Bratislava Workshop – 15/11/2023

AI4 | eosc

Deploy your AI-based service for inference using OSCAR

Amanda Calatrava

Universitat Politècnica de València (UPV)

Co-funded by the European Union

Life cycle of an AI service

Step 1

Build /develop AI service

Step 2

Train the model in the AI4EOSC platform

Step 3

Deploy your AI-based service for inference

OSCAR: an overview

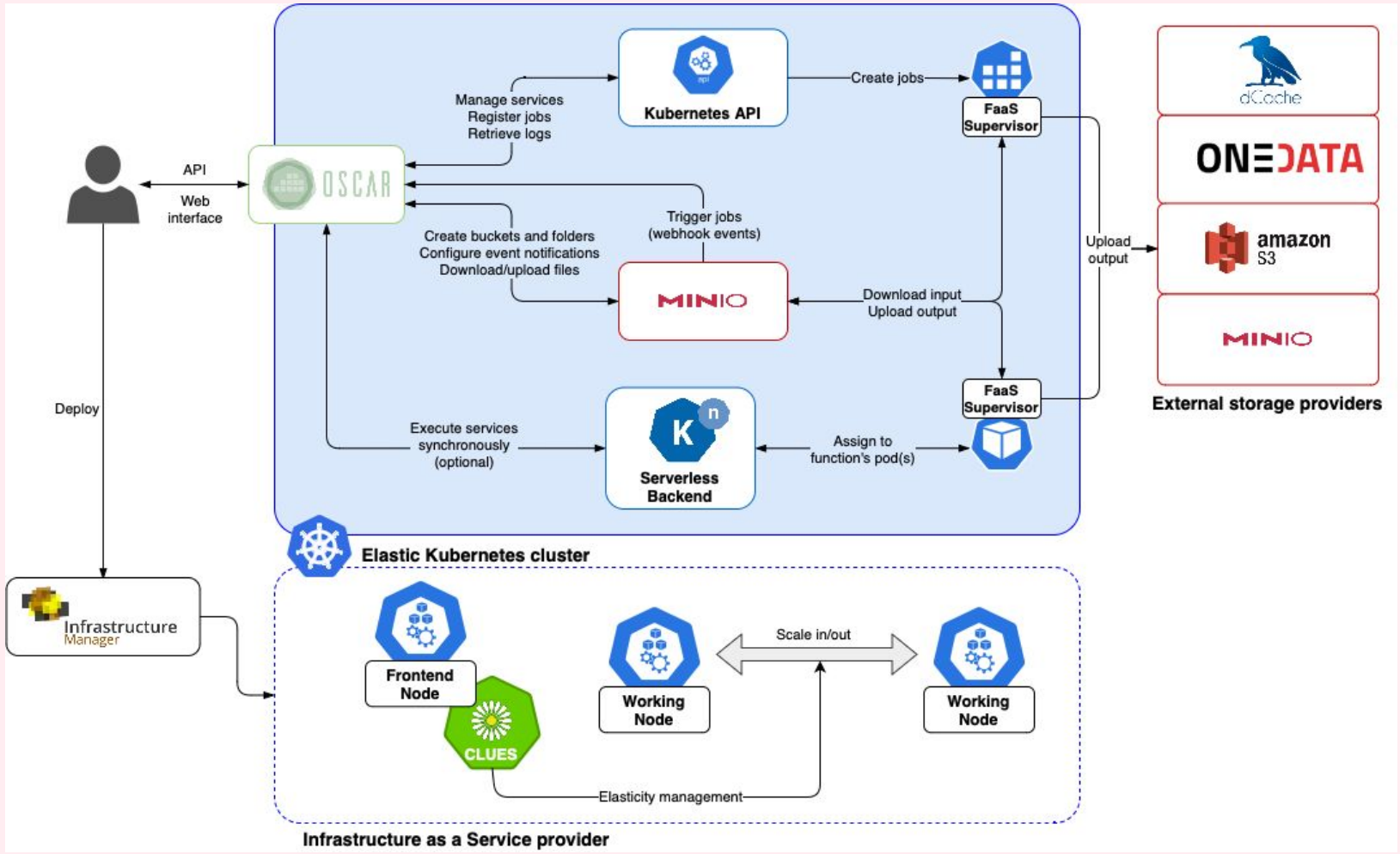
OSCAR is an open-source platform that:

- Support the **serverless** (event-driven + automated elasticity) computing model for data-processing applications on Kubernetes.
- Is easily deployed on **multi-Clouds** via the Infrastructure Manager (IM).
- Supports ARM-based low-consuming devices such as Raspberry PIs, enabling Serverless computing at **the Edge** and composing workflows along the computing continuum.
- Integrates with SCAR (runs Docker on AWS Lambda + AWS Batch) to create data-processing workflows along the **computing continuum**.

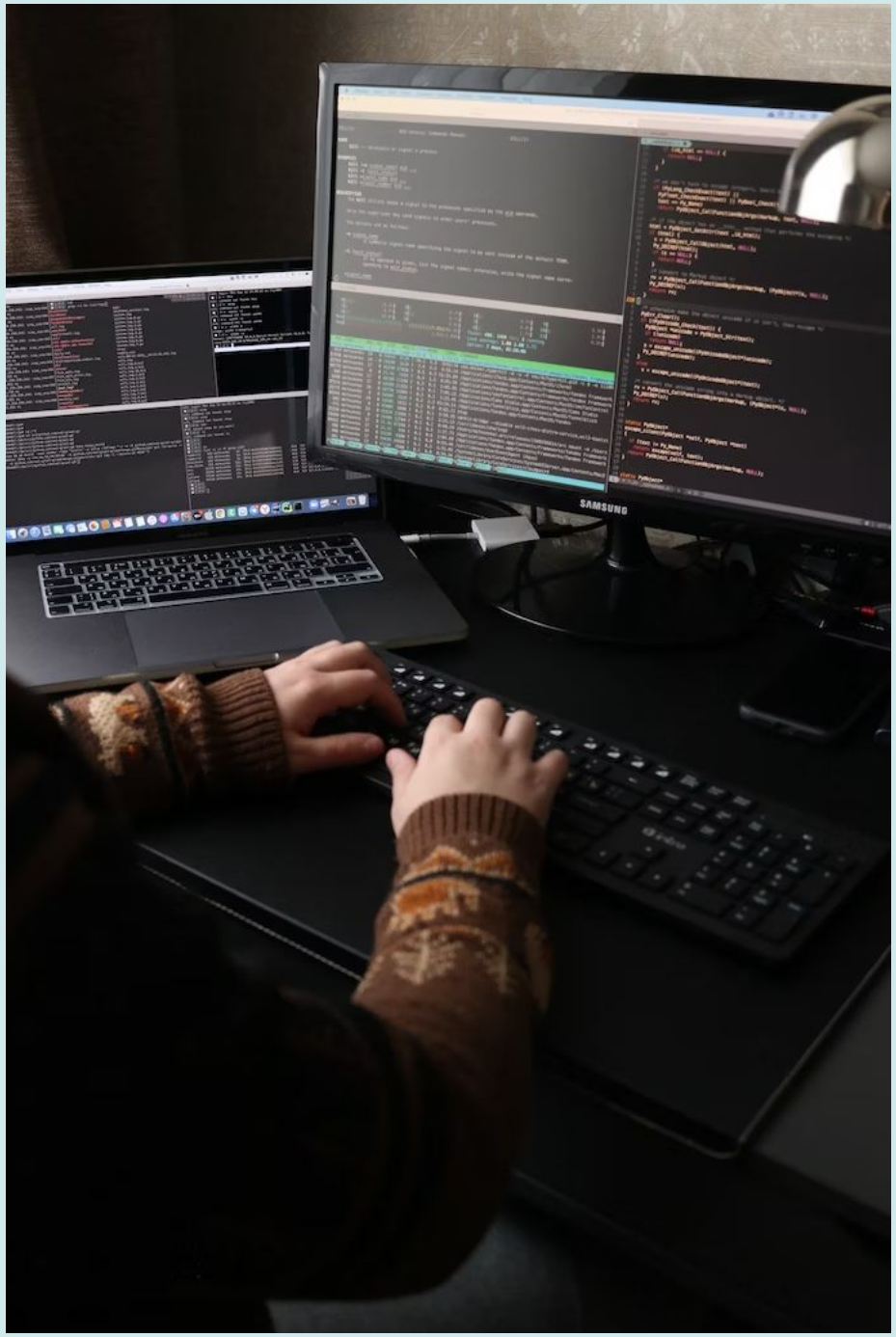
https://oscar.grycap.net

https://github.com/grycap/scar

OSCAR: architecture



- Supports the execution of highly-parallel event-driven data-processing serverless applications
- Integrated with:
  - IM Dashboard
  - EGI DataHub
  - EGI Check-In
  - dCache (via WebDav)



AI4 | eosc

Co-funded by the European Union

Try it yourself!

