Agent Foundry - MVP Architecture

Naming Clarity ✅

Agent Foundry = Complete product ("Heroku for AI Agents")

```
Agent Foundry
├── Platform (Runtime)
│   └── Evolved from Engineering Department POC
│       ├── Agent hosting infrastructure
│       ├── Admin API
│       ├── Chat UI
│       └── Voice (LiveKit)
│
└── Compiler (Build Tool)
    ├── DIS Parser
    ├── Agent Generator
    └── Mock API Generator
```

One-line pitch: Upload DIS dossier → Get hosted voice-enabled agent demo

---

Agent Foundry MVP Stack

Core Architecture

```
Agent Foundry Platform
│
├─ Compiler Service
│  ├─ DIS 1.6.0 Parser
│  ├─ Agent YAML Generator
│  └─ Mock API Generator
│
```

```
├─ Runtime Platform
│  ├─ Frontend (Next.js chat UI)
│  ├─ Backend (FastAPI MCP + admin)
│  ├─ Agent Registry (YAML hot-reload)
│  ├─ Agent Runtime (LangGraph)
│  └─ Voice Service (LiveKit)
│
└─ Storage
   ├─ SQLite (sessions, metrics, logs)
   └─ Filesystem (agent YAML, compiled artifacts)
```

Single EC2 Deployment

```
EC2 t3.small (~$25/month)
│
└─ Docker Compose
   ├─ foundry-frontend:3000
   ├─ foundry-backend:8000
   ├─ foundry-compiler:8001
   ├─ livekit:7880
   └─ redis:6379 (optional)
```

Rebranded File Structure

```
agent-foundry/
├── platform/              # Runtime (evolved from Engineering Dept)
│   ├── frontend/
│   │   ├── app/
│   │   │   ├── components/
│   │   │   │   ├── Chat.tsx
│   │   │   │   ├── VoiceChat.tsx
│   │   │   │   └── AgentSelector.tsx
│   │   │   └── page.tsx
│   │   └── package.json
│   │
│   ├── backend/
│   │   ├── api/
│   │   │   ├── chat.py
│   │   │   ├── admin.py        # Agent management
│   │   │   └── voice.py        # LiveKit integration
│   │   ├── agent_runtime/
│   │   │   ├── registry.py     # YAML hot-reload
│   │   │   └── langgraph_runner.py
│   │   └── main.py
│   │
│   └── agents/                 # Agent manifests
│       ├── pm-agent.yaml       # From Engineering Dept POC
│       └── {compiled-agents}.yaml
│
├── compiler/                   # DIS build tool
│   ├── src/
│   │   ├── parser/
│   │   │   └── dis_parser.py
│   │   ├── generators/
│   │   │   ├── agent_yaml.py
│   │   │   ├── mock_api.py
│   │   │   └── mcp_tools.py
│   │   └── api/
│   │       └── compiler_api.py  # Port 8001
│   └── templates/
│       ├── agent.yaml.j2
│       └── fastapi_mock.py.j2
│
├── infrastructure/
│   ├── terraform/
│   │   └── main.tf             # Single EC2
│   ├── docker-compose.yml      # All services
│   └── nginx.conf
│
├── data/                       # SQLite + logs
```

```
│   ├── foundry.db
│   └── logs/
│
└── .gitlab-ci.yml              # CI/CD
```

User Journey (Agent Foundry)

1. Upload DIS Dossier

```
POST https://foundry.ravenhelm.ai/api/compiler/upload
Content-Type: application/json

{
  "dossier": { ... DIS 1.6.0 JSON ... }
}

Response:
{
  "job_id": "compile-abc123",
  "status": "compiling"
}
```

2. Compiler Generates Artifacts

```
Compiler Service (30 seconds):
├─ Parse DIS dossier
├─ Generate agent YAML
├─ Generate mock APIs
├─ Save to /agents/banking-demo.yaml
└─ Notify platform: agent ready
```

3. Platform Loads Agent

```
Runtime Platform:
├─ Hot-reload agent from YAML
├─ Register tools (mock APIs)
├─ Create LiveKit room
└─ Return demo URL
```

4. User Accesses Demo

```
https://foundry.ravenhelm.ai/demo/banking-demo
├─ Chat UI loads
├─ Agent selector shows "Banking Support"
├─ User clicks "Enable Voice"
└─ Voice-enabled conversation with agent
```

