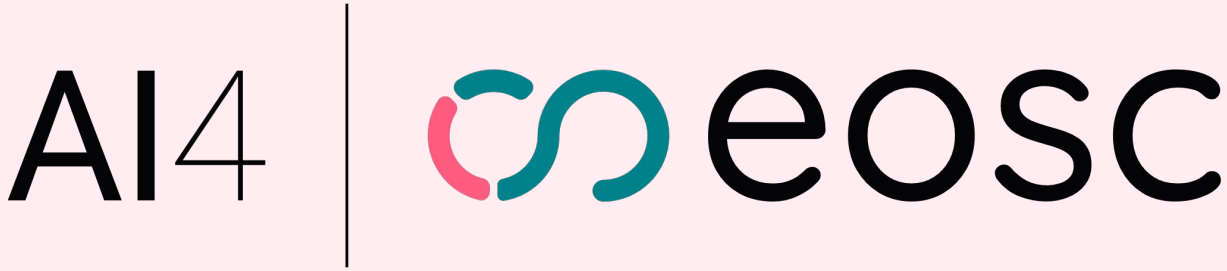
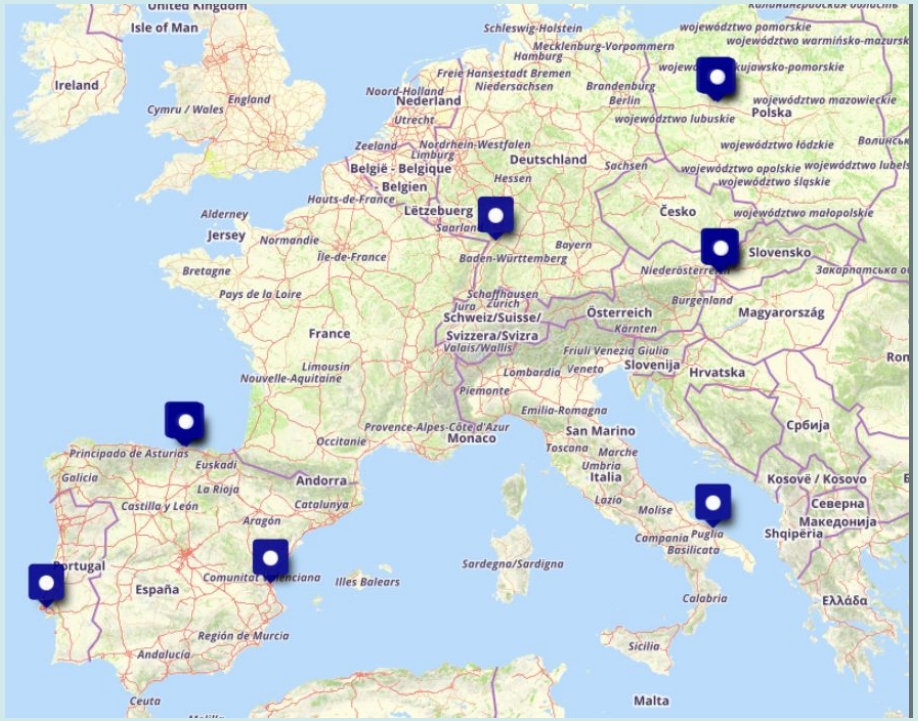