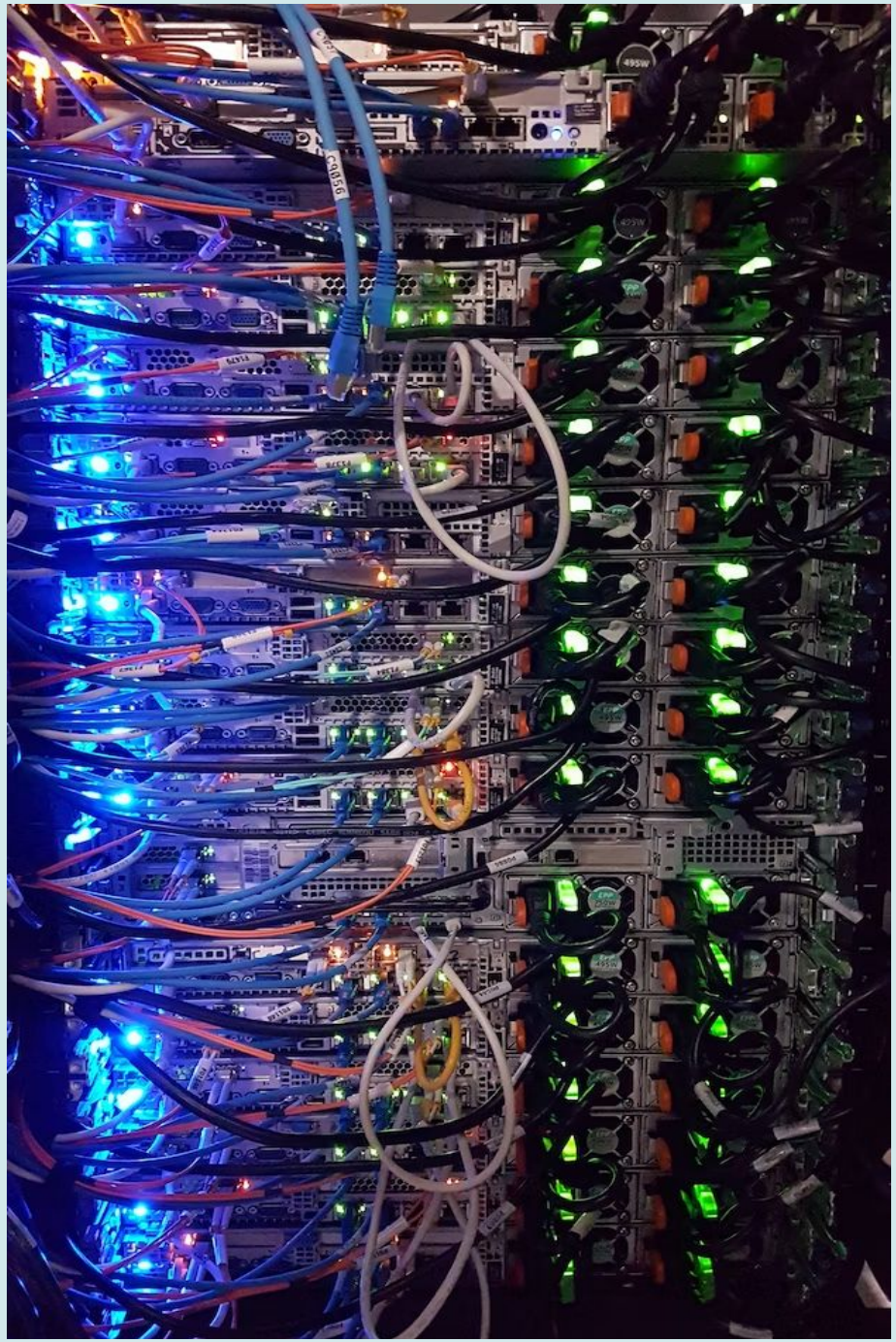
Deploy a local OSCAR cluster

- **Local deployment** (documentation):
  - **Dependencies:** Go, Docker, kind

```
$ curl -sSL http://go.oscar.grycap.net | bash
```

```
Do you want to use Knative Serving as Serverless Backend? [y/n] y
Do you want suport for local docker images? [y/n] y
```

```
[✓] OSCAR platform deployed correctly

 > You can now acces to the OSCAR web interface through https://localhost with the
following credentials:
  - username: oscar
  - password: [redacted]

 > You can now access to MinIO console through http://localhost:30300 with the
following credentials:
  - username: minio
  - password: [redacted]

[*] Note: To delete the cluster type 'kind delete cluster --name=oscar-test'
```

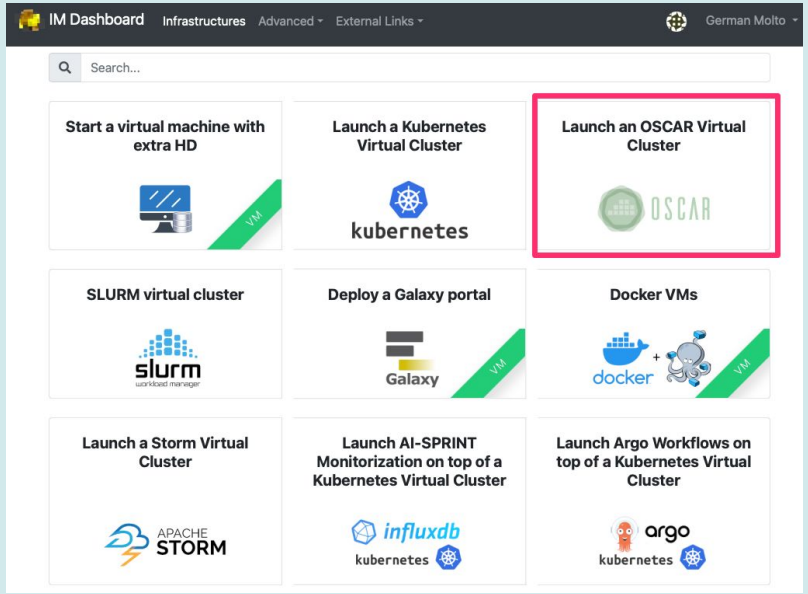


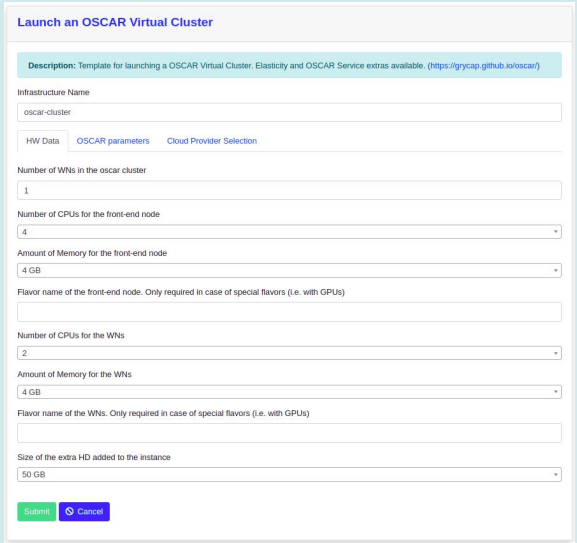
AI4 | eosc Co-funded by the European Union

Try it yourself!

Deploy a virtual OSCAR cluster

- **Automated Deployment**: through IM Dashboard (docs)
  - IM Dashboard allows the easy deployment of OSCAR Clusters in multi-Clouds, including the EGI Federated Cloud





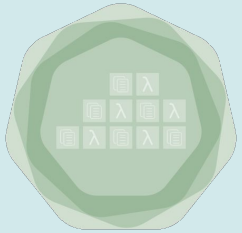
https://im.egi.eu

AI4 | eosc Co-funded by the European Union

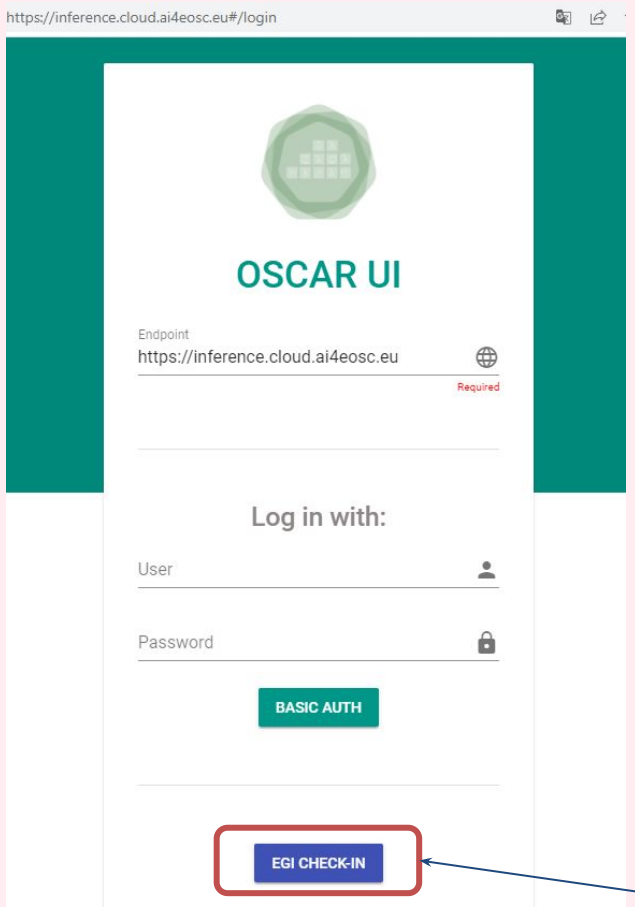
Try it yourself!

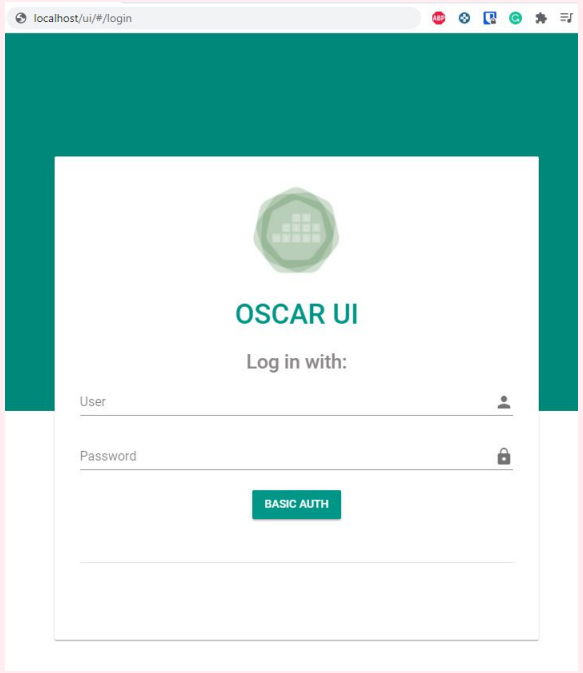
Access to an already deployed OSCAR cluster

- **Access to the AI4EOSC OSCAR cluster**
  - Go to https://inference.cloud.ai4eosc.eu and use EGI Check-in to authenticate and access the resource.
  - You need to be part of the AI4EOSC VO `vo.ai4eosc.eu` (if you're part of the project, you can request membership here) or the iMagine VO (vo.imagine-ai.eu)
  - Internal Documentation (in Confluence).
  - AI4OS Documentation.

 + AI4 | eosc

OSCAR UI





(*) https://localhost

You need to be part of the AI4EOSC VO

(**) https://inference.cloud.ai4eosc.eu

- Graphical web-based interface
- Easy to use
- User-friendly design
- Synchronous and asynchronous jobs
- Ready to use after deploying the local cluster (*)
- Integrated with EGI Check-in(**)

Documentation:
https://docs.oscar.grycap.net/usage/

OSCAR: services

```
functions:
  oscar:
  - oscar-test:
      name: grayify
      memory: 1Gi
      cpu: '1.0'
      image: grycap/imagemagick
      script: grayify.sh
      input:
      - storage_provider: minio.default
        path: gray/input
      output:
      - storage_provider: minio.default
        path: gray/output
      - storage_provider: webdav.dcache
        path: gray/output
storage_providers:
  webdav:
    dcache:
      hostname: my_dcache.com
      login: my_username
      password: my_password
```

