EUROPEAN LANGUAGE GRID

META-FORUM 2021, 15/16/17-11-2021

Integrating services into ELG

D. Galanis

http://www.european-language-grid.eu

ELG platform

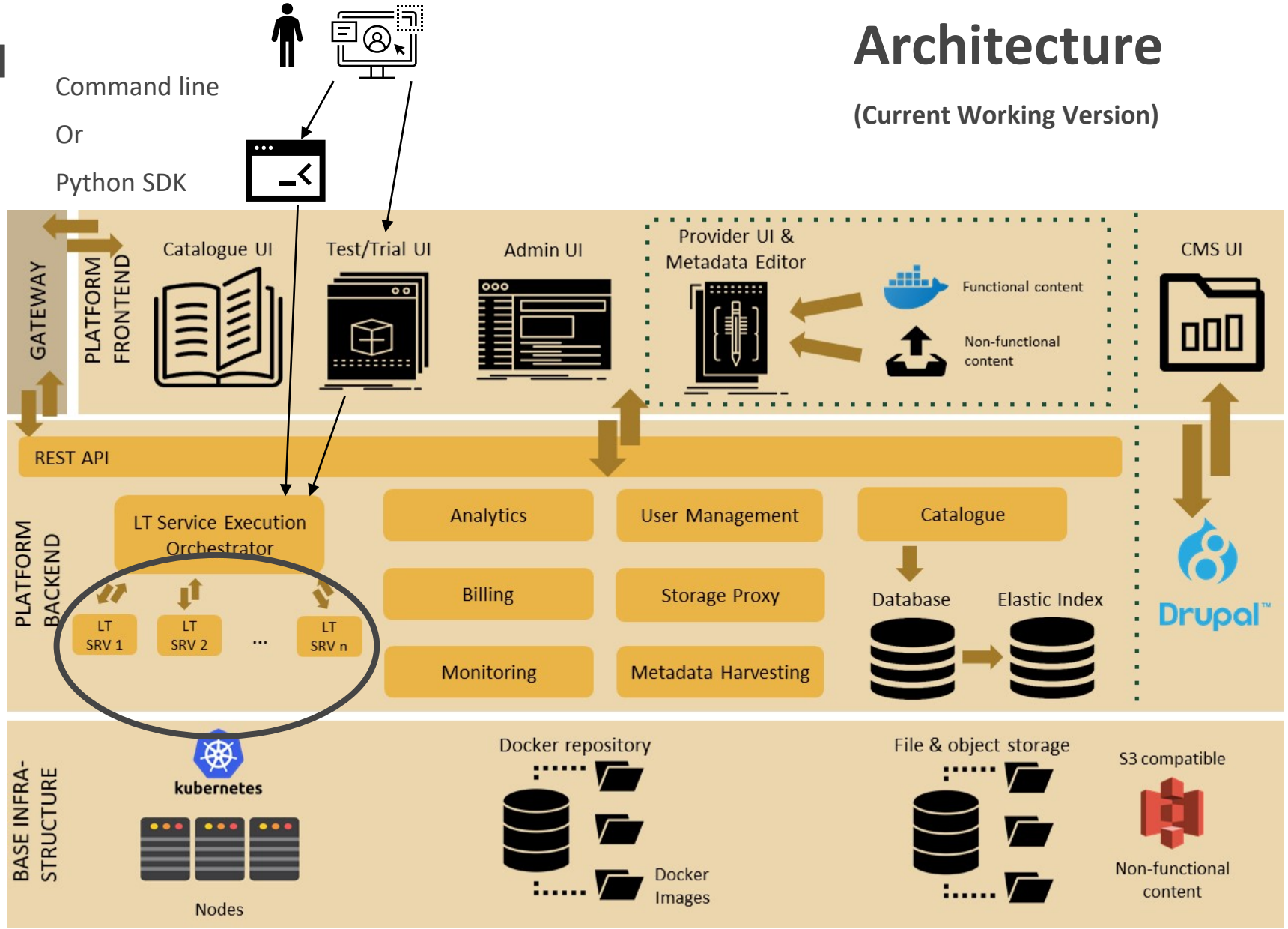
- It is web-based: https://live.european-language-grid.eu
- It is a catalogue for resources:
  - Each resource is documented with the required metadata
    - Datasets, **integrated LT services**, not integrated LT services
    - Machine learning models, lexica
- "Published" resources are visible by anyone; i.e. log-in is not required
- Log-in is required for using **integrated services**
- Anyone can become a **provider** and contribute a resource
- A validation process is required for publication

