Deployment Guide

CSC-640-MI-Part2 Telemetry API

Step-by-step deployment using Docker Compose

Prerequisites

Required Software:

- Docker Desktop (includes Docker Compose)
- Git (to clone the repository)

Optional:

- GitHub OAuth App (for authentication features)

Step 1: Clone the Repository

```
git clone https://github.com/islerm2-nku/CSC-640-MI-Part2.git
cd CSC-640-MI-Part2
```

What's included:

- FastAPI application code
- Python telemetry parser (irsdk)
- Docker configuration files
- SQLAlchemy models and database setup
- Example telemetry files
- OAuth authentication setup

Step 2: Configure Environment (Optional)

For GitHub OAuth:

1. Create a GitHub OAuth App at https://github.com/settings/developers
2. Copy `.env.example` to `.env`:

   ```
   cp .env.example .env
   ```

3. Edit `.env` with your OAuth credentials:

   ```
   OAUTH_CLIENT_ID=your_client_id
   OAUTH_CLIENT_SECRET=your_client_secret
   OAUTH_REDIRECT_URI=http://localhost/auth/callback
   ```

Note: OAuth is optional. API will work without it, but authentication-required endpoints will be inaccessible.

Step 3: Build and Start Containers

Using the setup script (recommended):

```
./setup.sh
```

Or manually:

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

What this does:

- Builds the web container (FastAPI + Uvicorn)
- Starts MySQL 8.0 database container
- Creates network between services
- Exposes port 80 for API access
- Exposes port 5678 for Python debugging

Step 4: Verify Database Tables

Tables are created automatically by SQLAlchemy.

Verify tables exist:

```
docker compose exec db mysql -uappuser -papppass -D app -e "SHOW TABLES;"
```

Expected tables:

- `session_info` - session metadata
- `weather` - track conditions
- `driver` - driver and car information
- `attribute_values` - telemetry time-series data

Output:

```
+--------------------+
```

Step 5: Verify Deployment

Test the API is running:

```
curl -i http://localhost/sessions
```

Or access interactive documentation:

- Swagger UI: http://localhost/docs
- ReDoc: http://localhost/redoc

Deployment Verification

alt text

nginx response

Key indicators:

- ✅ `HTTP/1.1 200 OK` - successful response
- ✅ `Server: uvicorn` - Uvicorn ASGI server is running
- ✅ `Content-Type: application/json` - API is returning JSON
- ✅ FastAPI auto-generated docs available at `/docs`

Container Status Check

Verify all containers are running:

```
docker compose ps
```

Expected output:

```
NAME                          STATUS    PORTS
csc-640-mi-part2-web-1        Up        0.0.0.0:80->80/tcp, 0.0.0.0:5678->5678/tcp
csc-640-mi-part2-db-1         Up        0.0.0.0:3306->3306/tcp
```

Alternative:

```
docker ps
```

Viewing Logs

Web container (FastAPI + Uvicorn):

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

Database container:

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

All containers:

```
docker compose logs -f
```

View real-time FastAPI logs:

FastAPI runs with hot-reload enabled, so code changes are reflected immediately.

Troubleshooting

Container won't start:

- Check port 80 isn't already in use: `lsof -i :80`
- Check port 3306 isn't in use: `lsof -i :3306`
- Restart Docker Desktop
- Run: `docker compose down && docker compose up --build -d`

Database connection fails:

- Ensure MySQL is ready: wait 15 seconds after starting
- Check DB logs: `docker compose logs db`
- Verify health check: `docker compose ps` (should show "healthy")

OAuth not working:

Stopping the Application

Stop containers (preserve data):

```
docker compose stop
```

Stop and remove containers:

```
docker compose down
```

Stop and remove containers + volumes (deletes database):

```
docker compose down -v
```

Restarting the Application

After stopping:

```
docker compose up -d
```

Quick restart:

```
./setup.sh
```

No need to rebuild unless:

- Python dependencies changed (requirements.txt)
- Dockerfile modified
- Volumes were deleted with `-v` flag

Code changes are hot-reloaded automatically (FastAPI development mode)

Summary

Deployment is complete when:

1. ✅ Containers are running (`docker compose ps`)
2. ✅ Database tables created (verified with `SHOW TABLES`)
3. ✅ API responds to requests at http://localhost
4. ✅ Interactive docs accessible at http://localhost/docs
5. ✅ Upload endpoint accepts `.ibt` files (with auth token)

Total deployment time: ~2 minutes

Resources:

- Interactive API docs: http://localhost/docs
- API overview: `api-overview.md`
- Example files: `telemetry/` directory

Testing Authentication

Get OAuth token:

1. Visit: http://localhost/auth/oauth/authorize
2. Authorize with GitHub
3. Copy the `access_token` from response

Test protected endpoint:

```
curl -H "Authorization: Bearer YOUR_TOKEN" \
  http://localhost/telemetry/upload
```

Or use Swagger UI:

1. Go to http://localhost/docs
2. Click "Authorize" button (🔒)
3. Paste your token

Questions?

Need help?

- Check logs: `docker compose logs`
- Review README: `README.md`
- Inspect containers: `docker inspect <container>`
- View API docs: http://localhost/docs

Thank you!