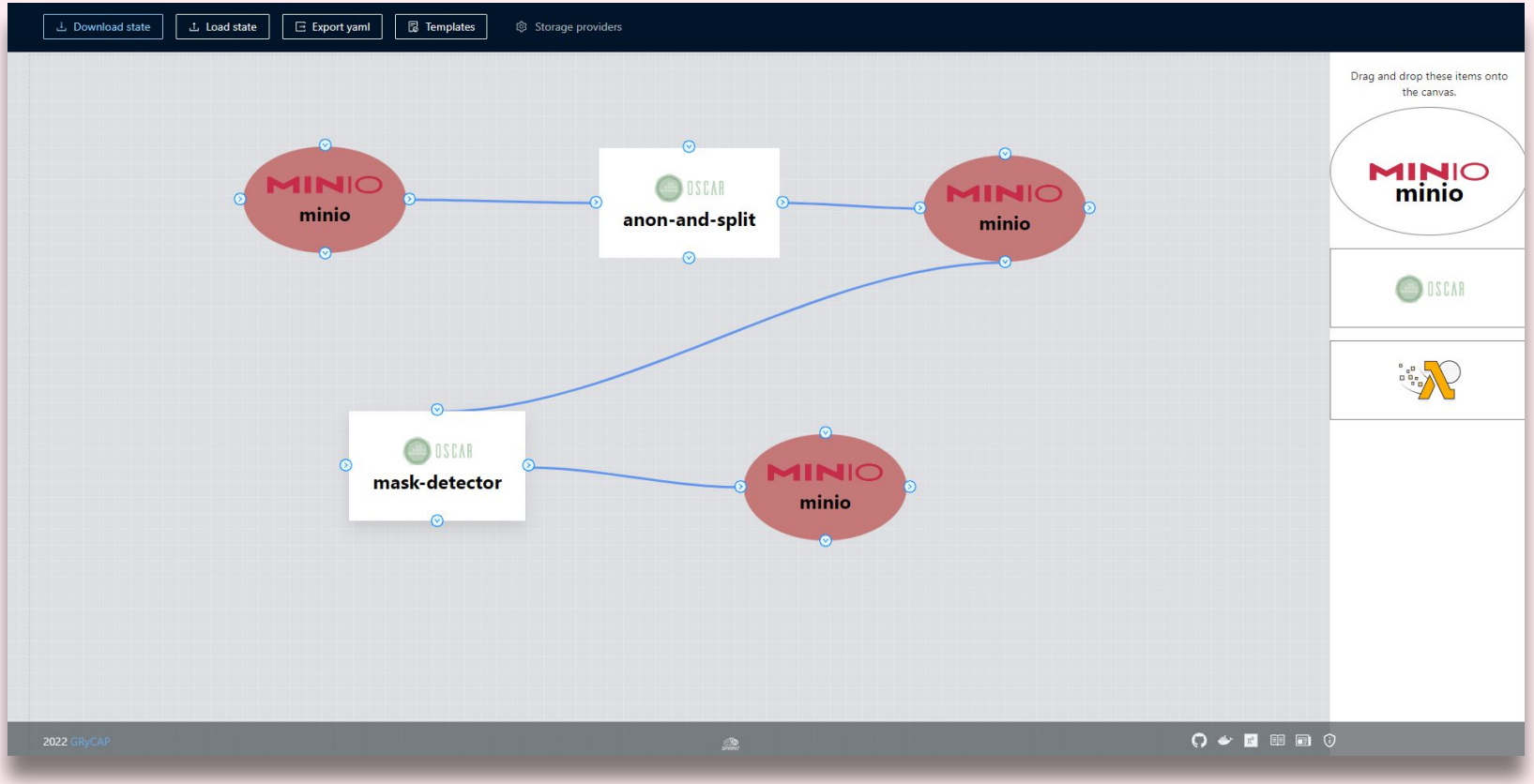
OSCAR Functions Definition Language (FDL)

OSCAR service basic components

- Cluster identification
- Memory and CPU requirements of the service
- Docker image with your runtime environment
- Script that will be executed on a provisioned Docker container
- (Optional) Storage input/output MinIO buckets to trigger executions and store results
- *Additional storage providers (optional)*

Documentation:
https://docs.oscar.grycap.net/fdl/

OSCAR: FDL Composer



https://composer.oscar.grycap.net/

- Visually design workflows for OSCAR.
- Graphical composition approach.
- Generate FDLs from the pipeline created.

Documentation: https://oscar.grycap.net/blog/post-oscar-fdl-composer/

OSCAR CLI

More powerful client. You can perform operations like:

- Apply a FDL file to create or edit a service
- Invoke a service synchronously
- Add an existing cluster to oscar-cli
- Usage with EGI Check-in:
  - OSCAR cluster with VO support and oidc-agent installed.
  - Use command oscar-cli cluster add with the flag –-oidc-account-name
- Needs to be installed by itself (not included in the local deployment)

Documentation: https://docs.oscar.grycap.net/oscar-cli/

Apply a FDL file to create or edit services in clusters.

```
Usage:
  oscar-cli apply FDL_FILE [flags]

Aliases:
  apply, a

Flags:
      --config string   set the location of the config file (YAML or JSON)
  -h, --help            help for apply
```

```
Usage:
  oscar-cli cluster add IDENTIFIER ENDPOINT {USERNAME {PASSWORD | \
  --password-stdin} | --oidc-account-name ACCOUNT} [flags]

Aliases:
  add, a

Flags:
      --disable-ssl              disable verification of ssl certificates for the
                                 added cluster
  -h, --help                     help for add
  -o, --oidc-account-name string OIDC account name to authenticate using
                                 oidc-agent. Note that oidc-agent must be
                                 started and properly configured
                                 (See:https://indigo-dc.gitbook.io/oidc-agent/)
      --password-stdin           take the password from stdin

Global Flags:
      --config string   set the location of the config file (YAML or JSON)
```

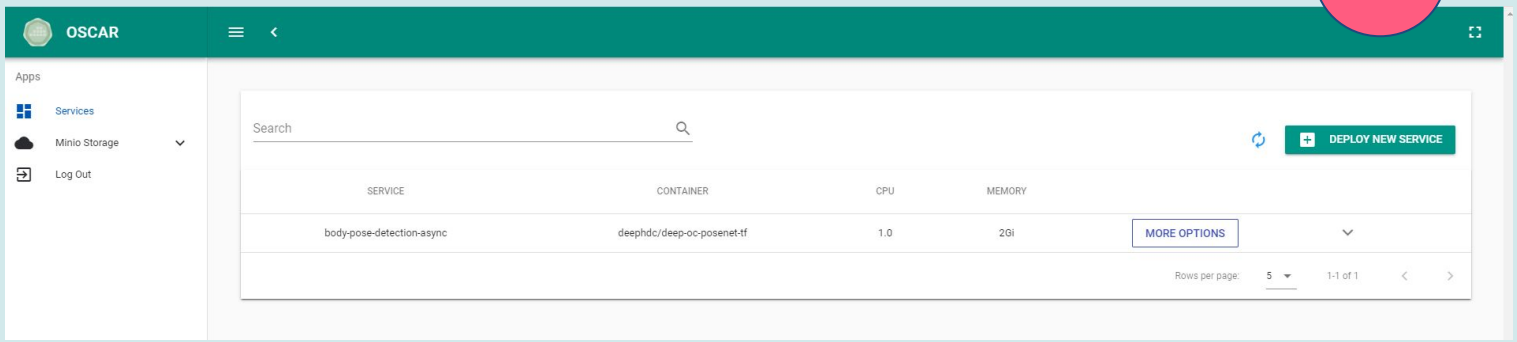
AI4 | eosc

Co-funded by the European Union

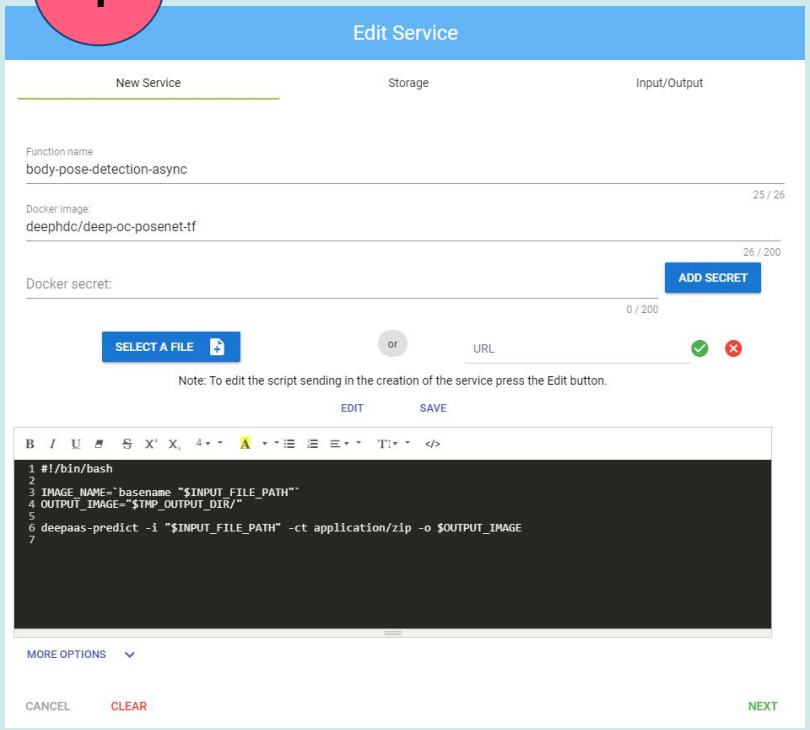
Try it yourself!

Creating a service with OSCAR-UI

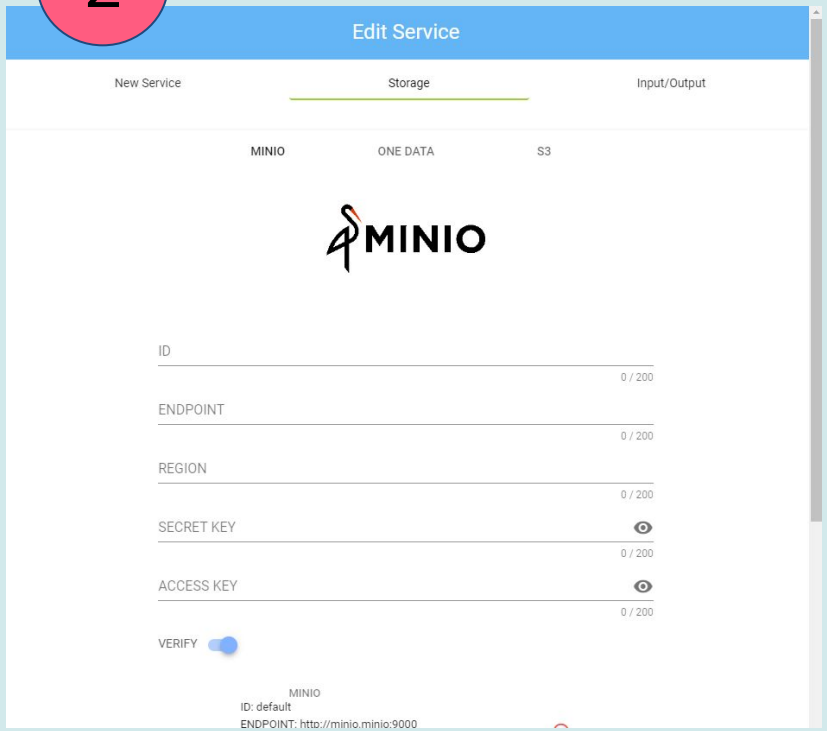
4



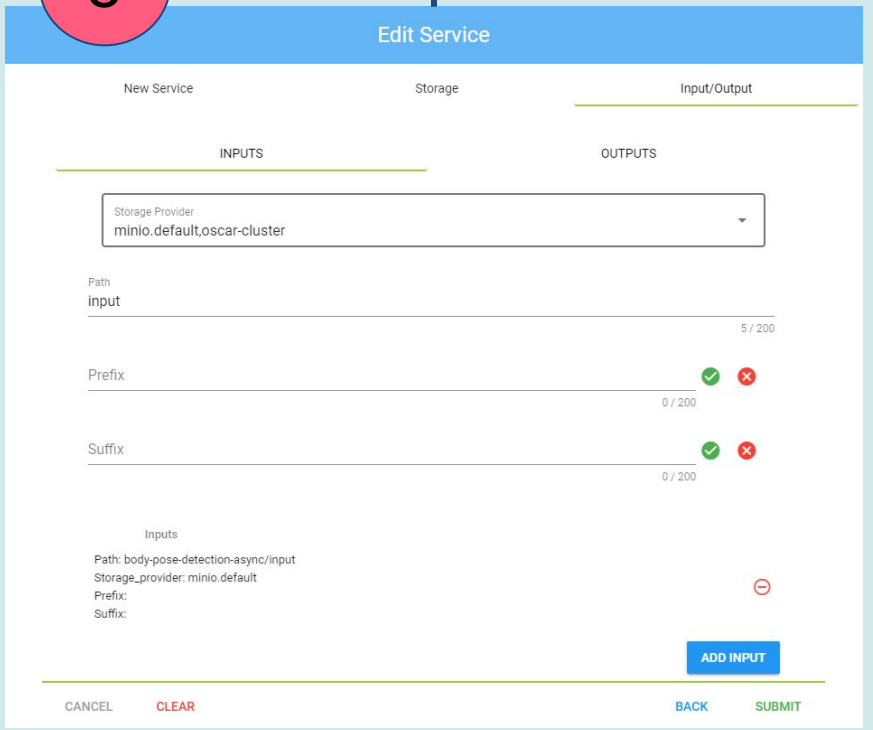
1



2



3



https://inference.cloud.ai4eosc.eu

AI4 | eosc Co-funded by the European Union

Try it yourself!