EUROPEAN LANGUAGE GRID

Architecture

(Current Working Version)

- All integrated LT services run as Docker containers in the Kubernetes cluster of ELG
- All integrated LT services can be called by a user via a common public API offered by the LT Service Execution server.
- Public API documentation: https://european-language-grid.readthedocs.io/en/stable/all/A3_API/LTPublicAPI.html

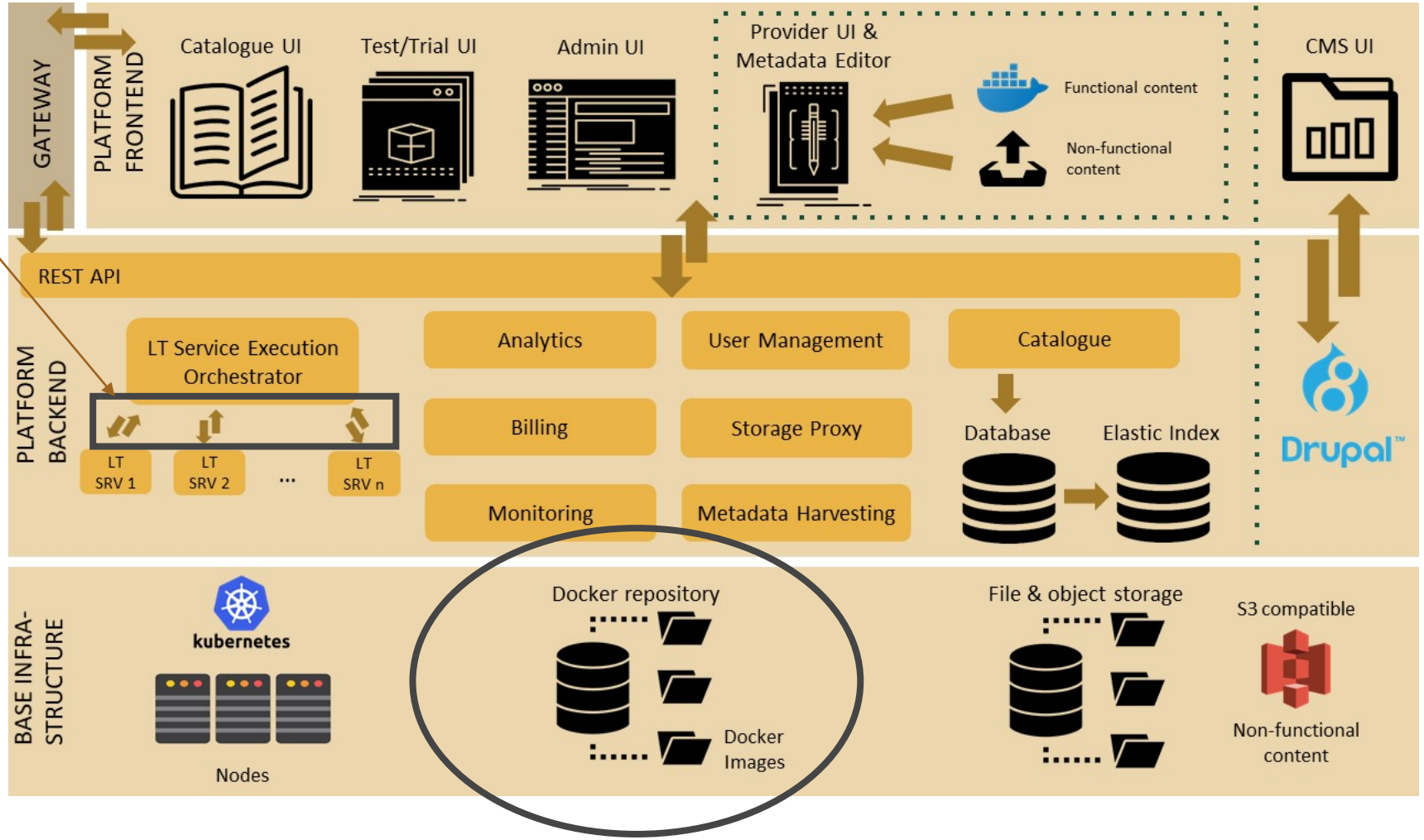


EUROPEAN LANGUAGE GRID

Architecture

(Current Working Version)

- An integrated LT service has to implement the internal LT Service API
- https://european-language-grid.readthedocs.io/en/stable/all/A3_API/LTInternalAPI.html
- Docker images are stored in Docker registries; DockerHub, Gitlab etc.



Provide an ELG-compatible LT service

- Currently, an LT service must fall into one of the following broad categories:
  - Information Extraction (IE)
  - Text Classification (TC)
  - Machine Translation (MT)
  - Automatic Speech Recognition (ASR)
  - Text to Speech Generation (TTS)
- Requirements:
  1. Implement the Internal LT API -> Expose an ELG compatible endpoint
  2. Dockerize

Otherwise it is a special case…and you should contact with the ELG tech. team.

(e.g. COREON SPARQL endpoint https://live.european-language-grid.eu/catalogue/lcr/8099)

Provide an ELG-compatible LT service: Technical Requirements

- **(1) Expose an ELG compatible endpoint:**
  - You MUST create an application that exposes an HTTP endpoint for the LT tool(s).
  - The application MUST consume requests that follow the ELG JSON format, call the underlying LT tool and produce responses again in the ELG JSON format (ELG LT Internal API)
  - E.g. for services that take as input plain text and return annotations (IE).

```
{
  "type":"text",
  "params":{...},   /* optional */
