Grid y Computación de Altas Prestaciones

GRyCAP

Instituto de Instrumentación para Imagen Molecular
Universitat Politècnica de València
Spain

SERVERLESS COMPUTING FOR DATA-PROCESSING APPLICATIONS ON MULTI-CLOUDS (HANDS-ON TUTORIAL)

Germán Moltó, Sebastián Risco, Alfonso Pérez, Miguel Caballer, Diana María Naranjo

IBERGRID 2019
September 23-26, Santiago de Compostela, Spain

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

i3M

WHERE ARE THE SLIDES?

- Slides available in Google Slides

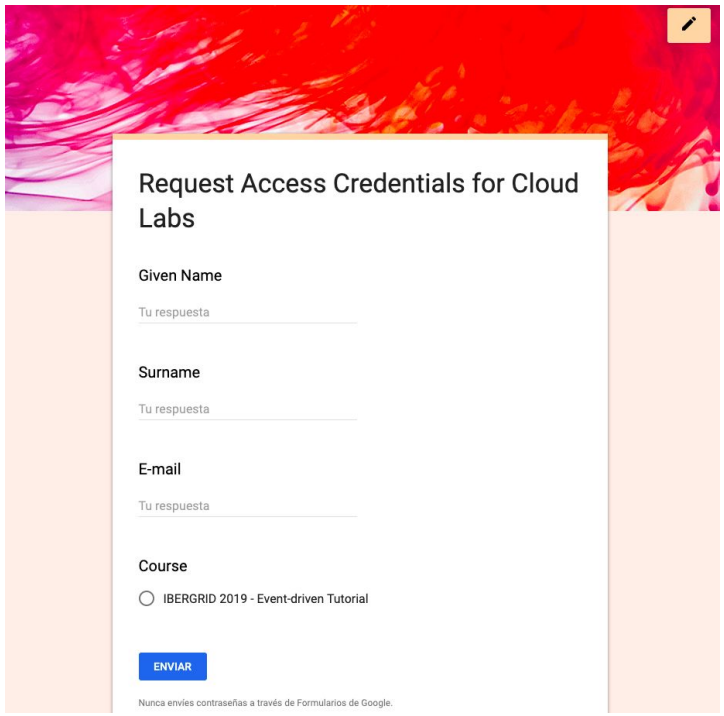
https://docs.google.com/presentation/d/1UWV2FvR2r22Rf6nYAv9wN9dgEeQCmceNFTLqrZ09-Wo/edit?usp=sharing

INTRODUCTION

- Tutorial session on Innovative Computing and Data Processing tools for Researchers
  - Basic tutorial on Event-Driven computing applied to data processing
- Duration: 1h30'
- Instructors:
  - Germán Moltó: Associate Professor at UPV, Spain.
  - Sebas Risco: Researcher at GRyCAP-I3M-UPV
- Goal of the session
  - Provide an overview of the benefits of serverless/event-driven computing exemplified through two open-source tools:
    - SCAR: Execute containers out Docker images on AWS Lambda (integrated with API Gateway and AWS Batch).
    - OSCAR: Deployment of an elastic Kubernetes cluster to support FaaS (Functions as a Service) for long-running executions (using OpenFaaS). Integrated with EGI Applications on Demand and EGI Data Hub.

Request Access to Cloud Labs

- We need your name and e-mail to provide you with temporary access credentials to the Cloud labs to carry out the hands-on activities.
- Scan the code and fill in the form.



INDEX

- Introduction & Lab Environment
- SCAR
- Use case 1: Imagemagick
- Use case 2: Plant Classification in AWS Batch
- OSCAR
- Use case 1: Plant Classification
- Use case 2: Multi-cloud workflow for video processing
- Conclusions

REQUIREMENTS

- This is a hands-on session.

- You should BRING:
  - A laptop with Internet connection and outbound SSH connectivity.
  - An SSH client (Terminal/iTerm, Putty, etc.)
  - A web browser.

- You GET:
  - A Linux user account in a pre-configured Virtual Machine with access credentials to use AWS services (to be used ***exclusively*** to perform the activities during this session). This is used to test the functionality of SCAR.

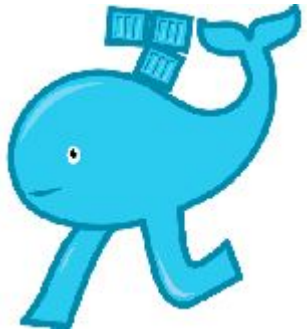
ABOUT THE LAB ENVIRONMENT

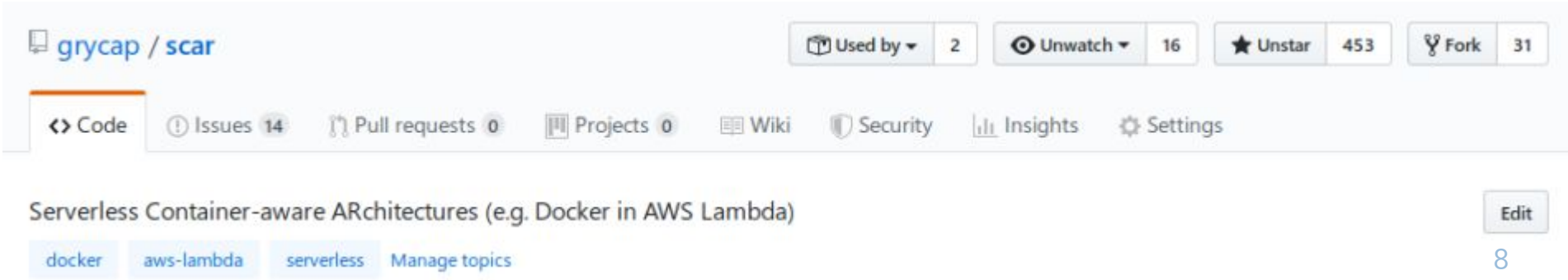
- The user accounts are linked to a master AWS account, which is linked to the instructor's VISA card.
- Do not forget to eliminate all the resources provisioned.
- Connect to **lab.cursocloudaws.net** via SSH using:
  - Username: alucloudXX
  - Password: SSH Password Column, as received in the welcome e-mail.
- Be sure to respect the naming conventions of the exercises since there are policies that prevent otherwise.