Creating a service with OSCAR-CLI

0 Configure `oidc-agent` and register an **EGI client**.

1 Add the cluster to **OSCAR-CLI**:

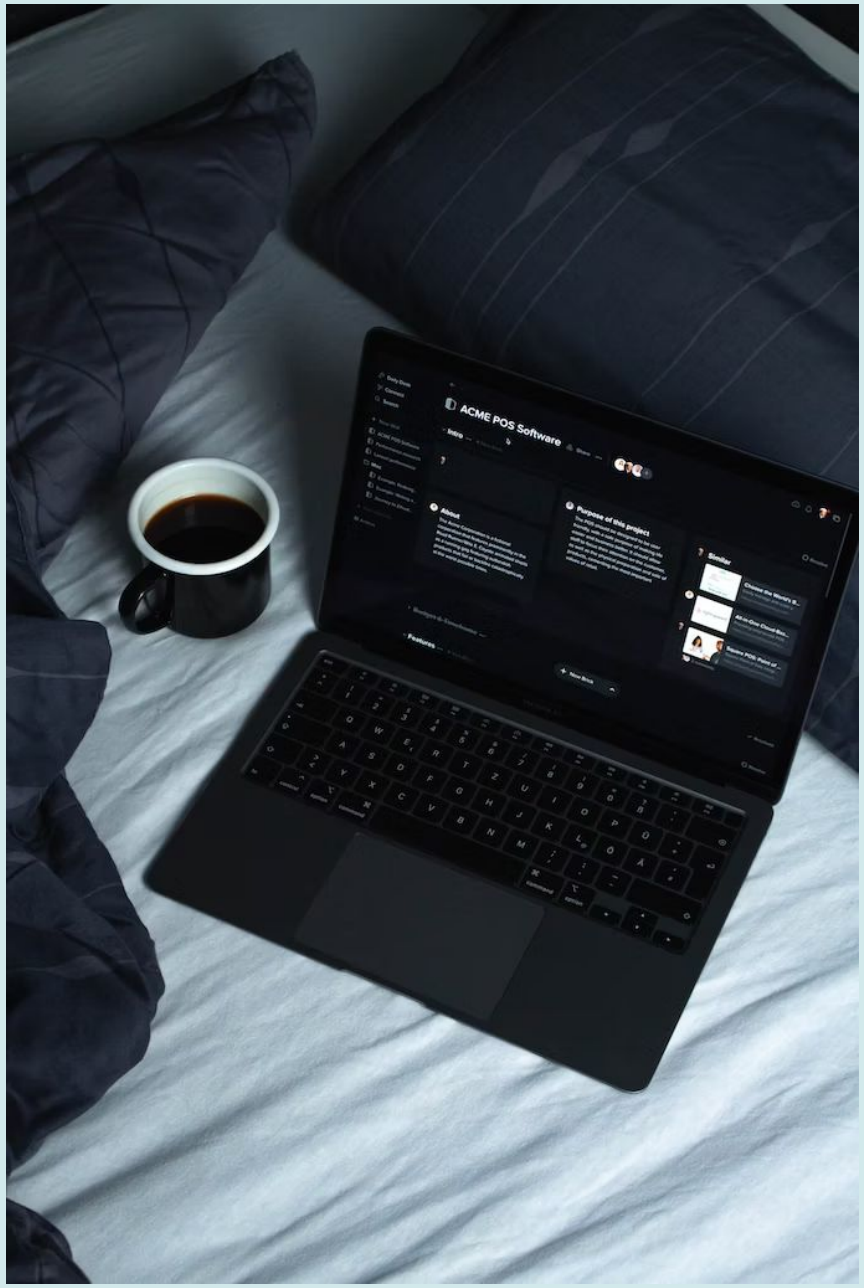
```
$ oscar-cli cluster add oscar-cluster
https://localhost oscar <pass> --disable-ssl
```

2 Create a service by using the FDL:

- We can use as an example the "body-pose-detection" from the DEEP Catalog.

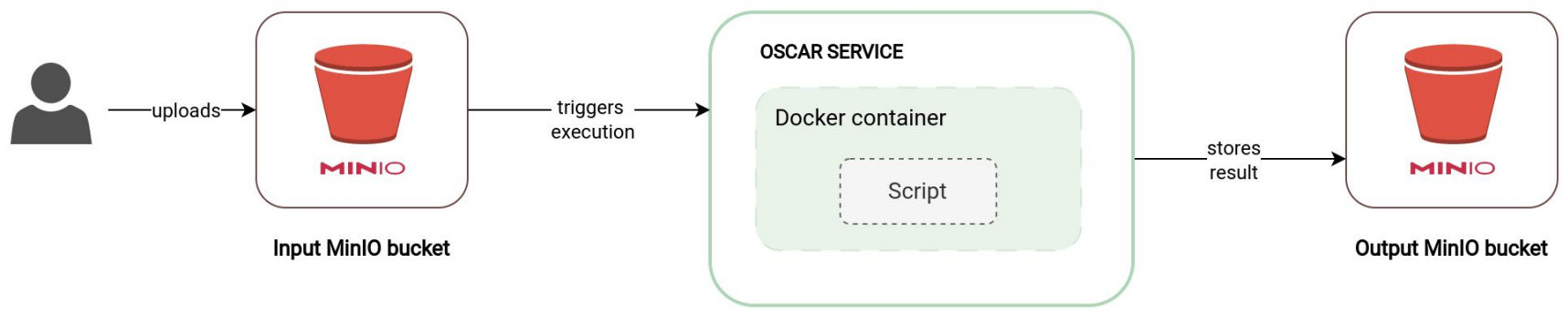
```
$ git clone https://github.com/grycap/oscar.git
```

```
$ oscar-cli apply body-pose-detection-async.yaml
```



OSCAR: async executions

- Programmable **object-storage** systems → **asynchronous** executions
- **Files uploaded** to the object-store **trigger the invocation** of a data-processing script that is run inside a container (out of user-defined Docker image) within a scalable Kubernetes cluster (i.e. batch jobs)
- Files uploaded to the object-store trigger a service and deposit the result in other object-store.
- The result can trigger the next service in our workflow.
- Useful for the execution of compute-intensive processing (job execution delays ~ few seconds)

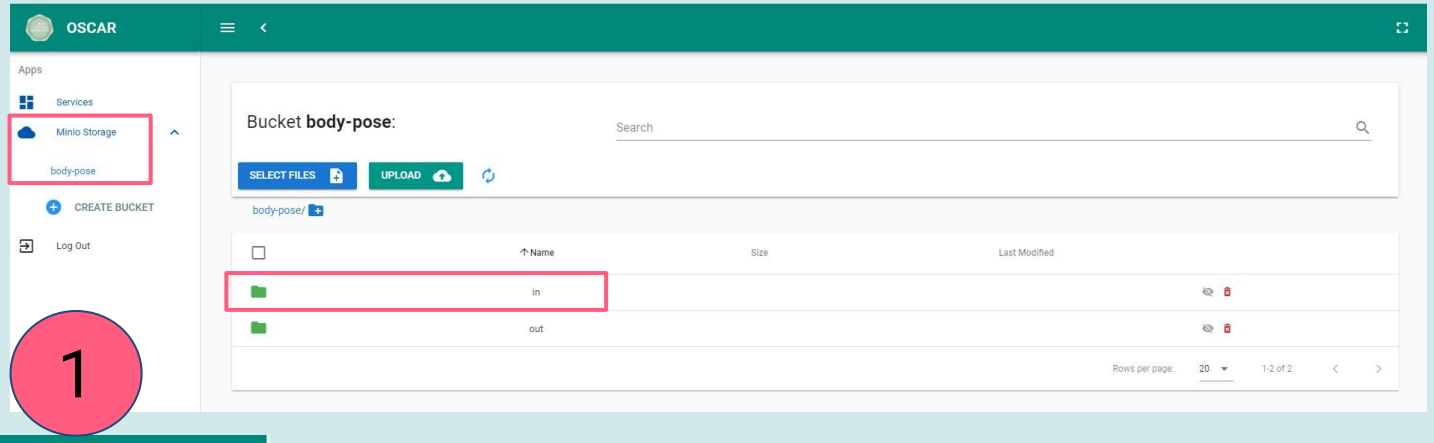


AI4 | eosc

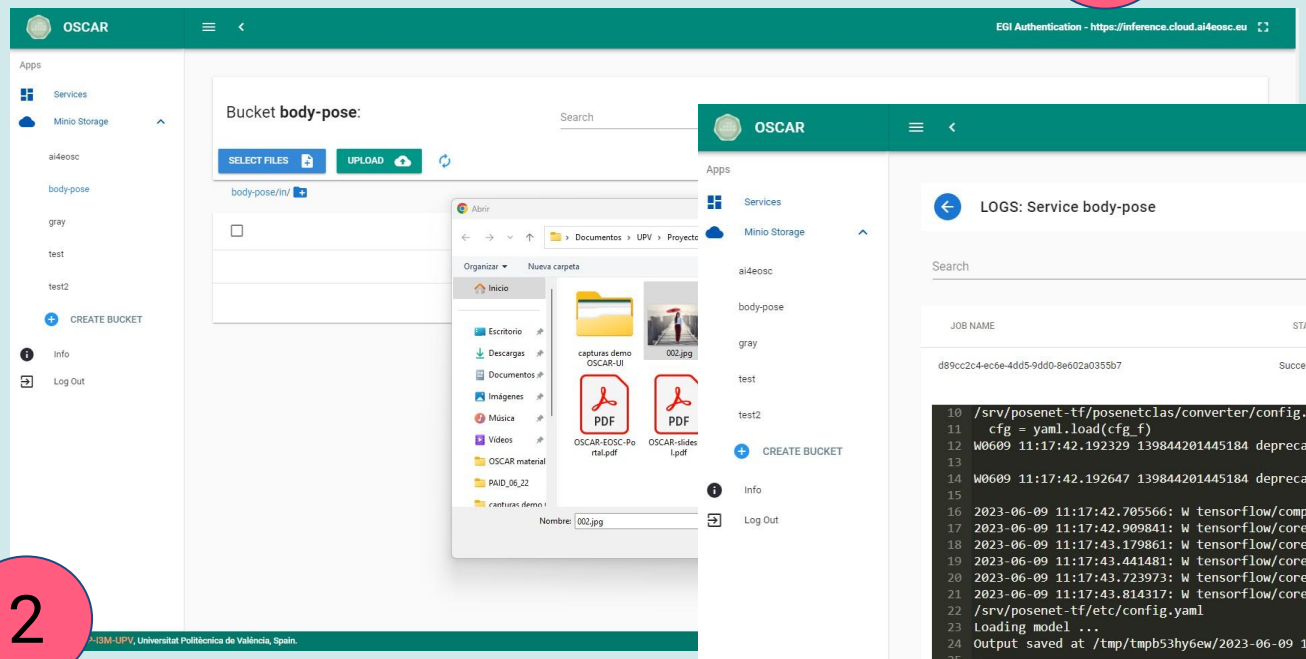
Co-funded by the European Union

Try it yourself!