  "content":"The text of the request",
  // mimeType optional - this is the default if omitted
  "mimeType":"text/plain",
  "features":{ /* arbitrary JSON metadata about this content, optional */ },
  "annotations":{ /* optional */
    "<annotation type>":[
      {
        "start":number,
        "end":number,
        "features":{ /* arbitrary JSON */ }
      }
    ]
  }
}
```

```
{
  "response":{
    "type":"annotations",
    "warnings":[...], /* optional */
    "features":{...}, /* optional */
    "annotations":{
      "<annotation type>":[
        {
          "start":number,
          "end":number,
          "features":{ /* arbitrary JSON */ }
        }
      ]
    }
  }
}
```

Provide an ELG-compatible LT service: Technical Requirements

- **(2) Dockerization**: You MUST Dockerize the application and upload the respective image(s) in a Docker Registry, e.g. GitLab.

Dockerfile 885 Bytes

Edit | Web IDE | Replace | Delete

```
# Base image.
FROM openjdk:8-jdk-alpine
#VOLUME /tmp

# SET TARGET DIR
ENV TARGETDIR /elg/

# Install tini and create unprivileged user
RUN apk add --no-cache tini && \
      addgroup --gid 1001 "elg" && \
      adduser --disabled-password --gecos "ELG User,,," \
      --home /elg --ingroup elg --no-create-home --uid 1001 elg

# Create target dir.
RUN install -d -o elg -g elg $TARGETDIR
# Copy everything to target dir.
COPY --chown=elg:elg dockerCmd ${TARGETDIR}dockerCmd
# Copy/Rename server app jar.
#ADD --chown=elg:elg /elg-ilsp-lt-services-rest/target/elg-ilsp-services-rest-0.0.1-SNAPSHOT-exec.jar ${TARGETDIR}dockerCmd/app.jar
ADD --chown=elg:elg /elg-ilsp-services-rest-0.0.1-SNAPSHOT-exec.jar ${TARGETDIR}dockerCmd/app.jar

USER elg:elg

# Set working dir.
WORKDIR ${TARGETDIR}dockerCmd
#/elg/

# Start
ENTRYPOINT ["sh", "runInContainer.sh"]
```