AI4EOSC templates

B. Esteban, K. Alibabaei, L. Berberi, V. Kozlov

Co-funded by
the European Union



CSIC
Consejo Superior de Investigaciones Científicas

KIT
Karlsruher Institut für Technologie

LIP

Ústav Informatiky
Slovenská akadémia vied

Universitat Politècnica de València

INFN
Istituto Nazionale di Fisica Nucleare

PSNC

WODR

MicroStep-MIS

predictia

AI4 | eosc

Co-funded by the European Union

AI4EOSC

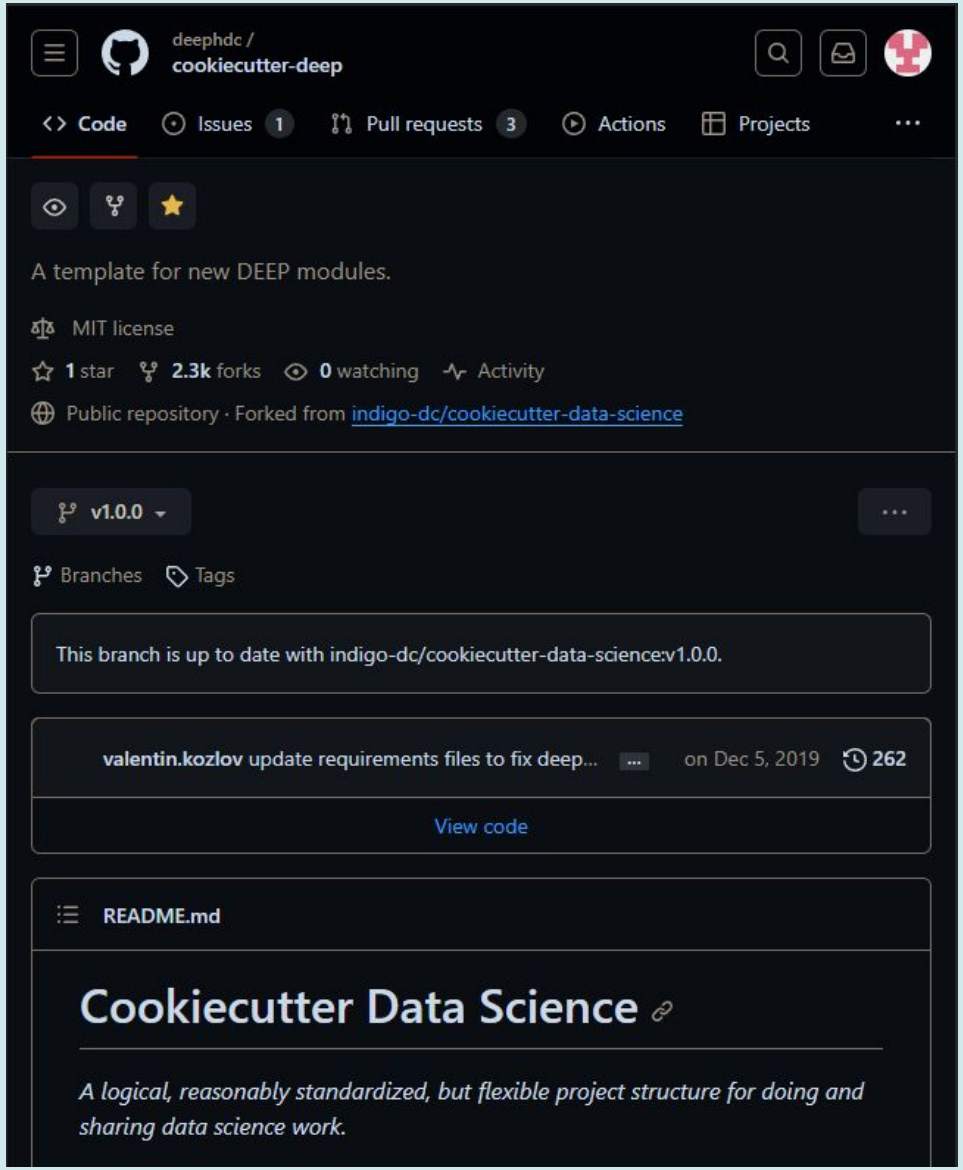
Artificial Intelligence for the #EOSC

- Evolution of the DEEP Hybrid DataCloud platform
- HORIZON-INFRA-2021-EOSC-01-04 call
- Runs September 1st 2022 – August 2025 (36 months)
- 7 academic partners
  + 2 SME
  + 1 non-profit organization

Advanced features for distributed, federated, composite learning, metadata provenance, MLOps, event-driven data processing, and provision of AI/ML/DL services

Why templates?

- Provide Consistency and Uniformity
- Speed-up development time
- Apply best learning practices
- Reduction in Errors
- Let you focus on what is important
- Are a great source of inspiration



Templates

https://github.com/deephdc/cookiecutter-deep

Versions:

- **master**: this is what 99% of users are probably looking for. Simple, minimal template, with the minimum requirements to integrate your code in DEEP.
- **child-module**: this is a fork of the master branch specifically tailored to users performing a retraining of an existing module. It only creates a Docker repo whose container is based on an existing module's Docker image.
- **advanced**: this is a more advanced template. It makes more assumptions on how to structure projects and adds more files than those strictly needed for integration.

Current state of advanced template

- Basic documentation generation files.
- Basic testing with unittest
- Customization of multiple API arguments by Schemas
- Basic files to start implementing CICD with Jenkins
- Basic logging and use of HTTP Responses

But, there is a lot of room for improvement.

Advantages of advanced template

Multiple integration modes:

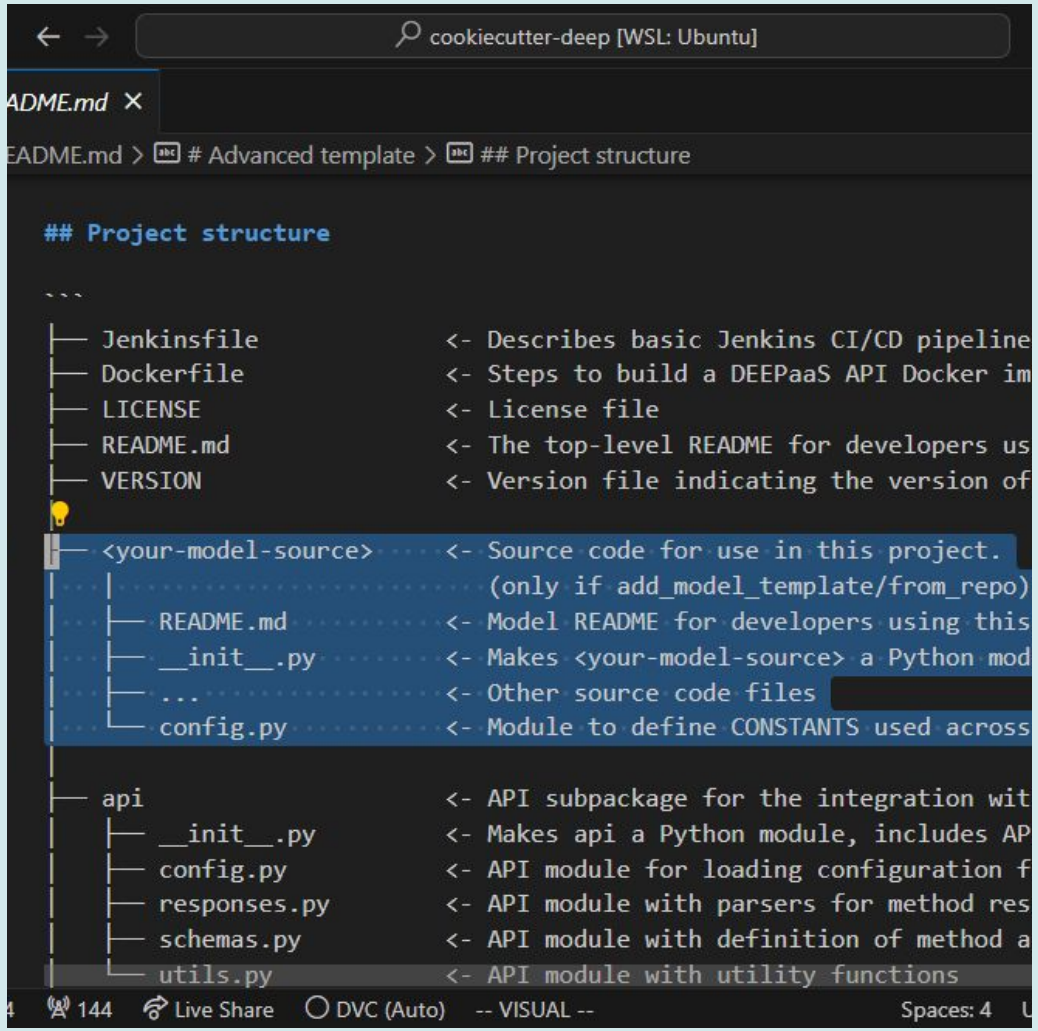
1. From model base template
2. From repository as submodule
3. As requirement for the API

Additional code features:

1. Templates for responses (pdf.)
2. Schemas module for parameters
3. Pre-made testing with pytest
4. Debug configurations for vscode

And more to come:

1. Drift detection with FROUROS
2. Jenkins SQA baseline v2
3. Data version control (with DVC)



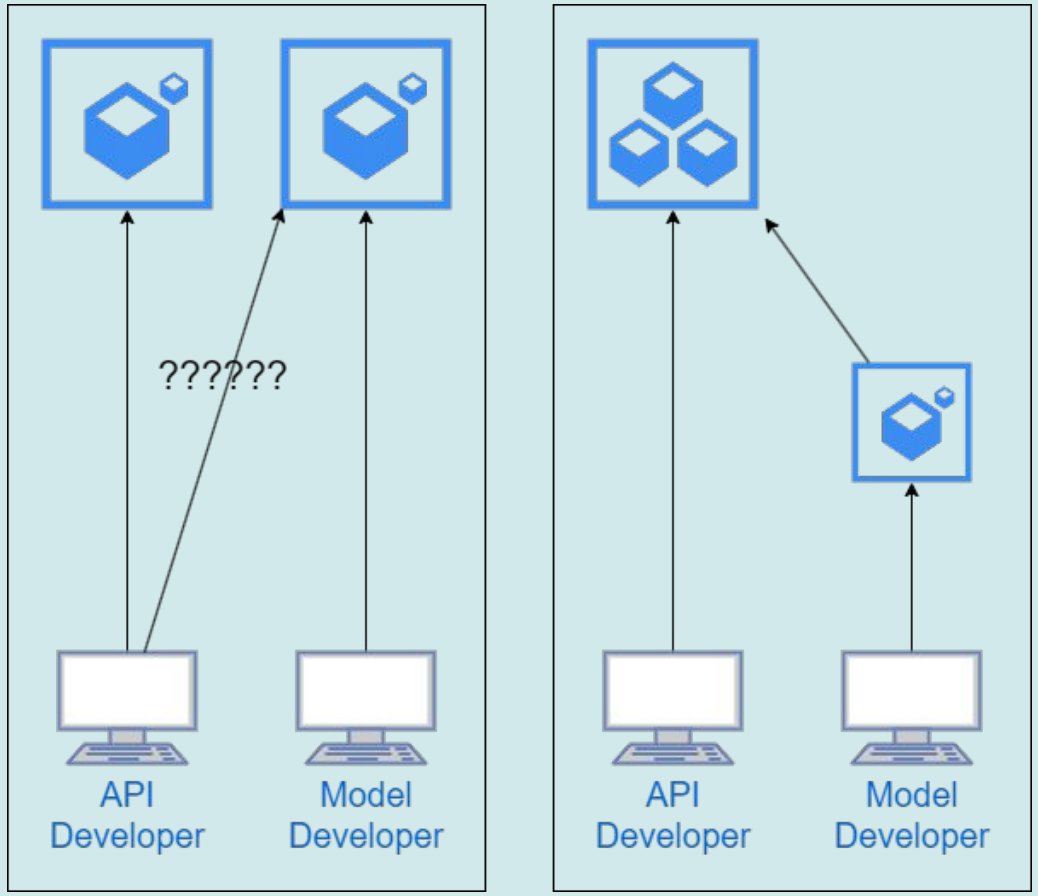
Multiple integration modes:

1. From model base template

Want to start your project from the base, following good practices?

You want are developer of the model source and the api. You want all to be highly coupled and in a unique same place.

> You can also paste your code here.



Multiple integration modes:

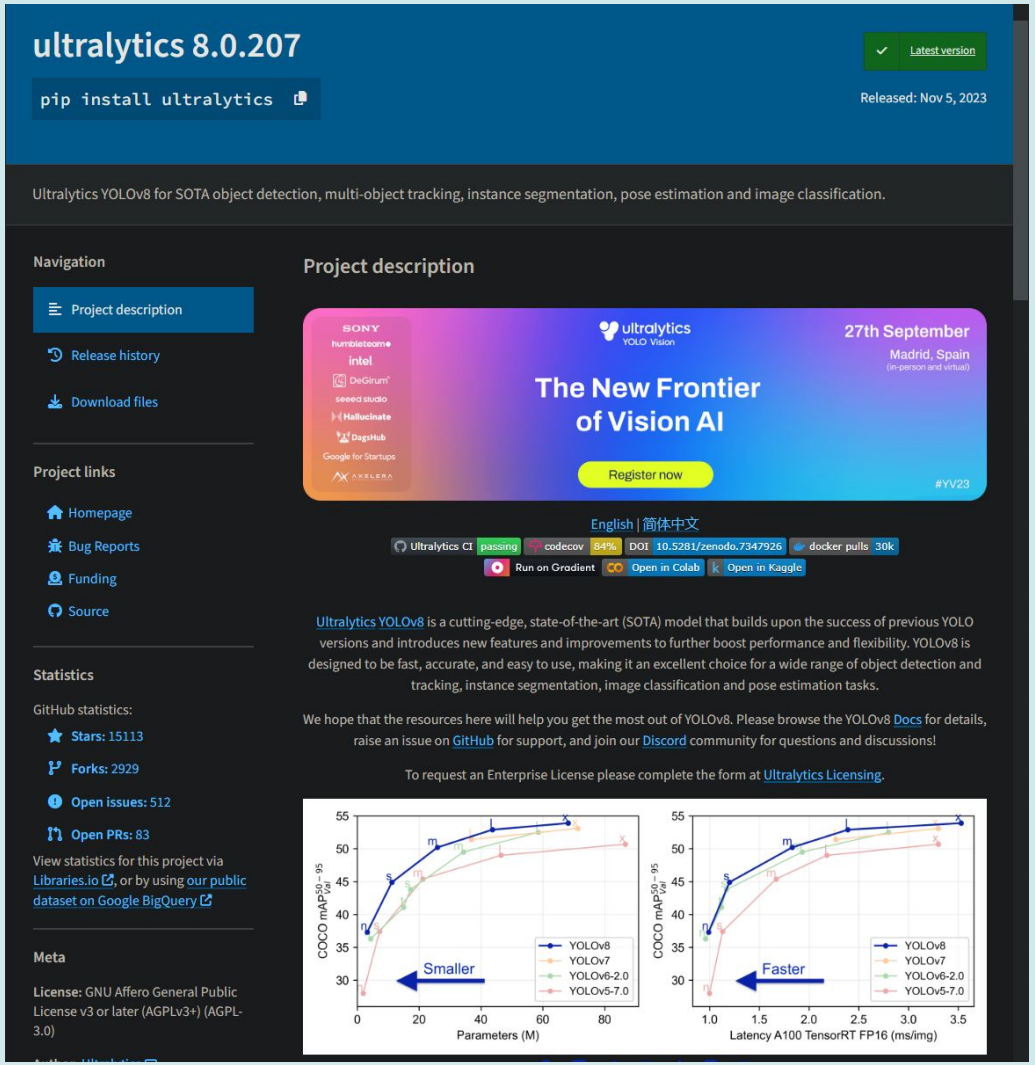
2. From repository as submodule

Your team have a source model and want to add the API?