SCAR: Introduction

Serverless Container-aware Architectures

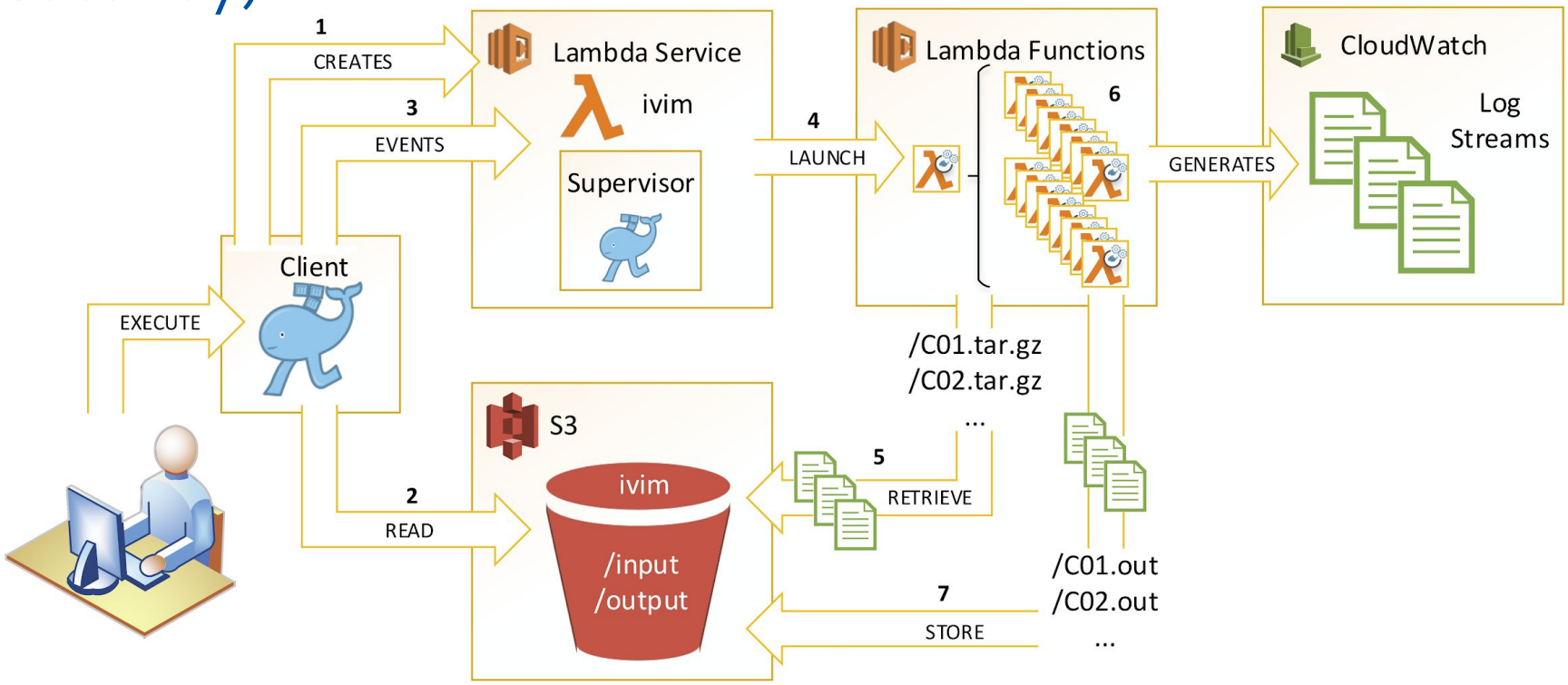
- https://github.com/grycap/scar
- Executes customized runtime environments provided by Docker containers (using uDocker) on AWS Lambda.
- Highly-parallel event-driven file-processing serverless applications
- Bring your own application and runtime





SCAR: Event-driven Computing

- Parallel invocations to Lambda functions that run the user's script in the Docker container to efficiently process data files uploaded to S3 (or invocations to API Gateway)



SCAR: Installation

Installing SCAR

(pre-installed in the lab machine; skip this)

- **Using pip3**
  - Update setuptools (min. version required >= 40.8.0):
    - `pip3 install -U setuptools`
  - Install SCAR using the PyPI package:
    - `pip3 install scar`
- **Cloning Github repository**
  - Clone Github repo:
    - `git clone https://github.com/grycap/scar.git`
  - Install Python dependencies:
    - `pip3 install -U setuptools`
    - `pip3 install -r requirements.txt`
    -
- **Install extra dependencies**
  - `sudo apt -y install zip unzip`

SCAR: Configuration (I)

Step 0. Invoke scar to pre-create config file:

```
scar ls
Config file '/home/alucloud00/.scar/scar.cfg' created.
Please, set a valid iam role in the file field 'role' before the first execution.
```

- Configuration File: $HOME/.scar/scar.cfg
- Log file: $HOME/.scar/scar.log

Step 1. Edit the conf. file to specify the IAM Role:

```
"iam": {
  "role": "arn:aws:iam::974349055189:role/cursocloudaws-lambda-serverless-role"
 }
```

The IAM Role determines the permissions of the Lambda functions to access other AWS services and resources.

SCAR: Configuration (II)

Step 2. Change the default allocated memory to 128 (to save some $$$)

```
...
"memory": 128
...
```

- Note that this can also be changed on a per-function basis.

SCAR: Validation

- **Step 3.** Validate SCAR's installation by creating a dummy function:

```
scar init -i grycap/cowsay -n alucloud$ID-cowsay
Using existent 'faas-supervisor' layer
Creating function package
Function 'alucloud00-cowsay' successfully created.
Log group '/aws/lambda/alucloud00-cowsay' successfully created
```

- $ID expands to your student ID.
- The function allows to run the (in)famous *cowsay* application (https://en.wikipedia.org/wiki/Cowsay), packaged as a Docker image in the grycap/cowsay Docker Hub repository.

SCAR: Validation

- **Step 4.** Execute the cowsay SCAR function:

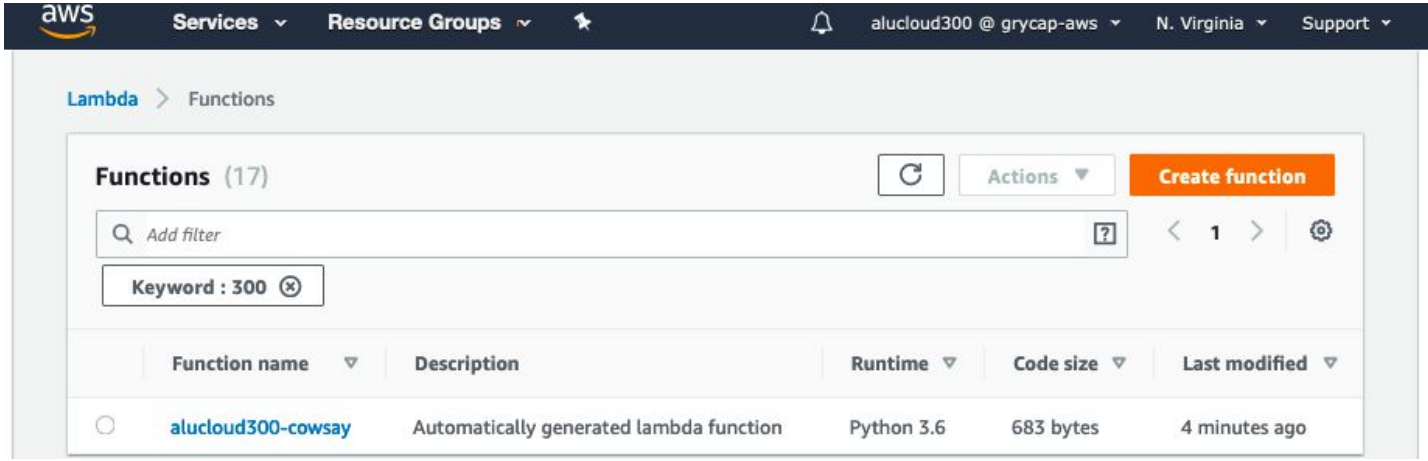
```
scar run -n alucloud$ID-cowsay
Request Id: de4d397b-76fe-4047-980d-40798d59b721
Log Group Name: /aws/lambda/alucloud00-cowsay
Log Stream Name: 2019/09/04/[$LATEST]b0d04731fd2b46ad81761436b23511b1
```

```
 ____________________________________________________________
/ Beifeld's Principle:                                       \
|                                                            |
| The probability of a young man meeting                     |
| a desirable and receptive                                  |
|                                                            |
| young female increases by pyramidical                      |
| progression when he                                        |
|                                                            |
| is already in the company of (1) a                         |
| date, (2) his wife, (3) a                                  |
|                                                            |
| better-looking and richer male friend.                     |
|                                                            |
\ -- R. Beifeld                                              /
 ------------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

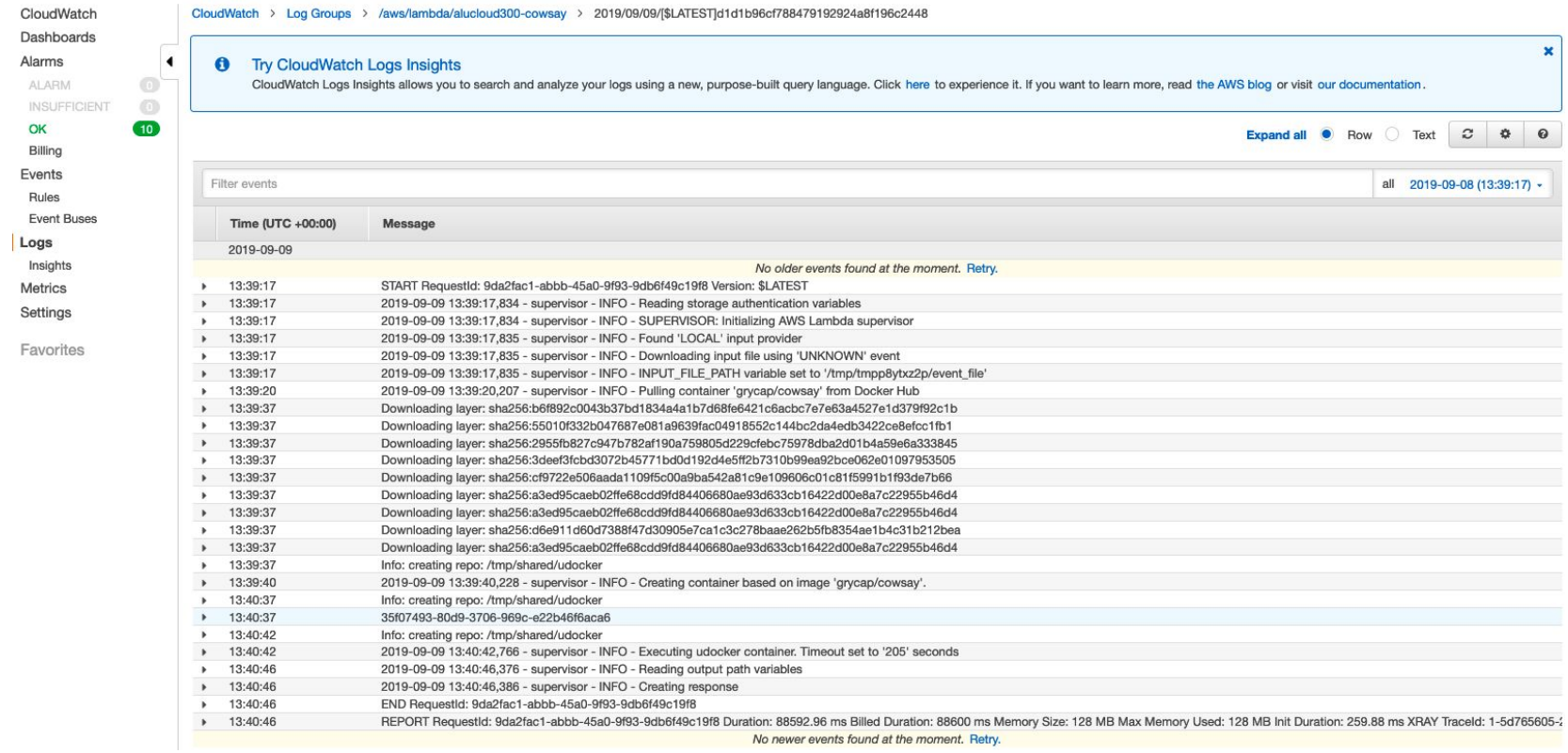
- The first invocation is significantly slower than the rest because the Docker image is being retrieved from Docker Hub. The container's file systems created with uDocker is being cached in the function's scratch directory (512 MB).

SCAR: Resources Created (I)

- **Step 5.** Check in the AWS Management Console the resources created.
  - https://console.aws.amazon.com/console/home
  - Use alucloudXX + the AWS Management Console Password
  - Lambda function and logs stored in CloudWatch Logs.



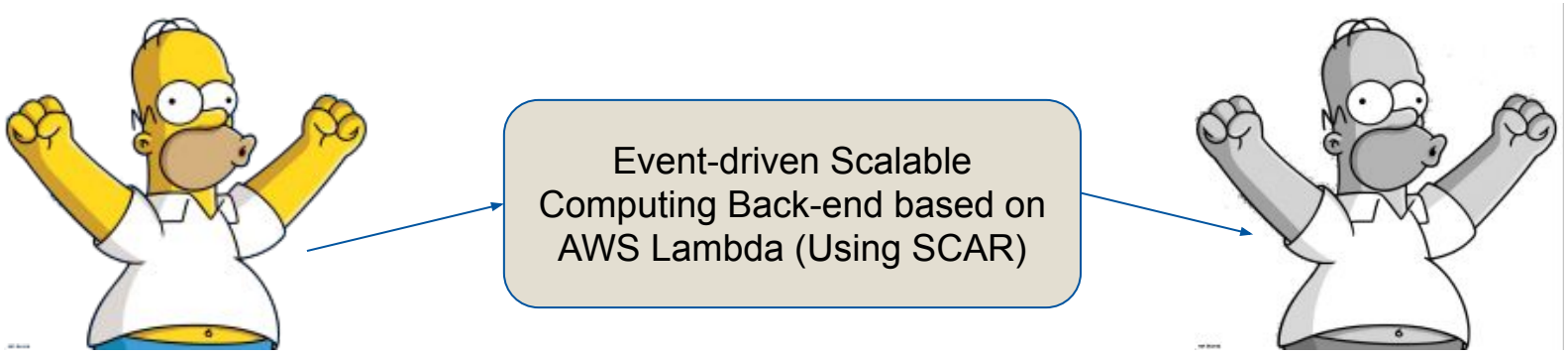
SCAR: Resources Created (II)



- The real consumed memory is shown.
- **STEP 6**: Delete function:
  - **`scar rm -n alucloud$ID-cowsay`**

USE CASE 1: Image Processing

- Goal: Apply an image transformation (grayify) on each file uploaded to an S3 bucket.
- Can be generalised to perform any event-driven processing.
- The back-end automatically scales (up to 3000 function invocations). Asynchronous invocations are queued up in AWS Lambda and retried in order to cope with the workload.



USE CASE 1: Files Involved

Run ImageMagick on AWS Lambda

- **Files required (available in /opt/ibergrid/use-case-1)**
  - Script to execute (grayify-image.sh)

