Minimal OpenTelemetry for InterSystems IRIS

This project sets up a **local OpenTelemetry environment** for **InterSystems IRIS**, providing **metrics**, **traces**, **logs** collection.
It uses Docker Compose to orchestrate all components and demonstrate how IRIS integrates with modern observability tools.

Prerequisites

- **Docker**
- **Docker Compose**

Verify installation:

```
docker --version
docker compose version
```

Quick Start

1.Start the environment

```
docker compose up -d
```

This launches:

- IRIS with a CPF merge file enabling OpenTelemetry metrics
- OpenTelemetry Collector (contrib build)
- Prometheus
- Jaeger
- Loki and Grafana

2.How to enable OTel metrics/logs in IRIS

The **CPF merge file** is already included and automatically loaded in `docker-compose.yaml`.
It contains the following configuration:

```
[Monitor]
OTELMetrics=1
OTELLogs=1
OTELLogLevel=INFO
```

This enables the IRIS monitor to expose metrics and logs via the OpenTelemetry exporter.

3.Trigger test traces from IRIS

Open a terminal session in the IRIS container:

```
docker compose exec iris iris session IRIS
```

Then run the built-in trace demo:

```
Do ##class(%Trace.Tracer).Test()
```

This sends trace spans from IRIS to the OpenTelemetry Collector, which then exports them to Jaeger.

Viewing Results

Jaeger - Traces

Open http://localhost:16686

1. Select your **Service Name** ("test_service" in this example)
2. Click **Find Traces**
3. Expand a trace to view span details, durations, and relationships

---

Prometheus - Metrics

Open http://localhost:9090

Try example queries:

```
iris_db_size_mb_megabytes
iris_cpu_usage
iris_res_seize_total
```

Grafana - Traces+Metrics+Logs

Open http://localhost:3000/drilldown

Debug output

You can also view logs and traces from debug output

```
docker compose logs -f otel-collector
```

Cleanup

```
docker compose down -v
```