You have are not the developer of the model, and it is not in PyPI.

> Splits maintenance of source.

> Model source in root folder.



Multiple integration modes:

3. As requirement for the API

You just want to create an API for a known or public model?

You are developing a service for a public model. You do not need/want the model source in your repository.

> Use the template without model src.

> Add the model to pip requirements.txt.

Additional code features:

1. Templates for responses (pdf.)
2. Schemas module for parameters

Inspire users on how to generate the response they need.

Object schemas simplify maintenance and visibility in APIs with large amount of arguments and hyperparameters.

Provide single points where user needs to modify the code.

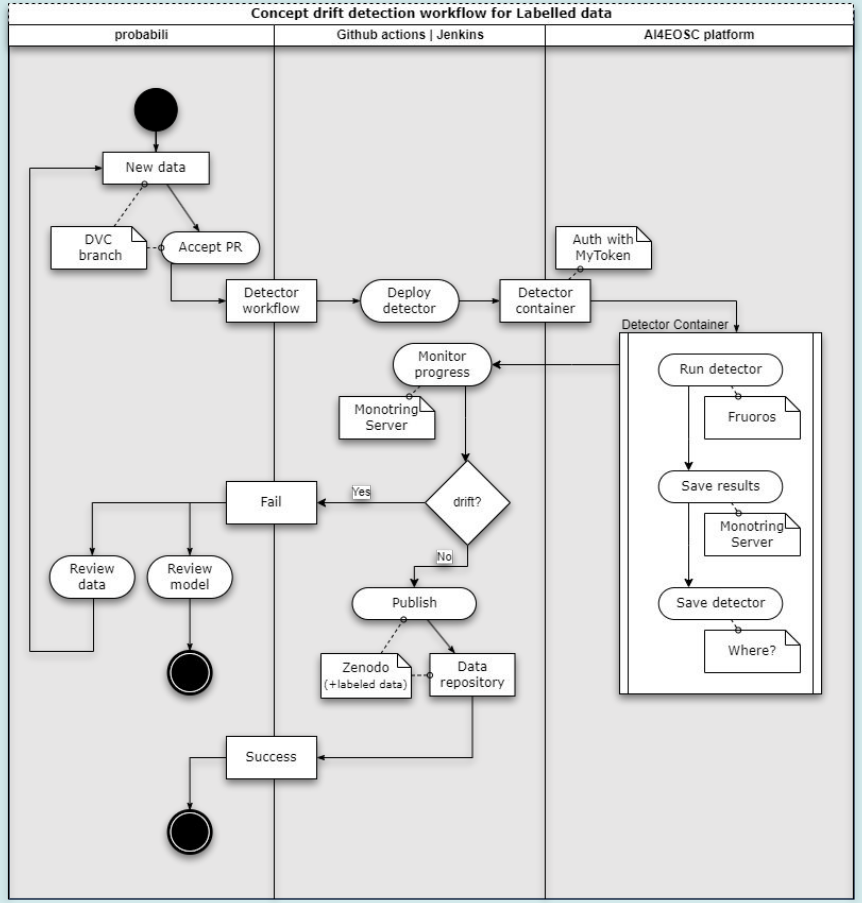
```python
# EXAMPLE of Prediction Args description
# = HAVE TO MODIFY FOR YOUR NEEDS =
You, 3 months ago | 1 author (You)
class PredArgsSchema(marshmallow.Schema):
    """Prediction arguments schema for api.predict function."""

    You, 4 months ago | 1 author (You)
    class Meta:  # Keep order of the parameters as they are defined.
        # pylint: disable=missing-class-docstring
        # pylint: disable=too-few-public-methods
        ordered = True

    model_name = ModelName(
        metadata={
            "description": "String/Path identification for models.",
        },
        required=True,
    )

    input_file = fields.Field(
        metadata={
            "description": "File with np.arrays for predictions.",
            "type": "file",
            "location": "form",
        },
        required=True,
    )

    accept = fields.String(
        metadata={
            "description": "Return format for method response.",
            "location": "headers",
        },
        required=True,
        validate=validate.OneOf(list(responses.content_types)),
    )

```

Additional code features:

3. Pre-made testing with pytest
4. Debug configurations for vscode

We know it is hard and long to generate comprehensible tests. Therefore, we can generate basic testing for you by simply testing the known points of the API in search for failures.

Additionally you can debug with breakpoints if you use vscode.



And more to come:

1. Jenkins SQA baseline v2

Simpler but powerful, add quality control to your CICD.
Get high score SQA badges with minimum effort.