```
FILE_NAME=`basename $INPUT_FILE_PATH`
OUTPUT_FILE=$TMP_OUTPUT_DIR/$FILE_NAME
convert $INPUT_FILE_PATH -type Grayscale $OUTPUT_FILE
```

  - Dockerfile with container definition (Dockerfile)

```
FROM bitnami/minideb:jessie
RUN install_packages imagemagick
ENV PATH
/usr/lib/x86_64-linux-gnu/ImageMagick-6.8.9/bin-Q16/:$PATH
```

https://hub.docker.com/r/grycap/imagemagick

Run ImageMagick on AWS Lambda

- **Optional file (recommended):**
  - SCAR configuration file (scar-imagemagick.yaml)

```
functions:
  alucloud00-imagemagick:
    image: grycap/imagemagick
    init_script: grayify-image.sh
    s3:
      input_bucket: alucloud00-iber
```

Function's name

Image id

Bucket's name

- Complete example on Github:
  https://github.com/grycap/scar/tree/master/examples/imagemagick

USE CASE 1: Image Processing

STEP 0: Configure SCAR's function definition file:

```
mkdir $HOME/ibergrid
cp -r /opt/ibergrid/use-case-1 $HOME/ibergrid
sed -i s/00/$ID/
$HOME/ibergrid/use-case-1/scar-imagemagick.yaml
```

This just replaces 00 in the file with the student identifier.

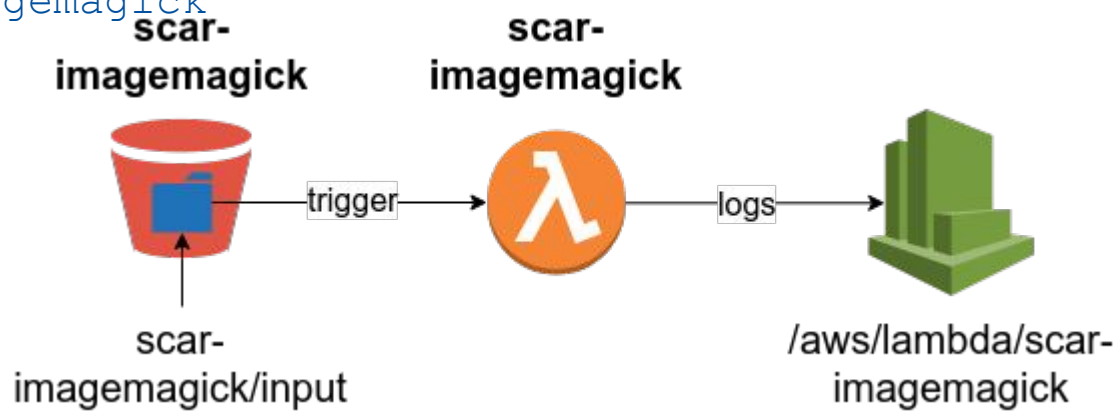
```
functions:
  alucloud00-imagemagick:
    image: grycap/imagemagick
    init_script: grayify-image.sh
    s3:
      input_bucket: alucloud00-iber
```

USE CASE 1: Function Creation

Run ImageMagick on AWS Lambda

- **STEP 1: Create the Lambda function:**

```
scar init -f
$HOME/ibergrid/use-case-1/scar-imagemagick.yaml
Using existent 'faas-supervisor' layer
Creating function package
Function 'alucloud00-imagemagick' successfully created.
Log group '/aws/lambda/alucloud00-imagemagick' successfully
created.
Folder 'alucloud00-imagemagick/input/' created in bucket
'scar-imagemagick
```

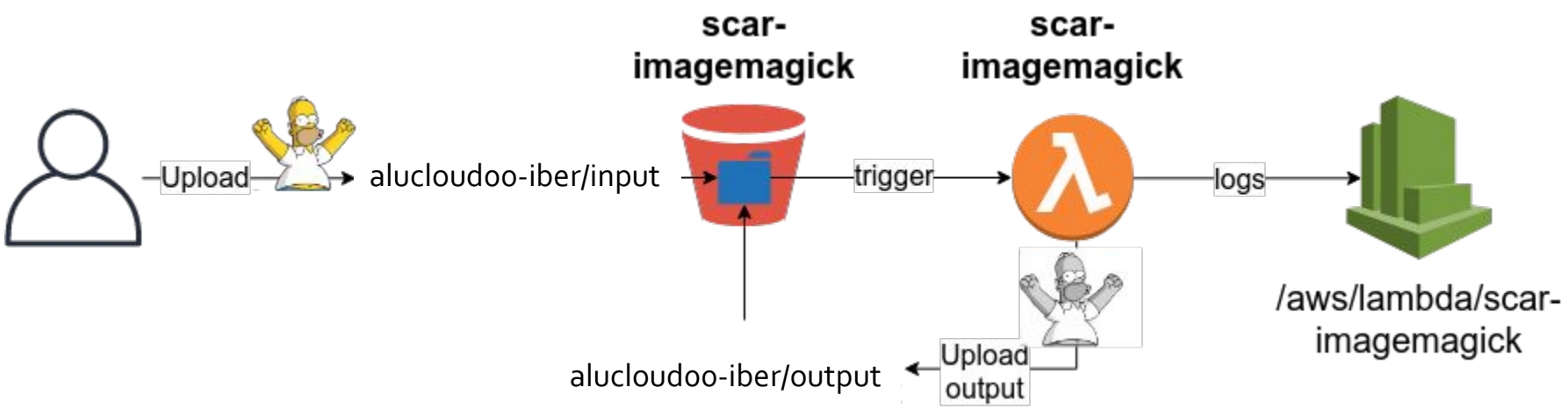


USE CASE 1: Trigger Function

Run ImageMagick on AWS Lambda

- **STEP 2: Upload a file to the S3 bucket:**

```
scar put -b alucloud$ID-iber/alucloud$ID-imagemagick/input
-p /opt/ibergrid/use-case-1/homer.png
Uploading file '/opt/ibergrid/use-case-1/homer.png' to bucket
'alucloud00-iber' with key 'alucloud00-imagemagick/input/homer.png
```



USE CASE 1: Logs

Run ImageMagick on AWS Lambda

- STEP 3: Check CloudWatch Logs:

```
scar log -f
$HOME/ibergrid/use-case-1/scar-imagemagick.yaml
```

```
START RequestId: 76dbb05a-6e9c-40a2-9e14-817cade20b30 Version: $LATEST
2019-09-04 14:47:58,859 - supervisor - INFO - Storage event found.
2019-09-04 14:47:58,859 - supervisor - INFO - S3 event created
2019-09-04 14:47:58,859 - supervisor - INFO - Reading storage authentication variables
2019-09-04 14:47:58,860 - supervisor - INFO - SUPERVISOR: Initializing AWS Lambda supervisor
2019-09-04 14:47:58,860 - supervisor - INFO - Found 'S3' input provider
2019-09-04 14:47:58,860 - supervisor - INFO - Downloading input file using 'S3' event
2019-09-04 14:47:58,861 - supervisor - INFO - Downloading item from bucket 'alucloud00-iber' with key
'alucloud00-imagemagick/input/homer.png'
2019-09-04 14:47:59,262 - supervisor - INFO - Successful download of file 'alucloud00-imagemagick/input/homer.png' from
bucket 'alucloud00-iber' in path '/tmp/tmp6xnxlnuh/homer.png'
2019-09-04 14:47:59,262 - supervisor - INFO - INPUT_FILE_PATH variable set to '/tmp/tmp6xnxlnuh/homer.png'
2019-09-04 14:48:01,402 - supervisor - INFO - Container image 'grycap/imagemagick' already available
2019-09-04 14:48:03,523 - supervisor - INFO - Container already available
Info: creating repo: /tmp/shared/udocker
2019-09-04 14:48:05,722 - supervisor - INFO - Executing udocker container. Timeout set to '283' seconds
2019-09-04 14:48:11,311 - supervisor - INFO - Reading output path variables
2019-09-04 14:48:11,320 - supervisor - INFO - Found 'S3' output provider
2019-09-04 14:48:11,321 - supervisor - INFO - Searching for files to upload in folder '/tmp/tmpxbdn8pww'
2019-09-04 14:48:11,321 - supervisor - INFO - Found the following files to upload: '['/tmp/tmpxbdn8pww/homer.png']'
2019-09-04 14:48:11,321 - supervisor - INFO - Uploading file
'alucloud00-imagemagick/output/76dbb05a-6e9c-40a2-9e14-817cade20b30/homer.png' to bucket 'alucloud00-iber'
2019-09-04 14:48:11,722 - supervisor - INFO - Changing ACLs for public-read for object in bucket 'alucloud00-iber' with key
'alucloud00-imagemagick/output/76dbb05a-6e9c-40a2-9e14-817cade20b30/homer.png'
2019-09-04 14:48:12,054 - supervisor - INFO - Creating response
END RequestId: 76dbb05a-6e9c-40a2-9e14-817cade20b30
REPORT RequestId: 76dbb05a-6e9c-40a2-9e14-817cade20b30      Duration: 13196.80 ms      Billed Duration: 13200 ms  Memory
Size: 128 MB  Max Memory Used: 128 MB
```

USE CASE 1: Data Access (CLI)

Run ImageMagick on AWS Lambda

- **STEP 4: Check S3 Bucket:**

```
scar ls -b alucloud$ID-iber/alucloud$ID-imagemagick
scar-imagemagick/input/homer.png
scar-imagemagick/output/08273a80-1ee1-4396-b096-
c598b8ebd01d/homer.png
```

- **Download output file:**

```
scar get -b alucloud$ID-iber/alucloud$ID-imagemagick/output -p  .
Downloading file
'alucloud00-imagemagick/output/181dbf2c-5ba8-4f06-8822-632940
60ce61/homer.png' from bucket 'alucloud00-iber' in path
'./alucloud00-imagemagick/output/181dbf2c-5ba8-4f06-8822-6329
4060ce61/homer.png'
```

USE CASE 1: Data Access (GUI)

STEP 5: Access the S3 Console

aws Services Resource Groups alucloud300 @ grycap-aws Global Support

Amazon S3 > alucloud300-iber > alucloud300-imagemagick > output > 41c27647-4891-4d59-a94a-9f888e937ed8

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload + Create folder Download Actions US East (N. Virginia)

Viewing 1 to 1

Name	Last modified	Size	Storage class
homer.png	Sep 9, 2019 3:54:03 PM GMT+0200	297.1 KB	Standard

Viewing 1 to 1

USE CASE 1: Removing Resources

STEP 6: Remove the function

```
scar rm -f
$HOME/ibergrid/use-case-1/scar-imagemagick.yaml

Log group '/aws/lambda/alucloud00-imagemagick' successfully deleted.
Bucket notifications successfully deleted
Function 'alucloud00-imagemagick' successfully deleted
```

Run ImageMagick on AWS Lambda

- **Conclusions:**
  - SCAR allows to execute container images stored in Docker Hub in AWS Lambda.
  - Execute customized user script inside the container.
  - Link S3 bucket and folders, CloudWatch Logs and Lambda functions with one command (scar init …)
  - Launch image processing with one command (scar put …)
  - Retrieve logs and function output.
  - Highly-scalable processing service with scale to zero (including cost!) and Free Usage Tier.

USE CASE 2: Plant Classification

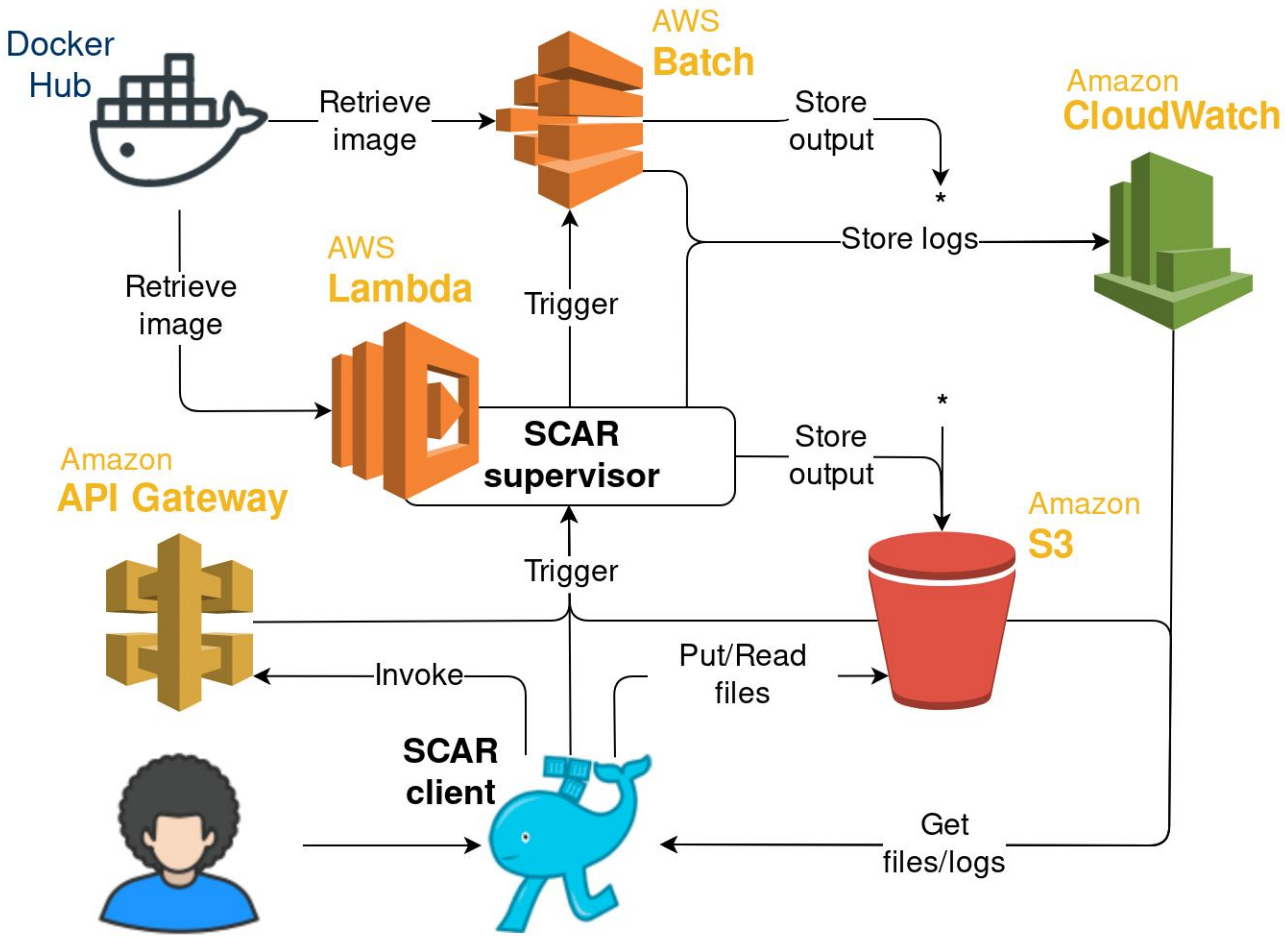
Plant Classification with Lasagne/Theano

- Example extracted from:
  - https://github.com/deephdc/plant-classification-theano
- Pretrained convolutional network
- Classifies a plant image among 6K plant species

Hybrid DataCloud

USE CASE 2: Plants Classification

SCAR: AWS Batch integration



USE CASE 2: Plants Classification

SCAR configuration file (scar-plant-classification.yaml)