Id/location of the image

A Docker **image** of an application contains the actual code of the tool and all required dependencies that are required to run it; e.g., the operating system, frameworks, settings, config files and libraries etc

Containers. are instantiations of images and can be thought of as lightweight virtual machines.

An image is created based on **Dockefile**

Command line:

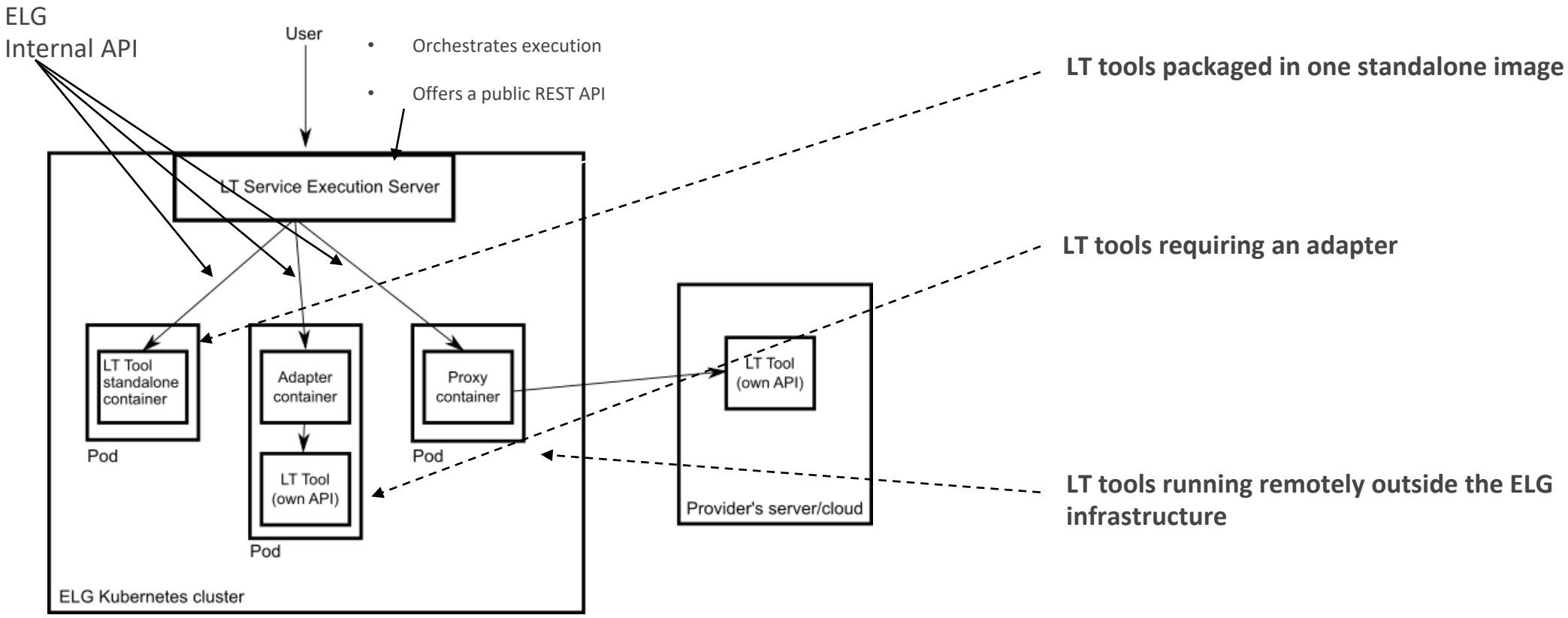
Login: docker login registry.gitlab.com

Create: docker build -t registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services .

