy works with any platform

Notes

Key Decisions Made

1. **Render.io Selected** for all MicroCODE web applications
2. **GitHub Actions Retained** for CI (testing, linting, building)
3. **Render.io Handles CD** (deployment) automatically
4. **Branch-Based Deployments** match Git workflow exactly:
    - `main` → Production (Starter Plus - $15/month)
    - `release/*` → Beta (Free tier - $0/month, < 24 concurrent users expected)
    - `develop` → Alpha (Free tier - $0/month)
5. **Mailgun Selected** for all app email (Foundation plan - $35/month, 50,000 emails)
6. **S3 Prefix Strategy** aligns with environments (`/prod/`, `/beta/`, `/alpha/`)
7. **Cost Target Met** ($57-58/month vs $100/month target - 43% under budget)

Advantages of This Approach

1. **Zero CI/CD Setup:** Render.io handles deployment automatically
2. **Perfect Workflow Match:** Branch deployments align with Git workflow
3. **Lower Cost:** 43% cheaper than ECS for beta phase ($57-58 vs ~$101/month)
4. **Beta & Alpha Environments:** Free tier for both `release/*` and `develop` branches saves $15/month
5. **Beta Free Tier:** Acceptable for low-traffic beta testing (< 24 concurrent users)
6. **Mailgun Integration:** Cost-effective email service ($35/month for 50,000 emails)
7. **Faster Setup:** 15 minutes vs 2-4 hours
8. **Better DX:** Preview deployments, clear status, simple config
9. **Less Maintenance:** No workflow YAML to maintain
10. **Three-Tier Environment:** Production (paid), Beta (free), and Alpha (free) for comprehensive testing

Considerations

1. **IAM Access Keys:** Render.io requires IAM user access keys (not roles)
    - Mitigation: Rotate keys every 90 days, use separate IAM users per service
    - Documented in AWS S3 + RENDER plan
2. **Vendor Lock-in:** Render.io is proprietary platform
    - Mitigation: Keep architecture portable, use environment variables
    - Can migrate to ECS/other platforms if needed
3. **Cost Scaling:** Render.io costs scale with plan tiers
    - Monitor costs monthly
    - Consider ECS migration if costs exceed $200/month

4. **Beta & Alpha Environment Limitations (Free Tier):**
    - Cold starts (30-60 second delay after 15 min inactivity)
    - Limited performance (may timeout on long operations)
    - Lower reliability (free tier SLA)
    - **Beta Mitigation:** Acceptable for low-traffic beta testing (< 24 concurrent users)
    - **Alpha Mitigation:** Use for integration testing only, not performance testing
    - **Consideration:** Upgrade to Starter Plus ($15/month) if:
        - Beta concurrent users exceed 24 regularly
        - Cold starts become problematic for beta testers
        - Performance issues arise
        - Need better reliability for beta testing

Next Steps

1. **Create Render.io Account** (if not already done)
2. **Connect GitHub Repository** to Render.io
3. **Create Production Service** (main branch → Starter Plus - $15/month)
4. **Create Beta Service** (release/* pattern → Free tier - $0/month, < 24 concurrent users expected)
5. **Create Alpha Service** (develop branch → Free tier - $0/month)
6. **Configure Environment Variables** per service (including S3_ENVIRONMENT)
7. **Configure Mailgun** (Foundation plan - $35/month for 50,000 emails)
8. **Test Auto-Deploy** with test commits to each branch
9. **Document Service URLs** and access patterns
10. **Train Team** on Render.io deployment workflow and environment mapping

Related Documents

- **AWS S3 + RENDER Plan:** `.github/docs/DNVs/(DVN-2025-12-19-TJM) Creation of AWS S3 + RENDER for all Apps [ALL].md`
- **Git Workflow:** `.github/docs/_DEV/DEVELOPER-GIT.md`
- **GitHub Actions:** `.github/.github/workflows/`

Environment-Specific Considerations

Alpha Environment (develop branch):

- Uses Render.io free tier (sleeps after 15 minutes inactivity)
- First request after sleep takes 30-60 seconds (cold start)
- Suitable for alpha testing and feature integration
- No cost, but limited performance
- S3 prefix: `/alpha/` for all storage operations

Beta Environment (release/bM.F.0 branches):

- Uses Free tier ($0/month) - Low concurrent usage expected (< 24 users)
- Sleeps after 15 minutes of inactivity (cold start: 30-60 seconds)
- Suitable for beta testing with low traffic volume
- Can upgrade to Starter Plus ($15/month) if usage increases or cold starts become problematic

- S3 prefix: `/beta/` for all storage operations

Production Environment (main branch):

- Uses Starter Plus plan ($15/month)
- Always running, highest priority
- S3 prefix: `/prod/` (blank, root level) for all storage operations

Mailgun Email Service:

- Foundation plan: $35/month
- Includes: 50,000 emails/month
- Overage: $1.30 per 1,000 emails
- Used by: All environments (Production, Beta, Alpha)
- Configuration: Same Mailgun account, different sender addresses per environment

Open Questions

- Should we create separate Render.io services per app (regattarc, ladders) or use one service with multiple environments?
- **Recommendation:** Separate services per app for isolation and independent scaling
- **Example:** `regattarc-production`, `regattarc-beta`, `regattarc-alpha` vs `ladders-production`, `ladders-beta`, `ladders-alpha`
- How do we handle database migrations in Render.io?
- **Recommendation:** Run migrations as part of deployment script or separate migration service
- **Consideration:** Alpha environment (free tier) may timeout on long migrations - use Beta/Production for migrations
- Should Alpha environment use same database as Beta/Production?
- **Recommendation:** Separate databases per environment for safety, or use database with environment-specific schemas/prefixes

---

Status: ☑ **APPROVED** - Render.io selected as deployment platform for all MicroCODE web applications.