Total time: < 1 minute from upload to live demo

Revised MVP Timeline

Week 1: Platform + LiveKit (12-15 hours)

Goal: Agent Foundry Platform running with voice

Day 1-2: LiveKit Integration (6 hours)

```
# platform/backend/api/voice.py
from livekit import api

@app.post("/api/voice/session/{session_id}")
async def create_voice_session(session_id: str):
    """Create LiveKit room for agent demo"""
    token = api.AccessToken(LIVEKIT_KEY, LIVEKIT_SECRET)
    token.with_identity(f"demo-user-{session_id}")
    token.with_grants(api.VideoGrants(
        room_join=True,
        room=f"foundry-{session_id}",
    ))

    return {
        "token": token.to_jwt(),
        "url": LIVEKIT_URL,
        "room": f"foundry-{session_id}"
    }
```

```
// platform/frontend/app/components/VoiceChat.tsx
export function VoiceChat({ sessionId, agentId }) {
  return (
    <LiveKitRoom
      token={token}
      serverUrl={LIVEKIT_URL}
      audio={true}
    >
      <ControlBar />
      <p>Speaking with {agentId}...</p>
    </LiveKitRoom>
  );
}
```

Day 3-4: Docker Compose (4 hours)

```
# docker-compose.yml - Agent Foundry Stack
version: '3.8'

services:
  foundry-frontend:
    build: ./platform/frontend
    ports:
      - "3000:3000"
    environment:
      - NEXT_PUBLIC_API_URL=http://foundry-backend:8000
      - NEXT_PUBLIC_COMPILER_URL=http://foundry-compiler:8001
      - NEXT_PUBLIC_LIVEKIT_URL=ws://livekit:7880

  foundry-backend:
    build: ./platform/backend
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=sqlite:////data/foundry.db
      - LIVEKIT_URL=ws://livekit:7880
    volumes:
      - ./data:/data
      - ./platform/agents:/agents

  foundry-compiler:
    build: ./compiler
    ports:
      - "8001:8001"
    volumes:
      - ./platform/agents:/output/agents
      - ./data:/data

  livekit:
    image: livekit/livekit-server:latest
    ports:
      - "7880:7880"
    volumes:
      - ./infrastructure/livekit.yaml:/etc/livekit.yaml
```

Day 5: AWS Deploy (3 hours)

```
# infrastructure/terraform/main.tf
resource "aws_instance" "agent_foundry" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t3.small"
  tags = {
    Name = "agent-foundry-platform"
  }

  user_data = <<-EOF
    #!/bin/bash
    yum install -y docker
    systemctl start docker

    curl -L https://github.com/docker/compose/releases/latest/download/dock
er-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
    chmod +x /usr/local/bin/docker-compose

    cd /opt/agent-foundry
    docker-compose up -d
  EOF
}
```

Deliverable: Agent Foundry Platform live with voice

Week 2: DIS Compiler (8-10 hours)

Goal: DIS dossier → deployed agent

Day 1-2: Compiler Core (6 hours)

```
# compiler/src/api/compiler_api.py
from fastapi import FastAPI, UploadFile
from .parser import parse_dis_dossier
from .generators import generate_agent_yaml

app = FastAPI(title="Agent Foundry Compiler")

@app.post("/compile")
async def compile_dossier(file: UploadFile):
    """
    Compile DIS dossier into Agent Foundry manifest
    """
    # Parse DIS
    content = await file.read()
    dossier = json.loads(content)
    parsed = parse_dis_dossier(dossier)

    # Generate agent YAML
    for agent in parsed["agents"]:
        yaml_content = generate_agent_yaml(agent)
        agent_id = agent["id"]

        # Save to platform agents directory
        with open(f"/output/agents/{agent_id}.yaml", "w") as f:
            f.write(yaml_content)

    return {
        "status": "compiled",
        "agents": [a["id"] for a in parsed["agents"]],
        "demo_url": f"https://foundry.ravenhelm.ai/demo/{parsed['agents'][0]['i
d']}"
    }
```

```
# compiler/src/generators/agent_yaml.py
from jinja2 import Template

AGENT_TEMPLATE = Template("""
name: {{ name }}
description: {{ description }}
system_prompt: |
{{ system_prompt | indent(2) }}
tools:
{% for tool in tools %}
  - {{ tool }}
{% endfor %}
workflow:
  type: conversation
  states:
    - listen
    - process
    - respond
""")

def generate_agent_yaml(agent: dict) → str:
    """Convert DIS agent to Agent Foundry YAML"""
    return AGENT_TEMPLATE.render(
        name=agent["name"],
        description=agent.get("description", ""),
        system_prompt=agent.get("instructions", ""),
        tools=extract_tools(agent)
    )
```