2. Data version control (with DVC)

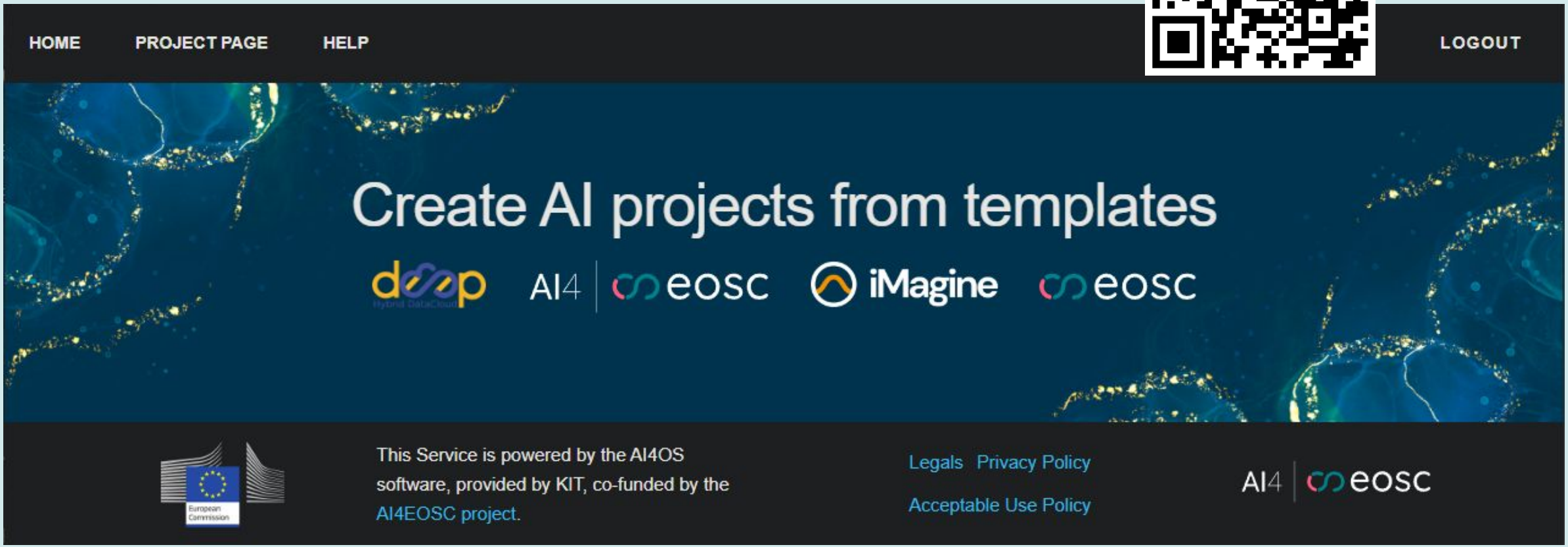
Add data version and control to your repository meanwhile it is stored in your favorite cloud solution.