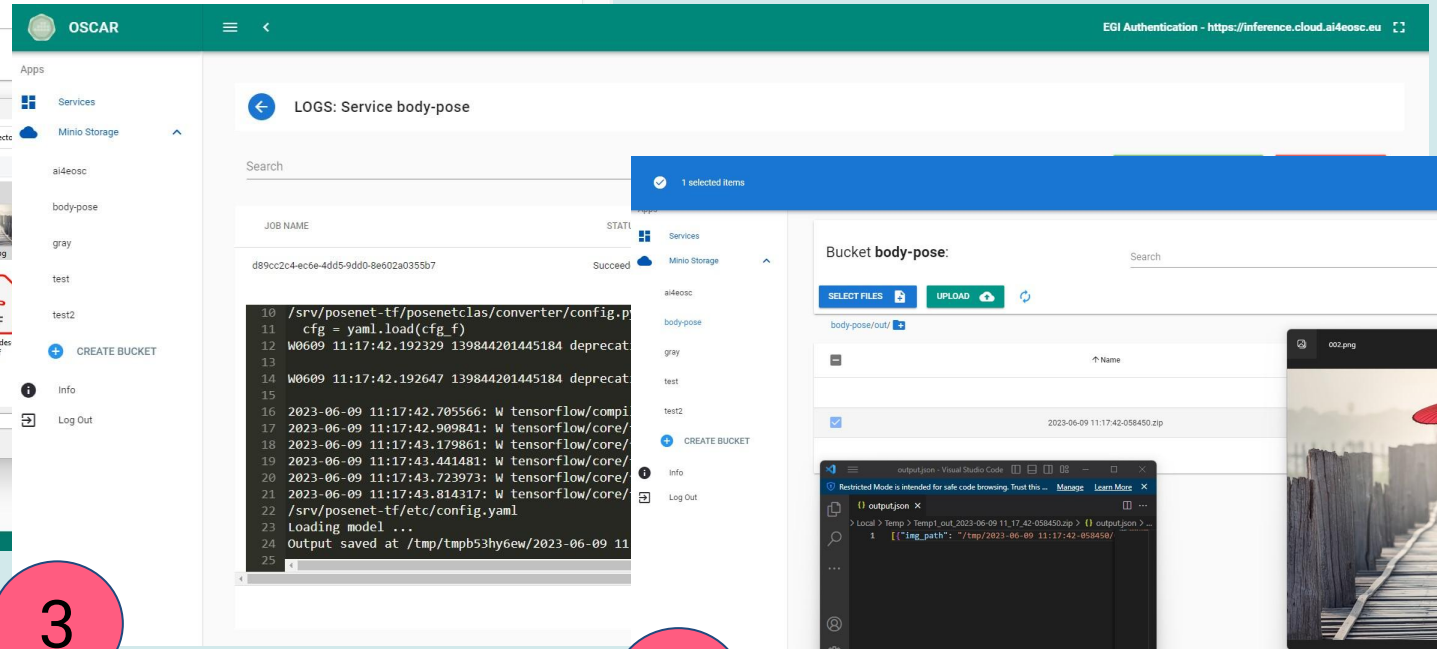
Running an async job with OSCAR UI



1



2



3

4

Hint! Test the "`body-pose-detection`" service in https://inference.cloud.ai4eosc.eu

AI4 | eosc

Co-funded by the European Union

Try it yourself!

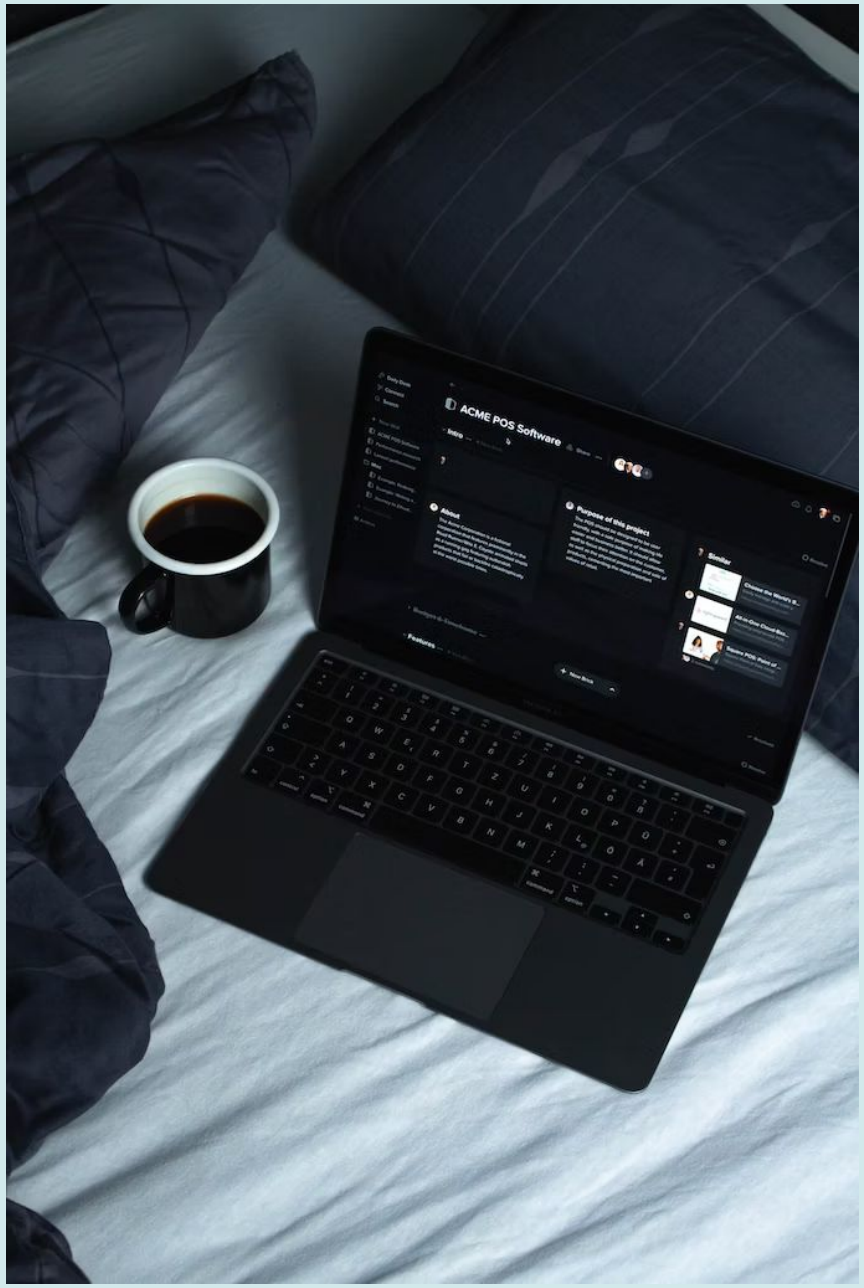
Running an async job with OSCAR-CLI

1

Once the service is created you can make the invocation with the following command, which will store the output on a minio bucket:

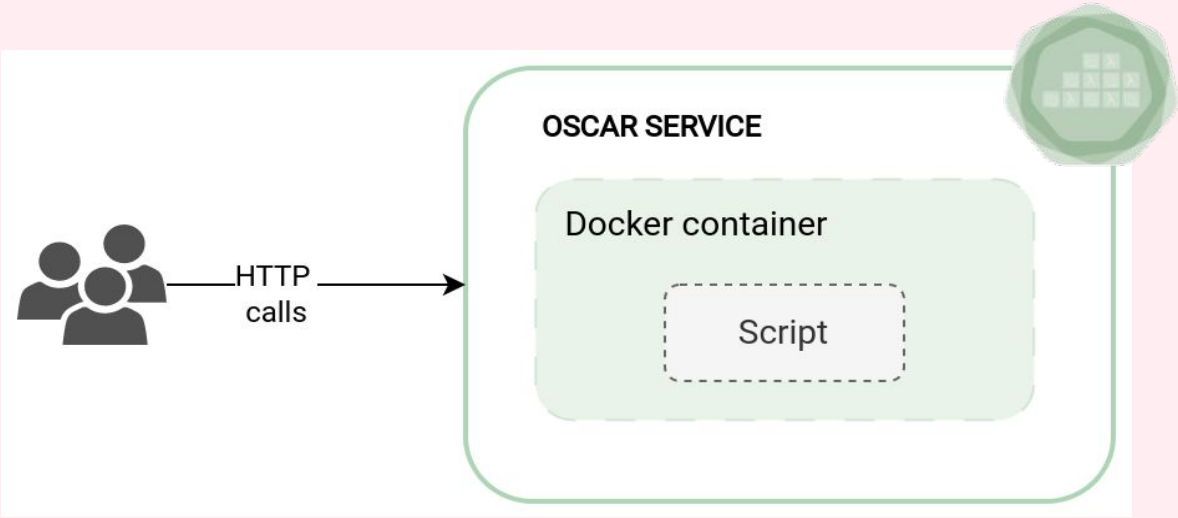
```
$ oscar-cli service put-file body-pose minio
images/001.jpg body-pose/input/001.jpg
```

```
Usage:
  oscar-cli service put-file SERVICE_NAME STORAGE_PROVIDER LOCAL_FILE \
   REMOTE_FILE [flags]

Aliases:
  put-file, pf

Flags:
  -c, --cluster string   set the cluster
  -h, --help             help for put-file

Global Flags:
      --config string   set the location of the config file (YAML or JSON)
```



OSCAR: sync executions

- Scalable **HTTP-based** endpoints (based on **KNative**) → **synchronous executions**
- Knative can maintain active a pool of execution workers per-service (user-defined via min_scale) to achieve fast response time.
- Useful for the execution of short-lived processing (job execution delays ~ less than a second)

