Swift Signals

Technical installation manual

Introduction

Swift Signals is a comprehensive traffic optimization system designed to analyze traffic patterns, simulate intersection performance, and optimize traffic light timing using artificial intelligence algorithms.

The system consists of multiple components that need to be installed and configured:

- **Frontend Application**: React-based web interface for traffic monitoring and control
- **Backend API-Gateway**: RESTful API service handling frontend and microservice communication
- **Backend Microservices**: GRPC microservices handling traffic data processing and AI algorithms
- **Databases**: Data storage for user management and traffic intersections, simulations, and optimization results
- **Container Orchestration**: Docker-based deployment for consistent environments

This guide will walk you through the installation process for development and production environments across different operating systems (Windows, macOS, and Linux).

Prerequisites

Before installing Swift Signals, ensure you have the following software installed on your system:

Git

- **Windows**: Git 2.42.0+ (Download from git-scm.com)
- **macOS**: Git 2.42.0+ (Install via Homebrew: **brew install git** or Xcode Command Line Tools)
- **Linux**: Git 2.42.0+ (Install via package manager: **sudo apt install git** or **sudo yum install git**)

Docker

- **Windows**: Docker Desktop 4.24.0+ (Download from docker.com)
- **macOS**: Docker Desktop 4.24.0+ (Download from docker.com or install via Homebrew: brew install --cask docker)
- **Linux**: Docker Engine 24.0.0+ and Docker Compose 2.21.0+

```
# Ubuntu/Debian
sudo apt update
sudo apt install docker.io docker-compose-v2

# CentOS/RHEL/Fedora
sudo yum install docker docker-compose
```

Docker Compose

- **Windows/macOS**: Included with Docker Desktop
- **Linux**: Docker Compose 2.21.0+

```
# If not installed with Docker
sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Installation

Clone the Swift Signals repository

```
git clone https://github.com/COS301-SE-2025/Swift-Signals.git
cd SwiftSignals
```

For Development

Move to the development directory

```
cd deployments/development
```

Copy the environment variables Windows

(Command Prompt):

```
copy .env.Example .env
```

Windows (PowerShell):

```
Copy-Item .env.Example .env
```

macOS/Linux:

```
cp .env.Example .env
```

Edit environment variables (Optional)

Before building, you may want to customize the environment variables in the .env file:

Windows:

```
notepad .env
```

macOS:

```
nano .env
# or
vim .env
# or
code .env  # if VS Code is installed
```

Linux:

```
nano .env
# or
vim .env
# or
gedit .env  # for GUI editor
```

Build using Docker Compose

```
docker-compose up -d --build
```

Visit the application

Once the containers are running, visit the application at:

- Development URL: http://localhost:80
- API Documentation: http://localhost:9090/docs

Verify installation

Check if all services are running:

```
docker-compose ps
```

View application logs:

```
docker-compose logs -f
```

Deployment

For Production

Move to the production directory

```
cd deployments/production
```

Copy the environment variables Windows

(Command Prompt):

```
copy .env.Example .env
```

Windows (PowerShell):

```
Copy-Item .env.Example .env
```

macOS/Linux:

```
cp .env.Example .env
```

Configure production environment variables

Important: Before deploying to production, update the **.env** file with appropriate production values:

```
# Edit the .env file with production-specific configurations
# - Database credentials
# - API keys
# - Security tokens
# - Domain names
# - SSL certificates
```

Deploy using Docker Compose

```
docker-compose up -d --build
```

Production verification

Verify all services are running correctly:

```
docker-compose ps
docker-compose logs --tail=100
```

Troubleshooting

Common Issues

Port conflicts:

```
# Check which process is using the port
# Windows
netstat -ano | findstr :80

# macOS/Linux
lsof -i :80
```

Permission issues (Linux):

```
# Add user to docker group
sudo usermod -aG docker $USER
# Log out and back in for changes to take effect
```

Container build failures:

```
# Clean up Docker resources
docker system prune -a
docker-compose down --volumes
docker-compose up -d --build
```