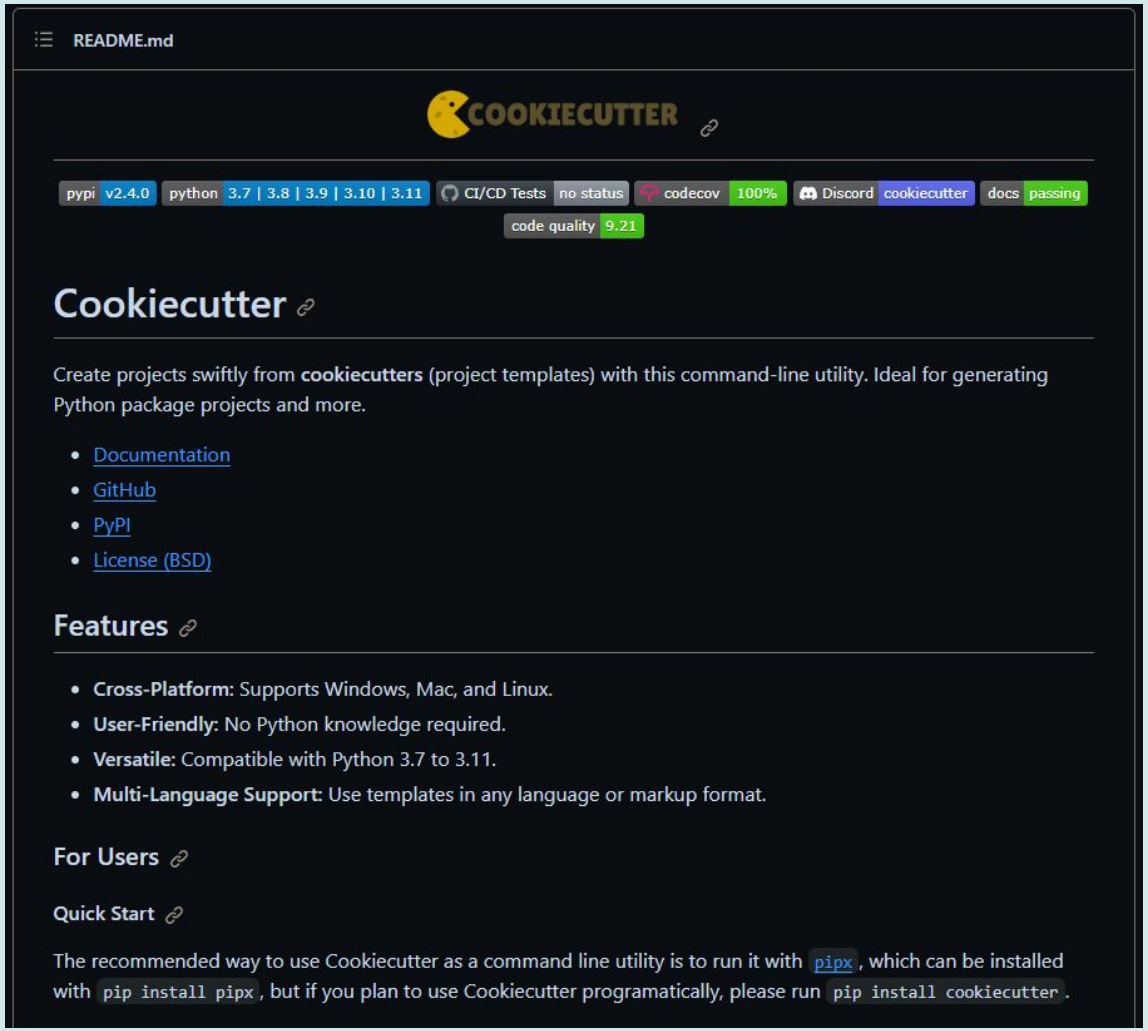
3. Drift detection with Frouros

Monitor and validate your new data before merging into “main” or publish it automatically.

How to start with templates?

templates.services.fedcloud.eu





Powered by Cookiecutter

Alternatively to using templates.services.fedcloud.eu , it is possible to use edge-versions directly with cookiecutter at:

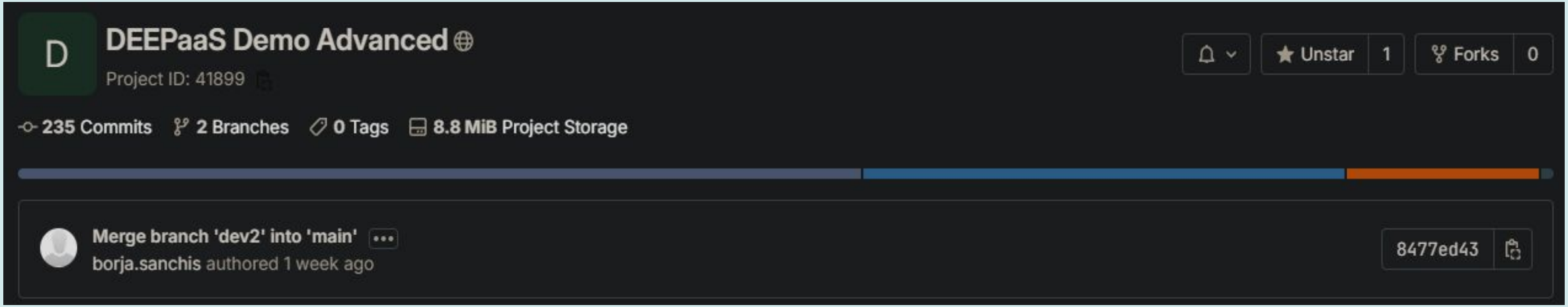
github.com/deephdc/cookiecutter-deep

For example:

> cookiecutter -c advanced \
https://github.com/deephdc/cookiecutter-deep

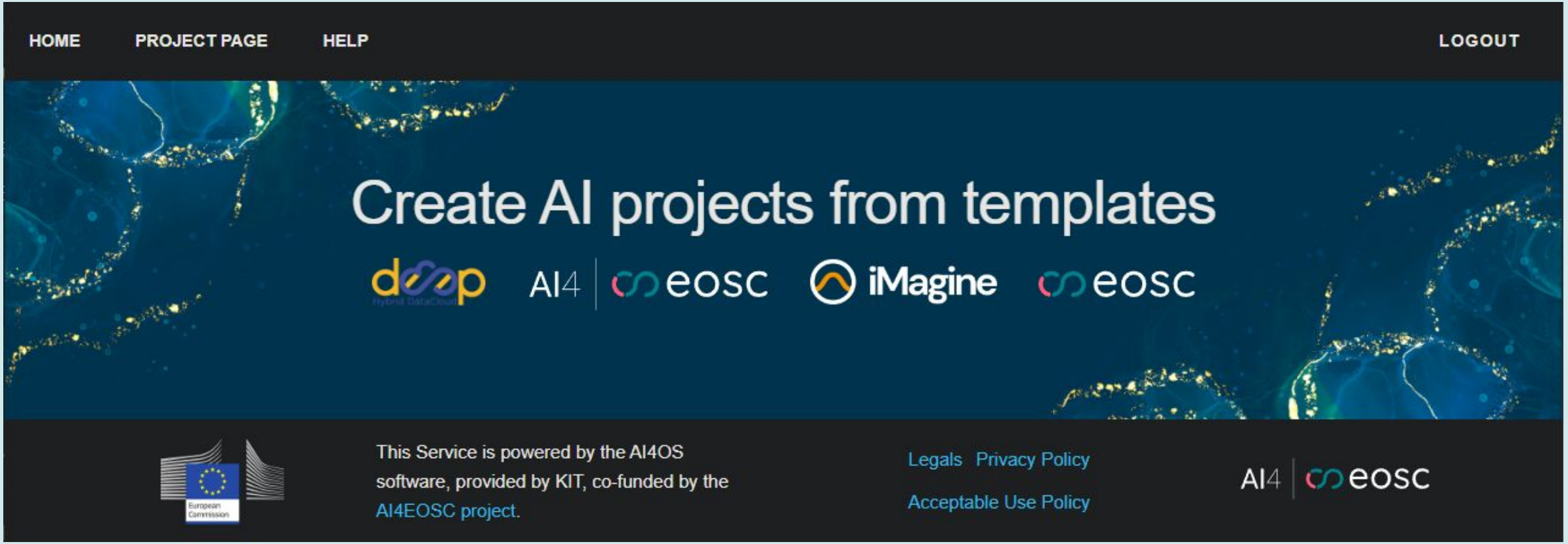
> cookiecutter -c v1.0.0\
https://github.com/deephdc/cookiecutter-deep

Time for a demo?

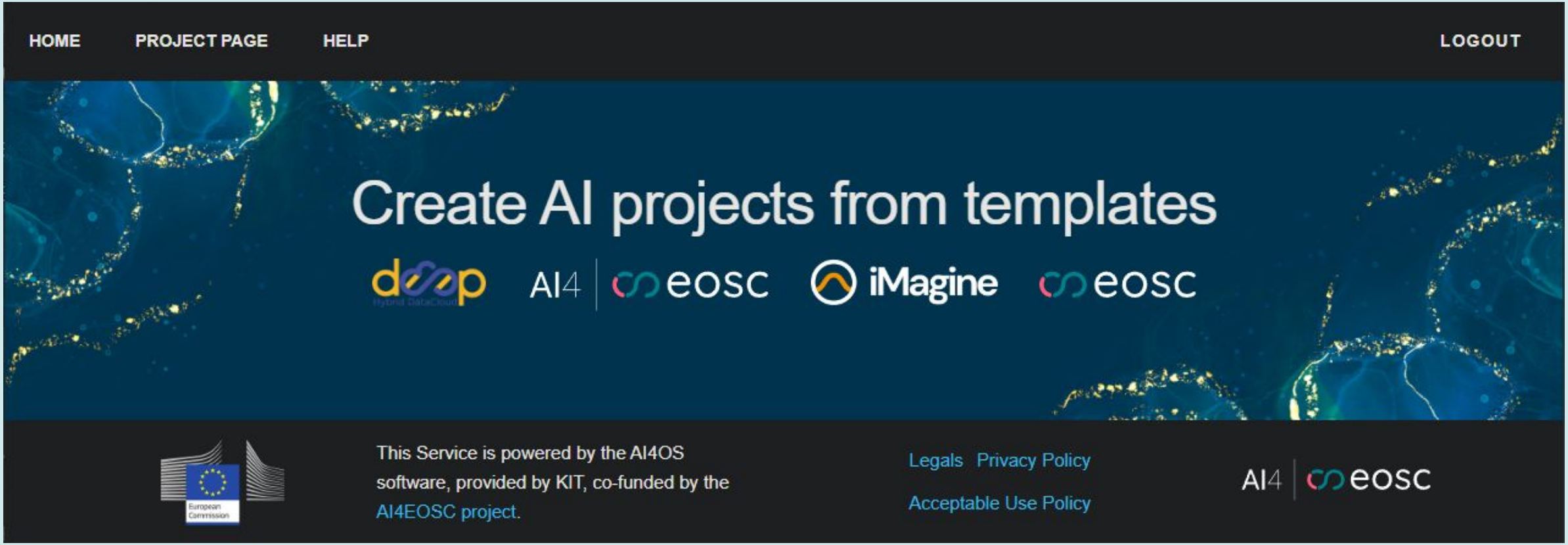


git.scc.kit.edu/m-team/ai/demo-advanced-api

Time for questions



Thank you for your time!



FAQ

- Flask returns debug info in web page!
  Yes, but for frontend debug purposes, DEEPaaS API is not a frontend framework (Yet).