Contributing to EDITO Platform

Deploy and share what you Create

Samuel Fooks

Flanders Marine Institute (VLIZ)

Funded by the European Union

What We'll Cover Today

- ✅ **Add Tutorial** - Creating interactive R/Python tutorials
- ✅ **Add Service** - Deploying containerized web applications
- ✅ **Add Process** - Deploying computational workflows
- ✅ **Add Data** - Contributing datasets using STAC standards

ADD TUTORIAL

Add Tutorial - Overview

Creating Interactive Tutorials

- **Goal:** Share your knowledge as interactive tutorials on EDITO
- **Format:** R Markdown (.Rmd) or Jupyter Notebooks
- **Platform:** EDITO Datalab Tutorials Catalog

Tutorial Requirements

What You Need

- ✅ **R** with RStudio (or Python with Jupyter)
- ✅ **GitHub account** - Public repository
- ✅ **EDITO account** - Beta tester access
- ✅ **Tutorial content** - R Markdown file with code and explanations

Tutorial Workflow

Step-by-Step Process

1. **Create tutorial** - Write `.Rmd` file with content
2. **Push to GitHub** - Make repository public
3. **Configure service** - Set up deployment URL in EDITO Datalab
4. **Test locally** - Verify tutorial runs correctly
5. **Register** - Add to `tutorials.json` in GitLab
6. **Submit** - Create merge request

Tutorial Structure

Recommended Folder Structure

```
my_tutorial/
├── tutorial.Rmd        # Main tutorial file
├── data/               # Data files
├── docker-compose.yml # Optional local testing
└── README.md           # Documentation
```

Key: Include good README.md and separate data/assets

Tutorial Registration

Adding to EDITO Catalog

File: `tutorials.json` in GitLab repository

Required fields:

- Name, abstract, authors
- Category, tags, types
- GitHub repository URL
- Deployment URL from EDITO Datalab

Process: Clone repo → Add entry → Merge request

ADD SERVICE

Add Service - Overview

Deploying Web Applications

- **Goal:** Deploy containerized web applications to EDITO Datalab
- **Examples:** R Shiny apps, Python Dash, Jupyter notebooks
- **Key Technology:** Docker + Kubernetes + Helm

Service vs Tutorial vs Process

Type	Purpose	Interaction	Example
Service	Interactive web app	User interface	Dashboard, API
Tutorial	Learning content	Step-by-step	R Markdown guide
Process	Data transformation	Batch job	ML model

Service Workflow

Step-by-Step Process

1. **Dockerize** - Create Dockerfile for your app
2. **Build & Push** - Push to container registry (GitHub Packages, Docker Hub)
3. **Create Helm Chart** - Configure Kubernetes deployment
4. **Test in Playground** - Deploy to service playground
5. **Submit** - Merge request for production

Dockerfile Example

Containerizing R Shiny App

```dockerfile
FROM rocker/shiny:4.5.0

# Install system dependencies
RUN apt-get update && apt-get install -y \
    libcurl4-openssl-dev libssl-dev

# Install R packages
RUN R -e "install.packages(c('shiny', 'arrow', 'leaflet'))"

# Copy app files
COPY app/ /srv/shiny-server/

EXPOSE 3838
CMD ["/start.sh"]
```

Helm Chart Structure

Kubernetes Deployment

```
my_service/
├── Chart.yaml          # Chart metadata
├── values.yaml         # Default values
├── values.schema.json  # UI configuration
└── templates/
    └── deployment.yaml # Kubernetes resources
```

Service - values.yaml Configuration

```yaml
service:
  image:
    version: "ghcr.io/yourusername/view-parquet:1.0.3"

networking:
  service:
    port: 3838
```

Components:

- Image repository and tag
- Service port configuration
- Resource limits (CPU, memory)
- Ingress configuration

ADD PROCESS

Add Process - Overview

Deploying Computational Workflows

- **Goal:** Deploy batch processing jobs and models to EDITO

What is a Process?