Day 3: Integration Testing (2 hours)

```
# End-to-end test
curl -X POST http://localhost:8001/compile \
  -F "file=@sample-dossier.json"

# Response:
{
  "status": "compiled",
  "agents": ["banking-support"],
  "demo_url": "https://foundry.ravenhelm.ai/demo/banking-support"
}

# Visit URL → Agent loaded, voice enabled
```

Deliverable: DIS → Live agent demo in < 1 minute

Week 3: Polish + CI/CD (6-8 hours)

GitLab Pipeline

```
# .gitlab-ci.yml - Agent Foundry Deployment
stages:
  - build
  - deploy

build:
  stage: build
  script:
    - docker-compose build
  only:
    - main

deploy:
  stage: deploy
  script:
    - ssh ec2-user@foundry.ravenhelm.ai "cd /opt/agent-foundry && git pull &&
docker-compose up -d --build"
  only:
    - main
  when: manual
```

Landing Page

```
// platform/frontend/app/page.tsx - Marketing site
export default function Home() {
  return (
    <div className="hero">
      <h1>Agent Foundry</h1>
      <p>Heroku for AI Agents</p>
      <p>Upload DIS dossier → Get voice-enabled agent demo</p>
      <UploadDossier />

      <div className="demos">
        <h2>Live Demos</h2>
        <AgentGallery />
      </div>
    </div>
  );
}
```

Deliverable: Production-ready Agent Foundry platform

Agent Foundry Brand Assets

Logo Concept

```
┌─────────────────────────────────────┐
│   🏭                    │
│  AGENT                    │
│  FOUNDRY                   │
│                    │
│  Heroku for AI Agents       │
└─────────────────────────────────────┘
```

Color Palette (Raven-themed)

```
Primary:   #1a1a2e (Midnight Blue)
Secondary: #16213e (Deep Navy)
Accent:    #0f3460 (Dark Cyan)
Highlight: #533483 (Purple)
Text:      #e94560 (Red accent)
```

URLs

```
Platform: https://foundry.ravenhelm.ai
Compiler: https://foundry.ravenhelm.ai/api/compiler
Docs:     https://docs.foundry.ravenhelm.ai
```

Critical Next Steps (This Week)

1. Rebrand Engineering Department → Agent Foundry

```
cd ~/Development/Projects/Engineering\ Department
mv engineeringdepartment agent-foundry
cd agent-foundry

# Update package.json, configs, etc.
find . -type f -exec sed -i '' 's/engineering-department/agent-foundry/g' {} +
```

2. Extract Agent YAML Schema

```
# Show me the current schema
cat agents/pm-agent.agent.yaml

# This becomes the template for DIS compiler
```

3. Add LiveKit

```
# Backend
pip install livekit-server-sdk-python

# Frontend
npm install @livekit/components-react

# Generate code from templates above
```

4. Create Docker Compose

```
# Copy docker-compose.yml from above
# Adjust paths to match your structure
docker-compose up -d

# Test: http://localhost:3000
```

Quick Win Checklist

Today (2 hours):

- [ ] Rebrand project → Agent Foundry
- [ ] Extract agent YAML schema
- [ ] Install LiveKit dependencies

Tomorrow (4 hours):

- [ ] Add LiveKit backend endpoint
- [ ] Add LiveKit frontend component
- [ ] Test voice locally

Day 3 (3 hours):

- [ ] Create docker-compose.yml
- [ ] Test full stack locally

Day 4-5 (4 hours):

- [ ] AWS EC2 + Terraform
- [ ] Deploy to AWS
- [ ] Point domain

End of Week 1: **Agent Foundry Platform live with voice**

What's the agent YAML schema? Let's lock that down first.

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

Agent Foundry MVP - Implementation Plan