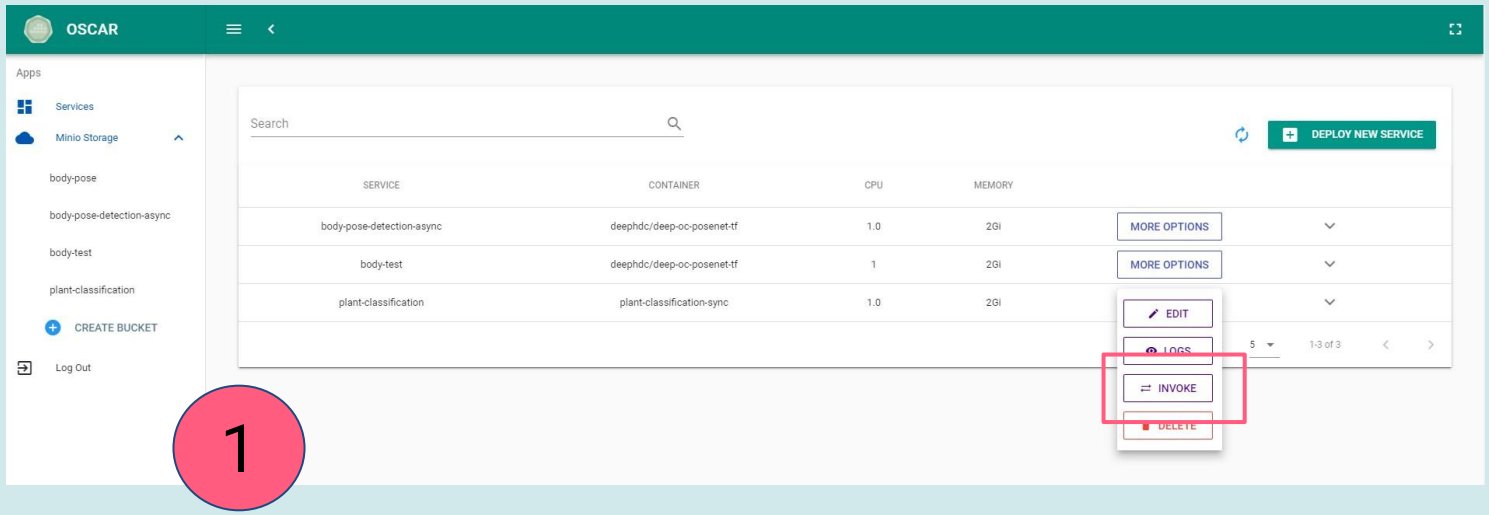


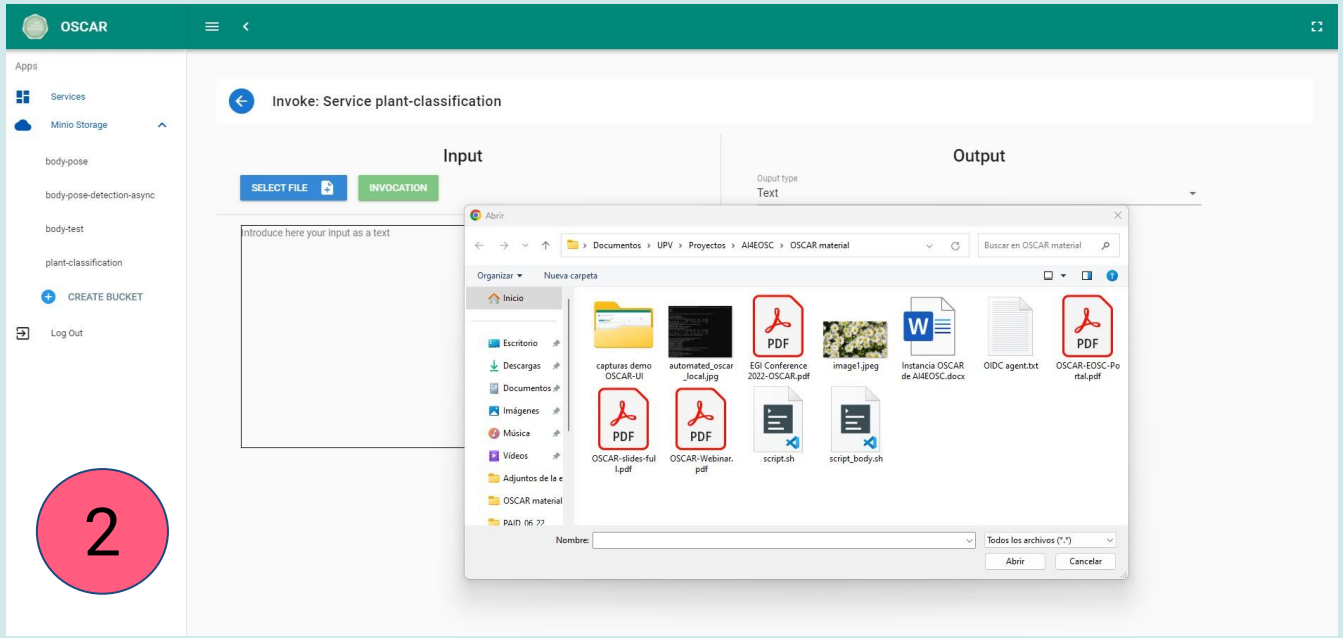
AI4 | eosc

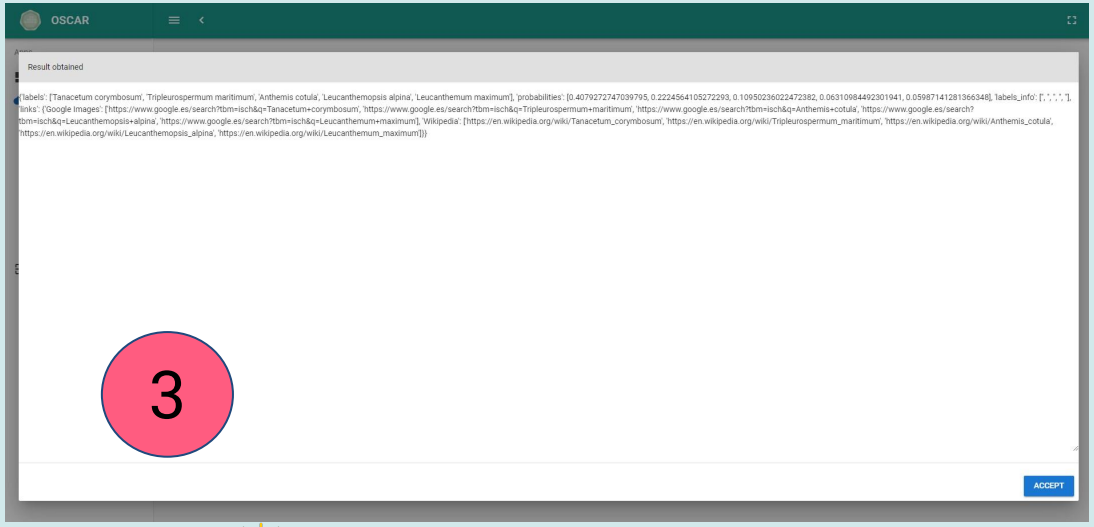
Co-funded by the European Union

Try it yourself!

Running a sync job with OSCAR UI

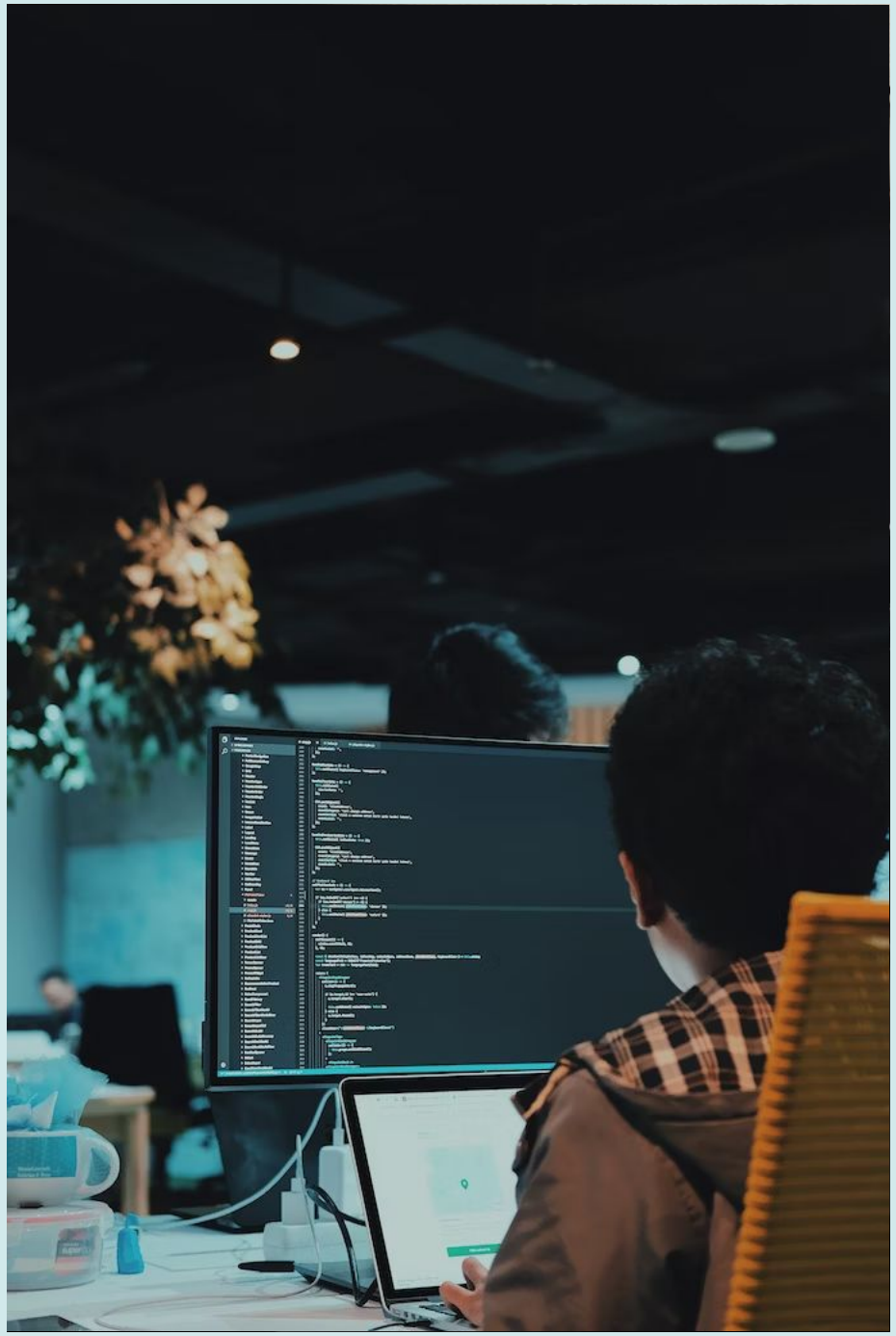






https://inference.cloud.ai4eosc.eu

Hint! Test the "`plants-classification`" service in https://inference.cloud.ai4eosc.eu



AI4 | eosc

Co-funded by the European Union

Try it yourself!

