Agent Foundry MVP - Implementation Plan

Agent Foundry MVP - Implementation Plan

Scope

Product: Agent Foundry (Ravenhelm Series A)

Domain: foundry.ravenhelm.dev

Infrastructure: Single AWS account, single EC2 instance

Timeline: 2-3 weeks

Stack

```
Single EC2 (t3.small, ~$15/month)
└─ Docker Compose
   ├─ foundry-frontend (Next.js)
   ├─ foundry-backend (FastAPI)
   ├─ foundry-compiler (FastAPI)
   └─ livekit-server

Storage: SQLite + filesystem
SSL: Let's Encrypt
CI/CD: GitLab → SSH deploy
```

Core Capabilities

Platform (Runtime)

- Agent hosting from YAML manifests
- Chat UI with WebSocket

- Voice integration (LiveKit)
- Agent registry (hot-reload)
- Admin API

Compiler (Build Tool)

- DIS 1.6.0 parser
- Agent YAML generator
- Mock API generator
- One-click deployment

User Flow

```
Upload DIS dossier
  ↓
Compiler generates agent YAML
  ↓
Platform hot-loads agent
  ↓
User gets demo URL with voice
```

Week 1: Platform + LiveKit

Days 1-2: LiveKit Integration

- [ ] Install LiveKit dependencies (backend + frontend)
- [ ] Create voice session endpoint
- [ ] Add voice UI component
- [ ] Test voice locally

Days 3-4: Docker Compose

- [ ] Create docker-compose.yml (all services)

- [ ] Create Dockerfiles (frontend, backend, compiler)
- [ ] Test full stack locally
- [ ] Create .env.dev template

Day 5: AWS Deploy

- [ ] Create Terraform config (EC2, security group, elastic IP)
- [ ] Deploy infrastructure
- [ ] Configure DNS: foundry.ravenhelm.dev → EC2 IP
- [ ] Install Docker + Docker Compose on EC2
- [ ] Deploy services
- [ ] Setup Let's Encrypt SSL
- [ ] Verify: https://foundry.ravenhelm.dev working

Week 2: DIS Compiler

Days 1-2: Compiler Core

- [ ] Extract agent YAML schema from existing pm-agent.yaml
- [ ] Build DIS parser (DIS 1.6.0 → dict)
- [ ] Build agent YAML generator (Jinja2 template)
- [ ] Create compiler API endpoints
- [ ] Test with sample DIS dossier

Days 3-4: Integration

- [ ] Connect compiler to platform agent registry
- [ ] Test end-to-end: DIS upload → agent live
- [ ] Add basic error handling
- [ ] Create demo dossier examples

Day 5: Polish

- [ ] Add upload UI to frontend
- [ ] Create landing page
- [ ] Basic documentation
- [ ] Test full workflow

Week 3: CI/CD + Production Ready

Days 1-2: GitLab Pipeline

- [ ] Create .gitlab-ci.yml
- [ ] Setup SSH deploy keys
- [ ] Test deploy pipeline
- [ ] Document deployment process

Days 3-4: Monitoring & Security

- [ ] Basic logging to SQLite
- [ ] Nginx reverse proxy config
- [ ] Security group hardening
- [ ] Backup strategy (snapshot SQLite)

Day 5: Launch Prep

- [ ] End-to-end testing
- [ ] Performance check
- [ ] Create user documentation
- [ ] Announce availability

Success Criteria

MVP Complete When:

- [ ] foundry.ravenhelm.dev accessible via HTTPS
- [ ] Can upload DIS dossier via UI
- [ ] Compiler generates valid agent YAML
- [ ] Agent appears in chat UI < 60 seconds
- [ ] Voice conversation works with agent
- [ ] GitLab pipeline deploys updates
- [ ] Demo URL shareable with others

Critical Path Items

Before Starting:

- [ ] Review existing Engineering Department codebase
- [ ] Extract pm-agent.yaml schema (compiler template)
- [ ] AWS account access confirmed
- [ ] foundry.ravenhelm.dev DNS access confirmed

Blockers to Resolve:

- Agent YAML schema format (need actual file)
- GitLab repo structure decision
- AWS credentials setup

Post-MVP (Not in Scope)

- Production environment (foundry.ravenhelm.ai)
- Multi-environment deployments
- Auto-scaling
- Advanced monitoring
- User authentication
- Database migration from SQLite

- Additional Series products

Cost

Monthly Operating Cost: ~$15-20

- EC2 t3.small: ~$15
- Elastic IP: ~$3
- Storage/bandwidth: ~$2
- Domains: $0 (owned)

One-time Setup: ~$0 (using free tier eligible resources)

Next Action

Share the agent YAML schema:

```
cat ~/Development/Projects/Engineering\ Department/engineeringdepartment/agents/pm-agent.agent.yaml
```

This locks the compiler output format and unblocks Week 2.