```
functions:
  alucloud00-plants:
    image: deephdc/deep-oc-plant-classification-theano
    memory: 128
    execution_mode: batch
    s3:
      input_bucket: alucloud00-iber
   init_script: bootstrap-plants.sh
   batch:
     vcpus: 1
     memory: 1024
```

- Complete example on Github: https://github.com/grycap/scar/tree/master/examples/plant-classification

USE CASE 2: Plants Classification on Batch

STEP 1: Add the following subnets to the SCAR configuration file ($HOME/.scar/scar.cfg):

```
"subnets": ["subnet-2bfb6c4f","subnet-c2f25afd"]
```

STEP 2: Adapt YAML to student's ID:

```
cp -r /opt/ibergrid/use-case-2 $HOME/ibergrid
sed s/00/$ID/ -i
$HOME/ibergrid/use-case-2/scar-plant-classification.yaml
```

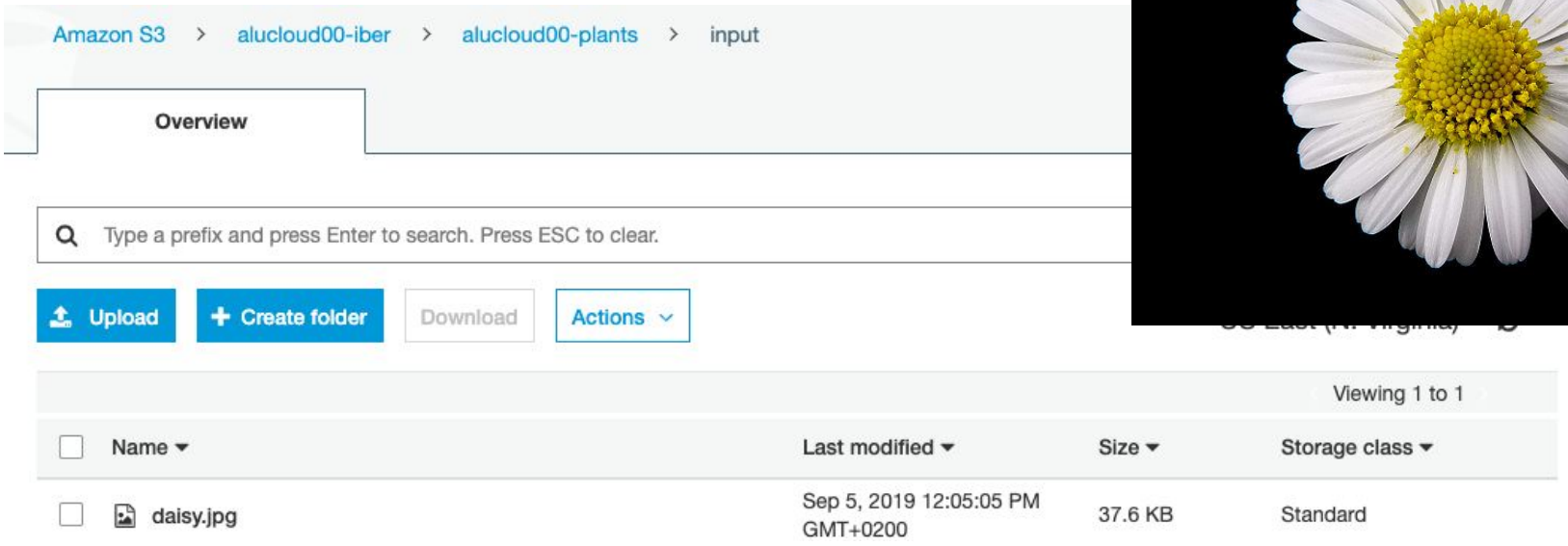
STEP 3: Create SCAR Function

```
scar init -f
$HOME/ibergrid/use-case-2/scar-plant-classification.yaml
```

USE CASE 2: Plants Classification on Batch

STEP 4: Upload file to bucket

