iSly AI Platform - Docker Deployment Guide

Overview

This guide provides step-by-step instructions to deploy the iSly AI agent platform using Docker Desktop. The platform consists of:
- **NextJS Web Application** (Port 3000)
- **PostgreSQL Database** (Port 5432)
- **Redis Cache** (Port 6379)
- **AI Agents Worker Service** (Background service)

Prerequisites

- Docker Desktop installed and running
- At least 4GB RAM available for containers
- Ports 3000, 5432, and 6379 available on your system

Quick Start

1. Environment Setup

First, update the environment variables in `.env.docker` with your actual API keys:

```
# Edit the .env.docker file
OPENAI_API_KEY=your_actual_openai_api_key
ANTHROPIC_API_KEY=your_actual_anthropic_api_key
GROQ_API_KEY=your_actual_groq_api_key
NEXTAUTH_SECRET=your_secure_random_string_here
POSTGRES_PASSWORD=your_secure_database_password
```

2. Build and Start Services

```
# Build all Docker images
docker compose --env-file .env.docker build

# Start all services in detached mode
docker compose --env-file .env.docker up -d
```

3. Verify Deployment

Check that all services are running:

```
# View running containers
docker compose ps

# Check service logs
docker compose logs web
docker compose logs postgres
docker compose logs ai-agents
```

Detailed Instructions

Building the Application

```
# Build only the web application
docker compose --env-file .env.docker build web

# Build only the AI agents service
docker compose --env-file .env.docker build ai-agents

# Build all services
docker compose --env-file .env.docker build
```

Starting Services

```
# Start all services
docker compose --env-file .env.docker up -d

# Start specific service
docker compose --env-file .env.docker up -d postgres

# Start with logs visible
docker compose --env-file .env.docker up
```

Accessing the Application

- **Web Interface**: http://localhost:3000
- **Health Check**: http://localhost:3000/api/health
- **Database**: localhost:5432 (username: isly_user)
- **Redis**: localhost:6379

Database Management

```
# Access PostgreSQL shell
docker exec -it isly-postgres psql -U isly_user -d isly_db

# Run database migrations (if using Prisma)
docker exec -it isly-web npx prisma migrate deploy

# View database tables
docker exec -it isly-postgres psql -U isly_user -d isly_db -c '\dt'
```

Monitoring and Logs

```
# View logs for all services
docker compose logs -f

# View logs for specific service
docker compose logs -f web
docker compose logs -f ai-agents
docker compose logs -f postgres

# View last 50 lines of logs
docker compose logs --tail=50 web
```

Stopping Services

```
# Stop all services
docker compose down

# Stop and remove volumes (WARNING: This deletes data)
docker compose down -v

# Stop specific service
docker compose stop web
```

Troubleshooting

Common Issues

1. Port Already in Use

If you get port binding errors:

```
# Check what's using the port
lsof -i :3000
lsof -i :5432

# Kill the process or change ports in docker-compose.yml
```

2. Database Connection Issues

```
# Check if PostgreSQL is ready
docker exec -it isly-postgres pg_isready -U isly_user

# Restart database service
docker compose restart postgres
```

3. Web Application Not Starting

```
# Check web service logs
docker compose logs web

# Rebuild web service
docker compose build --no-cache web
docker compose up -d web
```

4. AI Agents Not Processing

```
# Check AI agents logs
docker compose logs ai-agents

# Restart AI agents service
docker compose restart ai-agents
```

Health Checks

```
# Web application health
curl http://localhost:3000/api/health

# Database health
docker exec -it isly-postgres pg_isready -U isly_user

# Redis health
docker exec -it isly-redis redis-cli ping
```

Development Workflow

Making Code Changes

1. Make your code changes
2. Rebuild the affected service:
   `bash`
   `docker compose build web`
   `docker compose up -d web`

Database Schema Changes

1. Update your Prisma schema or SQL files
2. Rebuild and restart:
   `bash`
   `docker compose build web`
   `docker compose restart web`
   `docker exec -it isly-web npx prisma migrate deploy`

Adding New Environment Variables

1. Update `.env.docker`
2. Update `docker-compose.yml` if needed
3. Restart services:
   `bash`
   `docker compose down`
   `docker compose --env-file .env.docker up -d`

Production Considerations

Security

- Change all default passwords in `.env.docker`
- Use strong, unique secrets for NEXTAUTH_SECRET
- Consider using Docker secrets for sensitive data
- Enable SSL/TLS in production

Performance

- Increase `WORKER_CONCURRENCY` for AI agents based on your server capacity
- Monitor resource usage with `docker stats`
- Consider using Docker Swarm or Kubernetes for scaling

Backup

```
# Backup database
docker exec -it isly-postgres pg_dump -U isly_user isly_db > backup.sql

# Backup volumes
docker run --rm -v isly_postgres_data:/data -v $(pwd):/backup alpine tar czf /backup/
postgres_backup.tar.gz /data
```

Docker Desktop Integration

Using Docker Desktop GUI

1. Open Docker Desktop
2. Navigate to “Containers” tab
3. You should see the “ubuntu” stack with all services
4. Click on any service to view logs, stats, and manage containers
5. Use the “Actions” menu to start/stop/restart services

Resource Monitoring

- Monitor CPU and memory usage in Docker Desktop
- Adjust resource limits in Docker Desktop settings if needed
- Check container health status indicators

Support

If you encounter issues:
1. Check the logs using the commands above
2. Verify all environment variables are set correctly
3. Ensure Docker Desktop has sufficient resources allocated
4. Check that all required ports are available

For additional support, check the container logs and Docker Desktop dashboard for detailed error messages.