- Takes input data → transforms → output data
- Runs as batch job (not interactive)
- Examples: ML models, data pipelines, simulations

Process Workflow Pattern

Three-Stage Pattern

1. **Download** - Input data from S3 → `/data/input`
2. **Process** - Run scripts → output to `/data/output`
3. **Upload** - Results from `/data/output` → S3 storage

Key: All containers share `/data` directory

Process Workflow

Step-by-Step Process

1. **Dockerize** - Create Dockerfile with your scripts
2. **Build & Push** - Push to container registry
3. **Create Helm Chart** - Configure Kubernetes Job
4. **Define Commands** - Set processing commands in values.yaml
5. **Deploy** - Submit to process playground
6. **Submit** - Merge request for production

Process Structure

Example Directory Structure

```
my_process/
├── Dockerfile
├── Scripts/
│   ├── 01_data_preparation.R
│   └── 02_model_analysis.R
└── README.md
```

Helm Chart:

- `Chart.yaml` - Chart metadata
- `values.yaml` - Processing commands
- `templates/job.yaml` - Kubernetes Job definition

Kubernetes Job Template

Key Features

- **Init Container** - Downloads input data from S3
- **Processing Containers** - Run your custom commands sequentially
- **Upload Container** - Uploads results to S3
- **Shared Volume** - `/data` directory for all containers
- **Resource Management** - CPU and memory limits

ADD DATA

What is STAC?

SpatioTemporal Asset Catalog

STAC = Standardized JSON metadata format

- Describes **when, where, and what** your data contains
- Links to actual data files
- Searchable and discoverable
- Open standard (v1.0.0)

Specification: stacspec.org

STAC Structure

Key Components

- 📦 **Catalog** - Top-level container
- 📚 **Collection** - Groups related Items
- 📄 **Item** - Individual dataset with geometry, properties, assets
- 🔗 **Asset** - Link to actual data file

Required Fields:

- `id` , `type` , `stac_version`
- `geometry` (GeoJSON)
- `properties` (datetime)
- `assets` (data file links)

Creating STAC Items

From Your Data Files

Example Demo Script: dataset_to_stac_item

What it does:

- Reads NetCDF, Zarr, or Parquet files
- Extracts spatial bounds (lat/lon)
- Extracts temporal range (datetime) or asks for input
- Reads metadata (institution, title, license)
- Tries to make valid STAC item

Add Data - Workflow

Step-by-Step Process

1. **Prepare your data** - Ensure file is accessible via URL
2. **Create STAC item** - Obtain all appropriate metadata for the data (ex try dataset_to_stac_item.py)
3. **Post to EDITO** - UNDER CONSTRUCTION

Summary

What We Covered

- ✅ **Add Data** - STAC items from NetCDF/Zarr/Parquet
- ✅ **Add Tutorial** - R Markdown tutorials on GitHub
- ✅ **Add Service** - Dockerized web applications
- ✅ **Add Process** - Batch processing workflows

All use: Docker, GitLab, and EDITO playgrounds

Getting Started

Next Steps

1. **Get Account** - Sign up as beta tester at edito-infra.eu
2. **Access GitLab** - Create account at gitlab.mercator-ocean.fr
3. **Choose Your Path** - Data, Tutorial, Service, or Process
4. **Follow Documentation** - Check workshop repository

Repository: github.com/EDITO-Infra/edito-workshops-presentations

Resources

Helpful Links

Documentation:

- **EDITO Docs:** docs.dive.edito.eu
- **Workshop Repo:** github.com/EDITO-Infra/edito-workshops-presentations
- **STAC Spec:** stacspec.org

Platforms:

- **EDITO Datalab:** datalab.dive.edito.eu
- **GitLab:** gitlab.mercator-ocean.fr/pub/edito-infra

Contact: edito-infra-dev@mercator-ocean.eu