Push: docker push registry.gitlab.com/ilsp-nlpli-elg/elg-ilsp-lt-services

Provide an ELG-compatible LT service: Technical Requirements

- Three integration options are available.

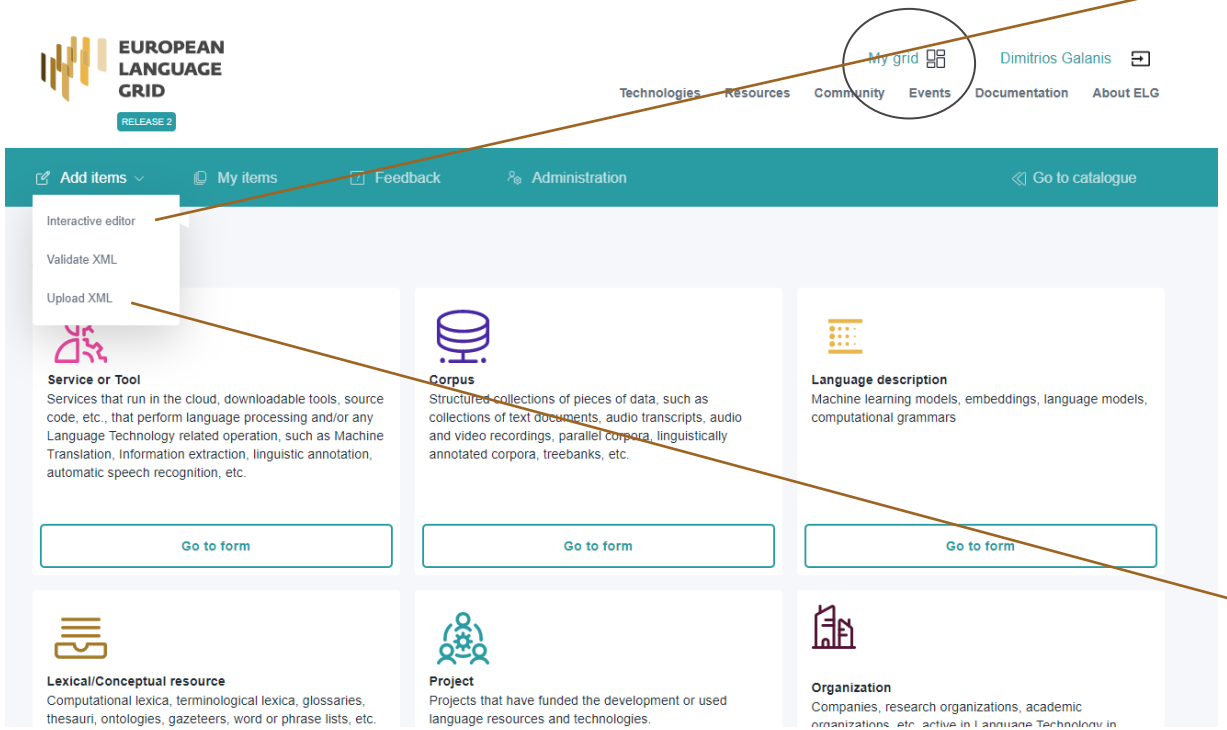


Provide an ELG-compatible LT service

- Documentation: https://european-language-grid.readthedocs.io/en/stable/all/3_Contributing/Service.html?highlight=boot#dockerize-your-service
- We provide examples on how to Dockerize
  - a Python-based tool
    - https://gitlab.com/european-language-grid/usfd/simple-python-tokeniser
  - a Java-based tool
    - https://gitlab.com/european-language-grid/usfd/elg-spring-boot-starter