Running a sync job with OSCAR CLI

1

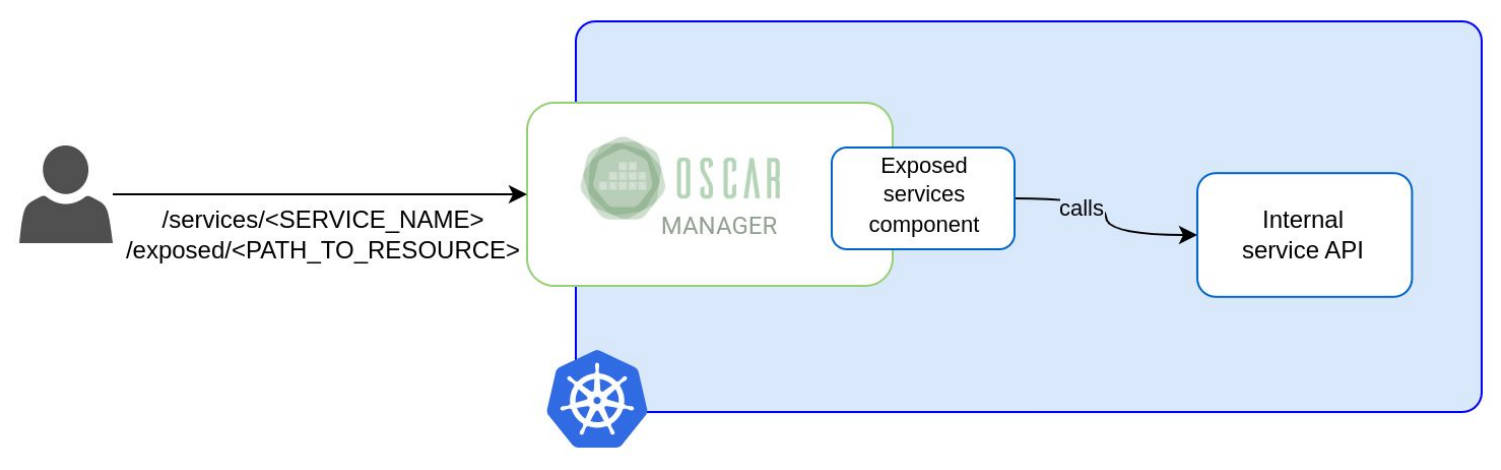
Once the service is created you can invoke it with the following command, which will trigger the execution of the service:

```
$ oscar-cli service run plant-classification
-i image.jpg -c oscar-ai4eosc-oidc
```

```
Usage:
  oscar-cli service run SERVICE_NAME {--input | --text-input} [flags]

Aliases:
  run, invoke, r

Flags:
  -c, --cluster string      set the cluster
  -h, --help                help for run
  -i, --input string        input file for the request
  -o, --output string       file path to store the output
  -t, --text-input string   text input string for the request

Global Flags:
      --config string   set the location of the config file (YAML or JSON)
```

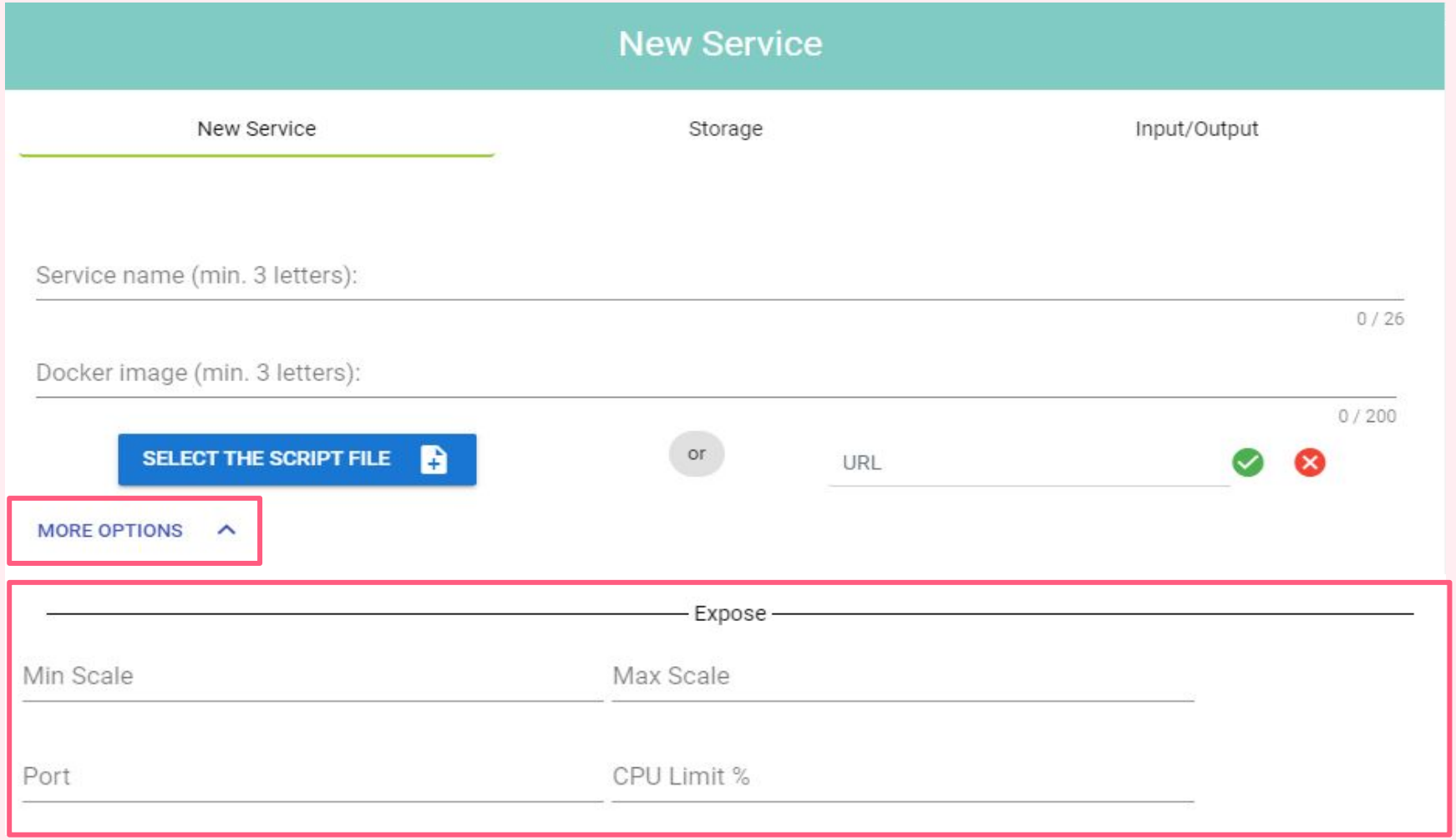
OSCAR: Exposed services

- Long-running services
- Fast inference of pre-trained AI models
- Avoids to load the AI model weights on each invocation
- Allows to directly expose the DEEPaaS API of the AI model from out of the OSCAR service
- Seamless auto-scaling based on average CPU within user-defined limits (bursts of stateless inference requests handled by multiple containers)



Documentation: https://docs.oscar.grycap.net/invoking/#exposed-services

OSCAR: Exposed services

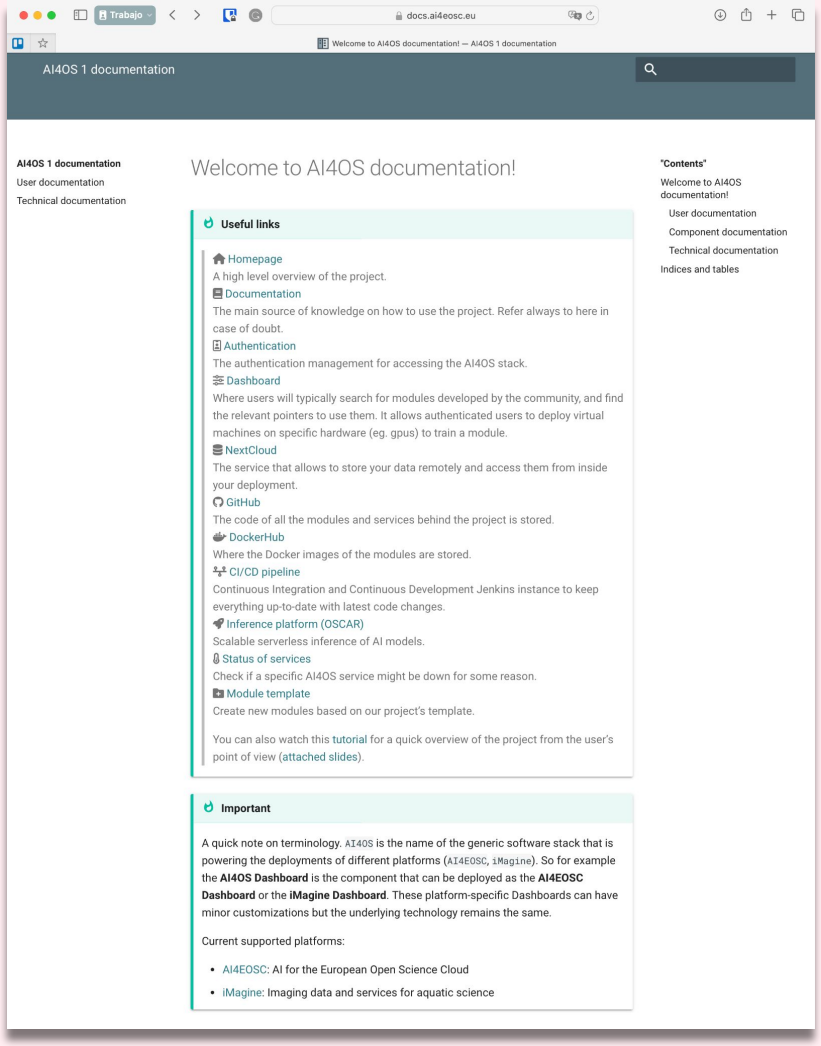


Documentation:
https://docs.oscar.grycap.net/invoking/#exposed-services

Integration with AI4-PAPI

- **AI4-PAPI** is a Platform API for interacting with the AI4EOSC services
- Integration with OSCAR:
  - Create and run OSCAR services that use AI pre-trained models from the AI4-EOSC marketplace
  - Make the inference process transparent to the user → no need to know specifics of OSCAR
- Current status → **In development**
  - https://github.com/ai4os/ai4-papi/tree/oscar

