ISR Mission Planning

Deployment & Setup Guide

Generated: 2025-12-05 11:32

Project Directory Structure

CLEAN STRUCTURE

```
isr_web/
├── server/
│   ├── agents/
│   │   ├── isr_agent.py   # Claude agent
│   │   └── graph.py       # GPT agent
│   ├── solver/
│   │   ├── orienteering_solver.py
│   │   ├── solver_bridge.py
│   │   ├── trajectory_planner.py
│   │   ├── sam_distance_matrix.py
│   │   ├── target_allocator.py
│   │   └── post_optimizer.py
│   └── main.py            # FastAPI app
├── webapp/
│   ├── index.html
│   ├── isr.js
│   └── isr.css
├── docs/                  # PDF generators
├── requirements.txt
├── Dockerfile
├── docker-compose.yml
└── run_planner.sh
```

SIBLING PACKAGES

```
isr_projects/          # Parent dir
├── isr_web/           # This app
├── path_planning_core/
│   ├── boundary_navigation.py
│   └── sam_wrapping.py
├── isr_editor/
│   ├── solver/
│   │   └── orienteering_interface.py
│   └── path_planning/
│       └── sam_navigator.py
├── isr_benchmark/
└── isr_agent/
```

```
NOTE: Docker build context must
be the parent isr_projects/
directory to include siblings.
```

FILES MOVED TO _deprecated/ (Safe to Delete)

```
• server/main_v2.py, ui.py        - Legacy versions
• webapp/index_v2.html, isr_v2.js - Old UI files
• server/continuationChatGPT-1    - Debug artifact
• orienteering_with_matrix.py     - Duplicate solver
• src/                            - Empty directories
• webapp/editor/                  - Old Tkinter app
```

Requirements & Environment Setup

requirements.txt

```
# Web Framework
fastapi>=0.122.0
uvicorn>=0.38.0
pydantic>=2.12.0
python-dotenv>=1.2.0

# LangGraph/LangChain
langgraph>=0.6.11
langchain>=0.3.27
langchain-anthropic>=0.3.22
langchain-core>=0.3.80

# OpenAI (GPT agent)
openai>=2.8.1

# Scientific
numpy>=2.0.0
matplotlib>=3.9.0

# Utilities
requests>=2.32.0
```

ENVIRONMENT VARIABLES

```
Required:
────────────────────────────
ANTHROPIC_API_KEY=sk-ant-...
  For Claude-based ISR agent

Optional:
────────────────────────────
OPENAI_API_KEY=sk-...
  For GPT-based agent

PYTHONPATH=/path/to/isr_projects
  Required for sibling imports

Create .env file:
────────────────────────────
echo "ANTHROPIC_API_KEY=key" > .env
echo "OPENAI_API_KEY=key" >> .env
```

.gitignore Configuration

```
# Python                        # Runtime/Generated
venv/                           agent_memory.json
__pycache__/                    ai_solution.json
*.pyc                           mission_solution.json
*.pyo                           environment.json
*.egg-info/
                                # Generated PDFs
# OS                            docs/*.pdf
.DS_Store
                                # Deprecated files
# IDE                           _deprecated/
.vscode/
.idea/                          # Environment secrets
*.swp                           .env, .env.local
```

Local Development Setup

QUICK START

```
# 1. Clone and setup
cd /path/to/isr_projects/isr_web
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

# 2. Set environment variables
export ANTHROPIC_API_KEY="your-api-key"
export PYTHONPATH=/path/to/isr_projects
```

```
# 3. Run the server
python -m uvicorn server.main:app --reload --port 8893

# 4. Open browser
open http://localhost:8893
```

run_planner.sh

```
#!/usr/bin/env bash
PLANNER_PORT=8893

# Kill existing process on port
lsof -ti:${PLANNER_PORT} | xargs kill

cd /path/to/isr_projects/isr_web
source venv/bin/activate
export PYTHONPATH=/path/to/isr_projects

python -m uvicorn server.main:app \
    --reload --port ${PLANNER_PORT}
```

API ENDPOINTS

```
GET  /              → Web UI
GET  /api/health    → Health check

POST /api/environment
     → Load mission environment

POST /api/solve
     → Run solver for all drones

POST /api/agent
     → Send message to AI agent

GET/POST /api/agent/memory
     → Manage agent memories
```

TROUBLESHOOTING

```
Port in use: lsof -ti:8893 | xargs kill   |   Import error: Check PYTHONPATH   |   API key: Verify .env
```

Docker Deployment

Dockerfile

```
FROM python:3.11-slim
WORKDIR /app

# Install dependencies
RUN apt-get update && apt-get install -y \
    gcc && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy sibling packages
COPY path_planning_core/ /app/path_planning_core/
COPY isr_editor/ /app/isr_editor/

# Copy application
COPY server/ /app/isr_web/server/
COPY webapp/ /app/isr_web/webapp/

ENV PYTHONPATH=/app
EXPOSE 8893

CMD ["python", "-m", "uvicorn", \
     "isr_web.server.main:app", \
     "--host", "0.0.0.0", "--port", "8893"]
```

docker-compose.yml

```
version: '3.8'

services:
  isr-web:
    build:
      context: ..  # isr_projects/
      dockerfile: isr_web/Dockerfile
    ports:
      - "8893:8893"
    environment:
      - ANTHROPIC_API_KEY=${ANTHROPIC_API_KEY}
      - OPENAI_API_KEY=${OPENAI_API_KEY:-}
    volumes:
      - ./agent_memory.json:/app/agent_memory.json
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f",
             "http://localhost:8893/"]
      interval: 30s
      timeout: 10s
      retries: 3
```

BUILD & RUN COMMANDS

```
# Option 1: Using docker-compose (recommended)
cd /path/to/isr_projects
export ANTHROPIC_API_KEY="your-key"
docker-compose -f isr_web/docker-compose.yml up --build

# Option 2: Manual docker build (from parent directory!)
cd /path/to/isr_projects
docker build -t isr-web -f isr_web/Dockerfile .
docker run -p 8893:8893 -e ANTHROPIC_API_KEY="your-key" isr-web

# Access: http://localhost:8893
```

Cloud Deployment Options

A: Cloud VM (AWS/GCP/Azure)

Cost: $5-20/month
Setup: Medium

1. Create VM (Ubuntu 22.04)
2. Install Docker
3. Clone repository
4. Set environment variables
5. Run docker-compose up -d

Pros: Full control, simple
Cons: Manual SSL, updates

B: PaaS (Railway/Render/Fly)

Cost: $5-25/month
Setup: Easy (git push deploys)

1. Connect GitHub repository
2. Set environment variables
3. Deploy automatically

Pros: Auto-deploy, HTTPS, scaling
Cons: Config for long agent calls

C: Container (ECS/Cloud Run)

Cost: $10-50/month
Setup: Complex

1. Push image to registry
2. Configure service
3. Set up load balancer
4. Configure auto-scaling

Pros: Production-ready, scalable
Cons: More setup, higher cost

D: Internal Server

Cost: $0 (existing infra)
Setup: Depends on IT

1. Request server access
2. Install Docker
3. Deploy container
4. Configure internal DNS

Pros: Data stays internal
Cons: IT approval, maintenance

RECOMMENDATION

For team sharing:
- Quick start → Railway/Render (B): Easy setup, auto-deploy, HTTPS included
- More control → AWS EC2 (A): ~$10/mo, full access, bring your own SSL
- Sensitive data → Internal server (D): On-premises, requires IT coordination