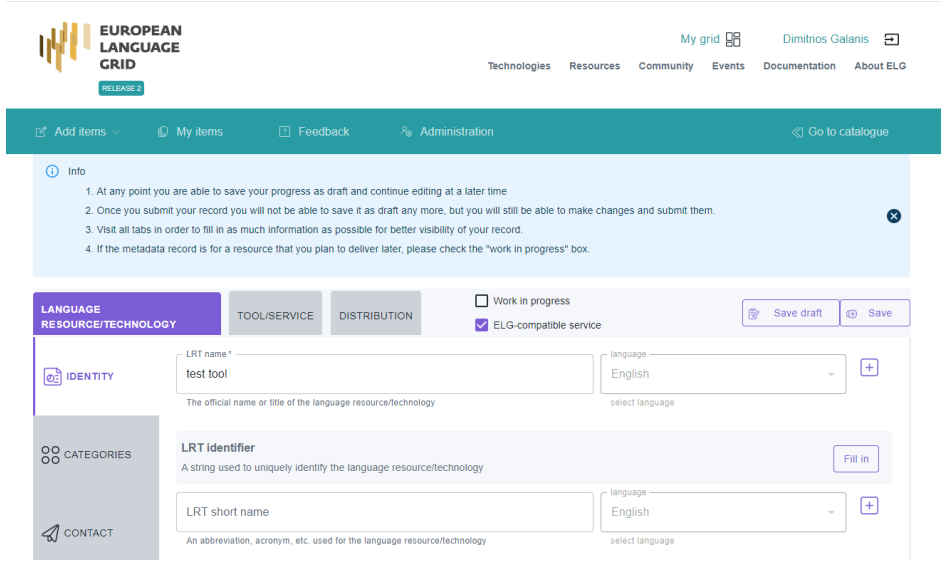
Provide an ELG-compatible LT service

Provide Metadata

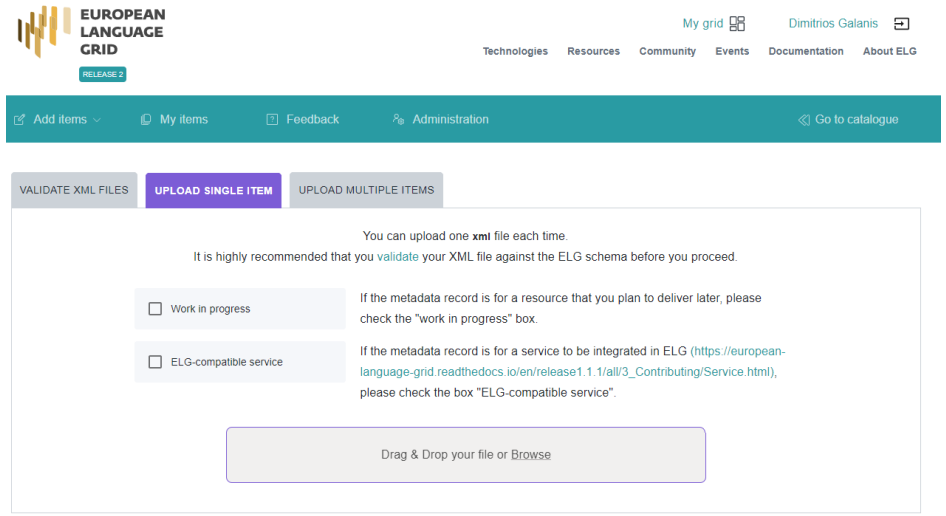


If a metadata record is syntactically valid then you can **submit it for publication**

Option A: interactive editor



Option B: upload metadata



Provide an ELG-compatible LT service

- The LT Service is assigned to two “validators”
  - A: Technical, metadata
  - B: Legal
- The LT tool is deployed to the ELG platform (@kubernetes) by validator **A**
- The LT Service is tested by **A** and the **LT provider**
  - Test via trial UIs
  - Check container logs, troubleshoot …
- When both validators approve the tool then it is **automatically published** to the catalogue.

Provide an ELG-compatible LT service