```
scar put -b alucloud$ID-iber/alucloud$ID-plants/input -p
/opt/ibergrid/use-case-2/daisy.jpg
```



USE CASE 2: Plants Classification on Batch

AWS Batch enqueues the job to be run:

AWS Batch
- Dashboard
- Jobs
- Job definitions
- Job queues
- Compute environments

Create job

Job queues

Last loaded: 12:05:45 pm 09/05/19

Name	Priority	SUBMITTED	PENDING	RUNNABLE	STARTING	RUNNING	FAILED	SUCCEEDE
alucloud00-plants	1	0	0	1	0	0	0	0
scar-plant-classification	1	0	0	0	0	0	0	0
scar-yolo-video	1	0	0	0	0	0	0	0

A new EC2 instance (VM) is (*eventually*) provisioned:

EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Reserved Instances
Dedicated Hosts
Scheduled Instances
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes

Launch Instance | Connect | Actions

search : m3.medium | Add filter | 1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
	i-0d7a693471aa0ef05	m3.medium	us-east-1b	running	Initializing	None	ec2-3-89-146-102.com...

Instance: i-0d7a693471aa0ef05 Public DNS: ec2-3-89-146-102.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

Instance ID	i-0d7a693471aa0ef05	Public DNS (IPv4)	ec2-3-89-146-102.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	3.89.146.102
Instance type	m3.medium	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-28-105.ec2.internal
Availability zone	us-east-1b	Private IPs	172.31.28.105
Security groups	default. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-83a213fb (default)
AMI ID	amzn-ami-2018.03.u-amazon-ecs-optimized (ami-0d09143c6fc181fe3)	Subnet ID	subnet-c2f25afd (subnet-default-1b-public)
Platform	-	Network interfaces	eth0
IAM role	ecsInstanceRole	Source/dest. check	True

USE CASE 2: AWS Batch Job Details

Job details

Job status

Status RUNNABLE

Created at 12:24:31 pm 09/05/19

Started at --

Queue arn:aws:batch:us-east-1:974349055189:job-queue/alucloud00-plants

Job attributes

Job id 0c5635f7-d148-486c-8b0e-79c82cbdf1cf

Job name alucloud00-plants

Job definition arn:aws:batch:us-east-1:974349055189:job-definition/alucloud00-plants:5

Depends on

Environment variables

STORAGE_PATH_INPUT_361423 alucloud00-iber

SCRIPT IyEgL2Jpbi9zaApjdXJsIGh0dHBzOi8vczMuYW1hem9uYXdzLmNvbS9zY2FyLXRl

AWS_LAMBDA_REQUEST_ID 0e1326db-f64e-41bf-acce-e723a154eca5

STORAGE_PATH_OUTPUT_361423 alucloud00-iber

CONTEXT {"function_name": "alucloud00-plants", "memory_limit_in_mb": 1024, "aws_request_id": "0e1326db-f64e-41bf-acce-e723a154eca5", "log_group_name": "/aws/lambda/alucloud00-plants", "log_stream_name": "2019/09/05/[$LATEST]9c002233b16947f096e111bfc779cfc2"}

AWS_LAMBDA_FUNCTION_NAME alucloud00-plants

EVENT {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "2019-09-05T10:24:30.514Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAJ2QG6PUWMNOYKCICY"}, "requestParameters": {"sourceIPAddress": "52.202.41.247"}, "responseElements": {"x-amz-request-id": "029353A8F28FE5E4", "x-amz-id-2": "V2aCSIxGb5yN7Xw0SMb/GkIRPQG/duQsQFtO1ka0QH61OPgJmHQ0hJ3Xlgw", "s3": {"s3SchemaVersion": "1.0", "configurationId": "NmFjMmFhZTYtYWNhNC00NjViLTk4YWEtMDQ3NGI4YTI5NjQx", "bucket": {"name": "alucloud00-iber", "ownerIdentity": {"principalId": "A1B93C6Q3HXTTW"}, "arn": "arn:aws:s3:::alucloud00-iber"}, "object": {"key": "alucloud00-

USE CASE 2: AWS Batch Job Logs

CloudWatch > Log Groups > /aws/batch/job > alucloud00-plants/default/8d6e57b3-ee30-4eb2-abfa-d7087aa24159

Try CloudWatch Logs Insights

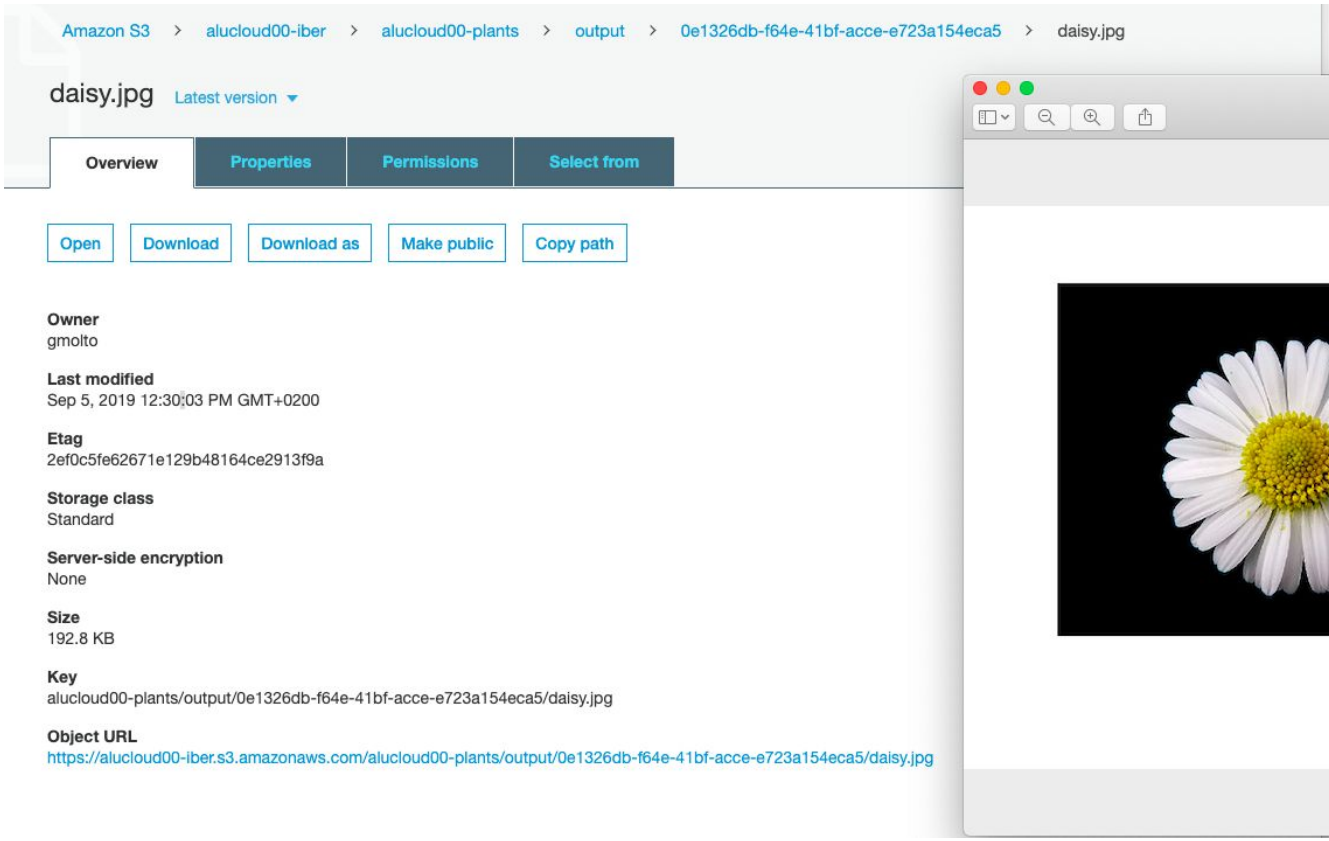
CloudWatch Logs Insights allows you to search and analyze your logs using a new, purpose-built query language. Click here to experience it. If you want to learn more, read the AWS blog or visit our documentation.

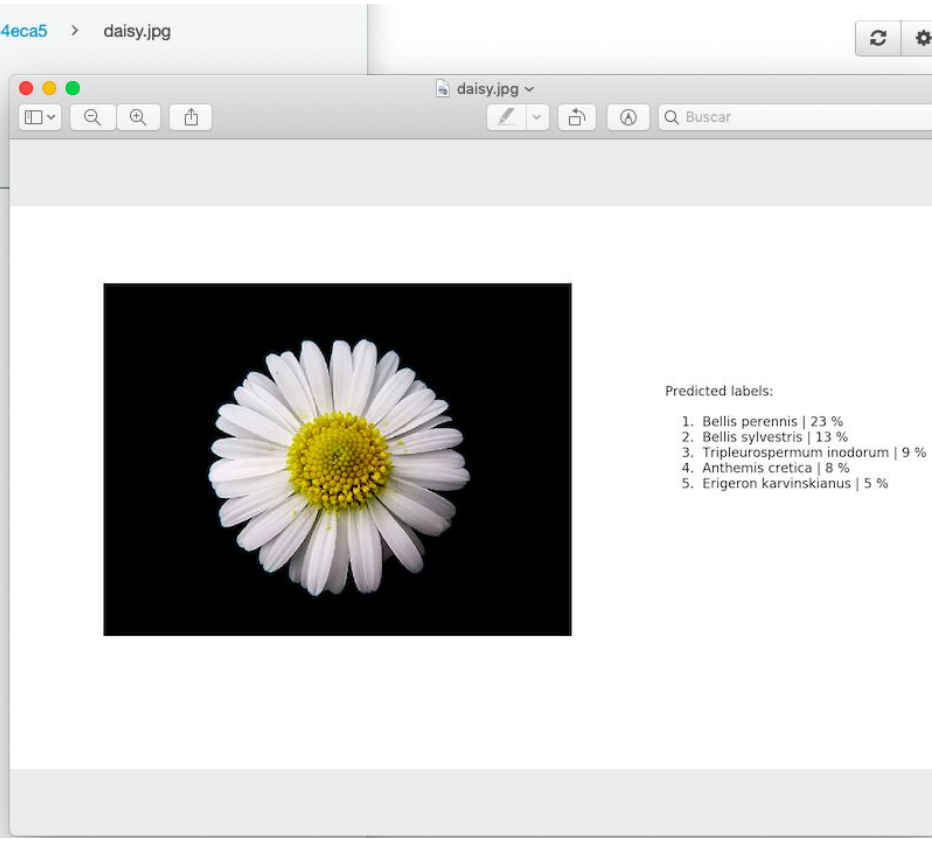
Expand all ● Row ○ Text

Filter events | all 30s 5m 1h 6h 1d 1w custom

Time (UTC +00:00)	Message
2019-09-05	
	No older events found at the moment. Retry.
10:29:07	2019-09-05 10:29:07,755 - supervisor - INFO - Storage event found.
10:29:07	2019-09-05 10:29:07,757 - supervisor - INFO - S3 event created
10:29:07	2019-09-05 10:29:07,757 - supervisor - INFO - Reading storage authentication variables
10:29:07	2019-09-05 10:29:07,757 - supervisor - INFO - SUPERVISOR: Initializing Binary supervisor
10:29:07	2019-09-05 10:29:07,757 - supervisor - INFO - Found 'S3' input provider
10:29:07	2019-09-05 10:29:07,758 - supervisor - INFO - Downloading input file using 'S3' event
10:29:07	2019-09-05 10:29:07,758 - supervisor - INFO - Downloading item from bucket 'alucloud00-iber' with key 'alucloud00-plants/input/daisy.jpg'
10:29:07	2019-09-05 10:29:07,994 - supervisor - INFO - Successful download of file 'alucloud00-plants/input/daisy.jpg' from bucket 'alucloud00-iber' in path '/tmp/tmpdz2
10:29:07	2019-09-05 10:29:07,995 - supervisor - INFO - INPUT_FILE_PATH variable set to '/tmp/tmpdz2pojdl/daisy.jpg'
10:29:07	2019-09-05 10:29:07,995 - supervisor - INFO - Script file created in '/tmp/tmpdz2pojdl/script.sh'
10:29:07	2019-09-05 10:29:07,996 - supervisor - INFO - Executing user defined script: '/tmp/tmpdz2pojdl/script.sh'
10:30:02	2019-09-05 10:30:02,696 - supervisor - INFO - % Total % Received % Xferd Average Speed Time Time Time Current
10:30:02	Dload Upload Total Spent Left Speed
10:30:02	0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0 0 0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0 100 3092 100 3092 0 0 31232 0 --:--:-- --:--:-- --:--:-- 30920
10:30:02	SCRIPT: Invoked classify_image.py. File available in /tmp/tmpdz2pojdl/daisy.jpg.
10:30:02	OUTPUT FILE: /tmp/tmp2hc3t52h/daisy.jpg
10:30:02	WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.
10:30:02	/usr/local/lib/python2.7/dist-packages/matplotlib/patches.py:83: UserWarning: Setting the 'color' property will overridethe edgecolor or facecolor properties.
10:30:02	warnings.warn("Setting the 'color' property will override"
10:30:02	Loading the model...
10:30:02	Image number: 0
10:30:02	2019-09-05 10:30:02,697 - supervisor - INFO - Reading output path variables
10:30:02	2019-09-05 10:30:02,697 - supervisor - INFO - Found 'S3' output provider
10:30:02	2019-09-05 10:30:02,698 - supervisor - INFO - Searching for files to upload in folder '/tmp/tmp2hc3t52h'
10:30:02	2019-09-05 10:30:02,698 - supervisor - INFO - Found the following files to upload: '['/tmp/tmp2hc3t52h/daisy.jpg']'
10:30:02	2019-09-05 10:30:02,698 - supervisor - INFO - Uploading file 'alucloud00-plants/output/0e1326db-f64e-41bf-acce-e723a154eca5/daisy.jpg' to bucket 'alucloud0
10:30:02	2019-09-05 10:30:02,856 - supervisor - INFO - Changing ACLs for public-read for object in bucket 'alucloud00-iber' with key 'alucloud00-plants/output/0e1326db
10:30:02	2019-09-05 10:30:02,973 - supervisor - INFO - Creating response
	No newer events found at the moment. Retry.

USE CASE 2: AWS Batch Job Result





- Output file automatically generated. Flower identified and tagget with a certain confidence.
- The elastic cluster will eventually scale to zero

USE CASE 2: Terminating Resources