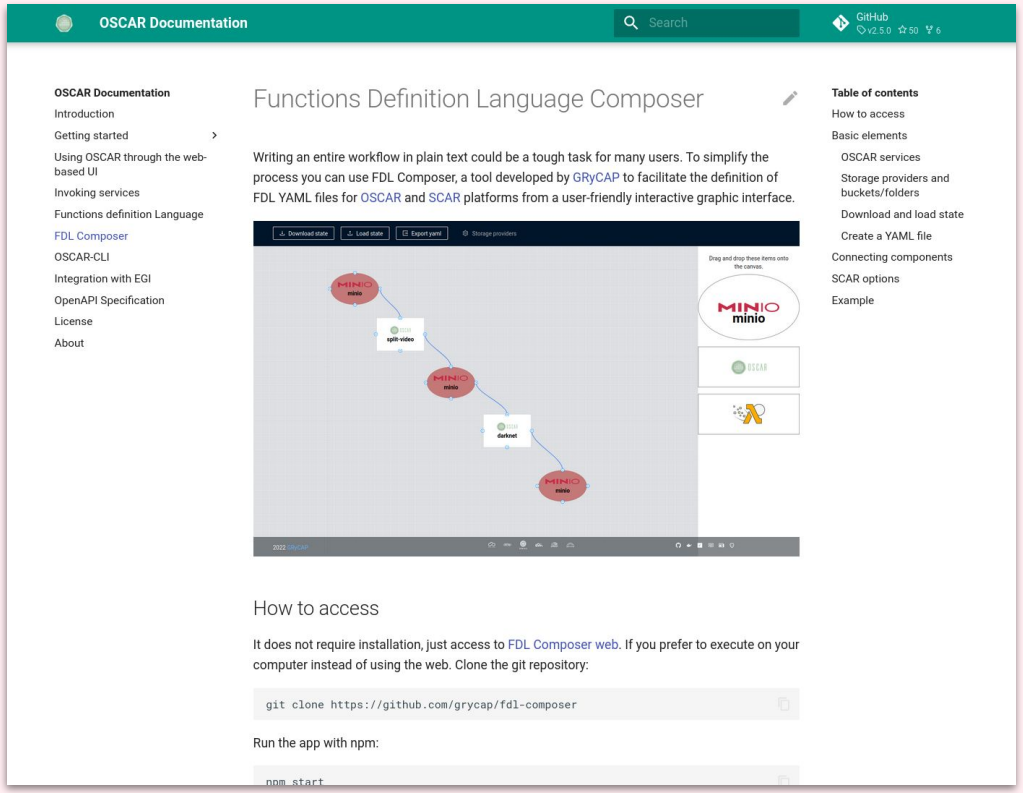
Documentation and Support



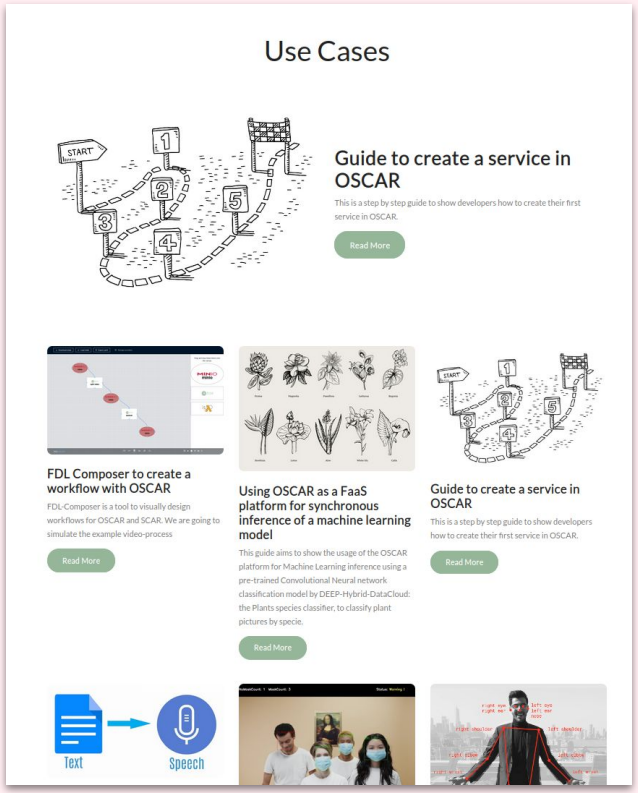
https://doc.ai4eosc.eu

- We can deploy your AI model in the AI4EOSC's OSCAR cluster, to be used for inference. Just let us know:
  - The repository for your Docker image.
  - A sample input file used for inference.
  - The command required to run in the container to trigger the inference of the AI model with the sample input file.
- Open a **ticket** in the AI4EOSC's **Jira** and tag Amanda Calatrava / Germán Moltó or drop an e-mail to products@grycap.upv.es with the tag [AI4EOSC] in the subject.
- **The UPV team is here to help you!**

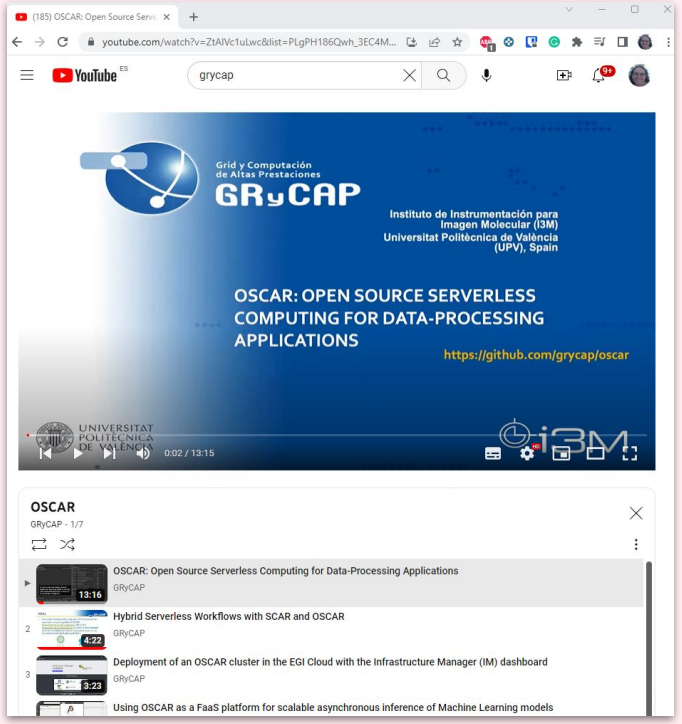
More information



https://docs.oscar.grycap.net



https://oscar.grycap.net/blog/



https://www.youtube.com/watch?v=ZtAlVc1uLwc&list=PLgPH186Qwh_3EC4MInkDkClOXsGYQW_JN

Bratislava Workshop – 15/11/2023

AI4 | eosc

Co-funded by
the European Union

AI4EOSC

amcaar@i3m.upv.es

ai4eosc.eu

Thank you! Any questions?

Amanda Calatrava

AI4 | EOSC

Backup slides

Training: Bringing your AI models to EOSC

Co-funded by
the European Union

oidc-agent installation

- Add in `/etc/apt/sources.list` the appropiate repositories and then:

```
$ sudo apt update
$ sudo apt-get install oidc-agent
$ export OIDC_SOCK=/tmp/oidc-forward
$ eval `oidc-agent-service use`
```

official documentation

- Register an EGI client following the instructions:

```
$ oidc-gen --pub --issuer https://aai.egi.eu/auth/realms/egi --scope "email eduperson_entitlement eduperson_scoped_affiliation eduperson_unique_id" <cli_name>
```

  - Follow the instructions to activate the device with the generated code.

- Execute the following command to ensure that the `$OIDC_SOCK` variable gets the proper value:

```
$  sudo oidc-agent-service start
```

oscar-cli installation

- Commands for Linux 20.04 (official documentation)

```
$  go install github.com/grycap/oscar-cli@latest
```

- OSCAR CLI commands are available in: `/home/<user>/go/bin`

```
$  export PATH="/home/ubuntu/go/bin:$PATH"
```

- You can test the environment by executing:

```
$ oscar-cli cluster list
```

- And start using OSCAR:

```
$ oscar-cli cluster add oscar-cluster https://localhost oscar <oscar_pass>
--disable-ssl
```

```
$ oscar-cli service run plant-classification -i image.jpg -c oscar-ai4eosc-oidc
```

Go installation

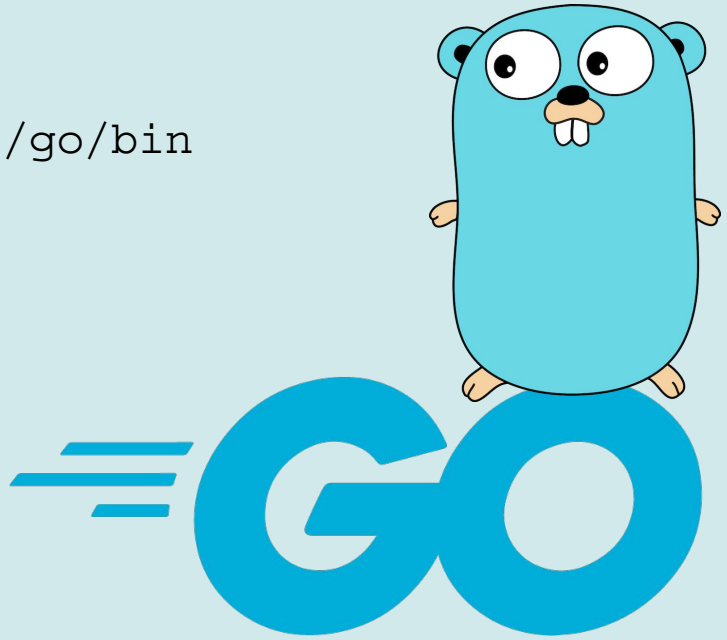
- Commands for Linux 20.04 (official documentation)

```
$ sudo apt update
$ curl -OL https://go.dev/dl/go1.20.4.linux-amd64.tar.gz
$ sudo tar -C /usr/local -xvf go1.20.4.linux-amd64.tar.gz
$ sudo nano ~/.profile
```

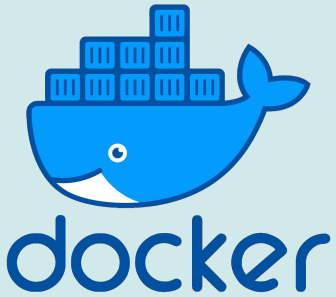
  - Add to the end of the file: `export PATH=$PATH:/usr/local/go/bin`

- Check that everything is in place:

```
$ go version
```



Docker installation



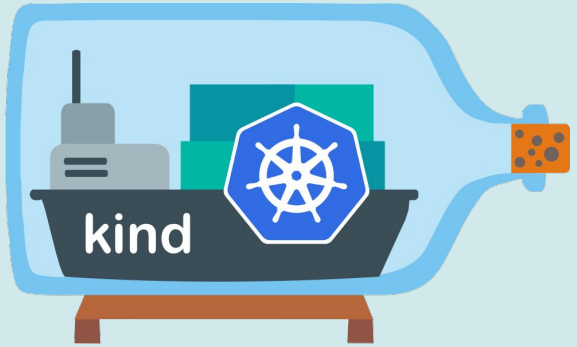
- Commands for Linux 20.04 (official documentation)

```
$ sudo apt update

$ sudo apt install -y apt-transport-https ca-certificates curl
software-properties-common

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"

$ sudo apt update

$ apt-cache policy docker-ce

$ sudo apt install -y docker-ce
```

- If you want to use docker commands without writing “sudo” each time, you can add the user to the *docker* group (to apply the changes, log out and enter again):

```
$ sudo gpasswd -a ${USER} docker
```

Kind installation



- Commands for Linux 20.04:

```
$ go install sigs.k8s.io/kind@v0.19.0
$ export PATH="/home/ubuntu/go/bin:$PATH"
```

- For other distributions and options, see the official documentation .