```
scar rm -f
$HOME/ibergrid/use-case-2/scar-plant-classification.yaml

Log group '/aws/lambda/alucloud00-plants' successfully deleted.
Bucket notifications successfully deleted
Function 'alucloud00-plants' successfully deleted.
Job definitions deleted
Job queue delete
```

- Output bucket is never deleted (to prevent accidental data loss). You can remove it with AWS CLI:

```
aws s3 rb --force s3://alucloud$ID-iber

delete: s3://alucloud00-iber/alucloud00-imagemagick/input/
delete:
s3://alucloud00-iber/alucloud00-imagemagick/output/1c230bdd-0112-4373-a54d-a86b7f8fcb30/homer.png
delete:
s3://alucloud00-iber/alucloud00-imagemagick/output/3e621627-a5a9-4484-8ba4-aca601057aef/homer.png
delete:
s3://alucloud00-iber/alucloud00-plants/output/0e1326db-f64e-41bf-acce-e723a154eca5/daisy.jpg
```

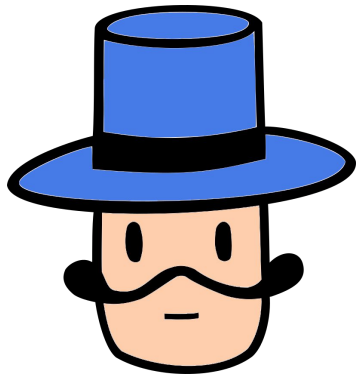
SCAR: Summary & Recap

- The integration of SCAR with AWS Batch allows to support the execution of computationally intensive jobs, packaged as Docker images, in response to events (such as file uploads to S3).
- AWS Batch supports scale-to-zero, thus cutting down the costs of the computing back-end.
- You can have a computing service deployed in the Cloud at zero cost that can rapidly and automatically scale to hundreds of instances to manage the incoming job workload.
- There is more functionality in SCAR not covered in this tutorial:
  - API Gateway as an event source, to provide a REST API as a service to trigger the execution of your computing back-end.

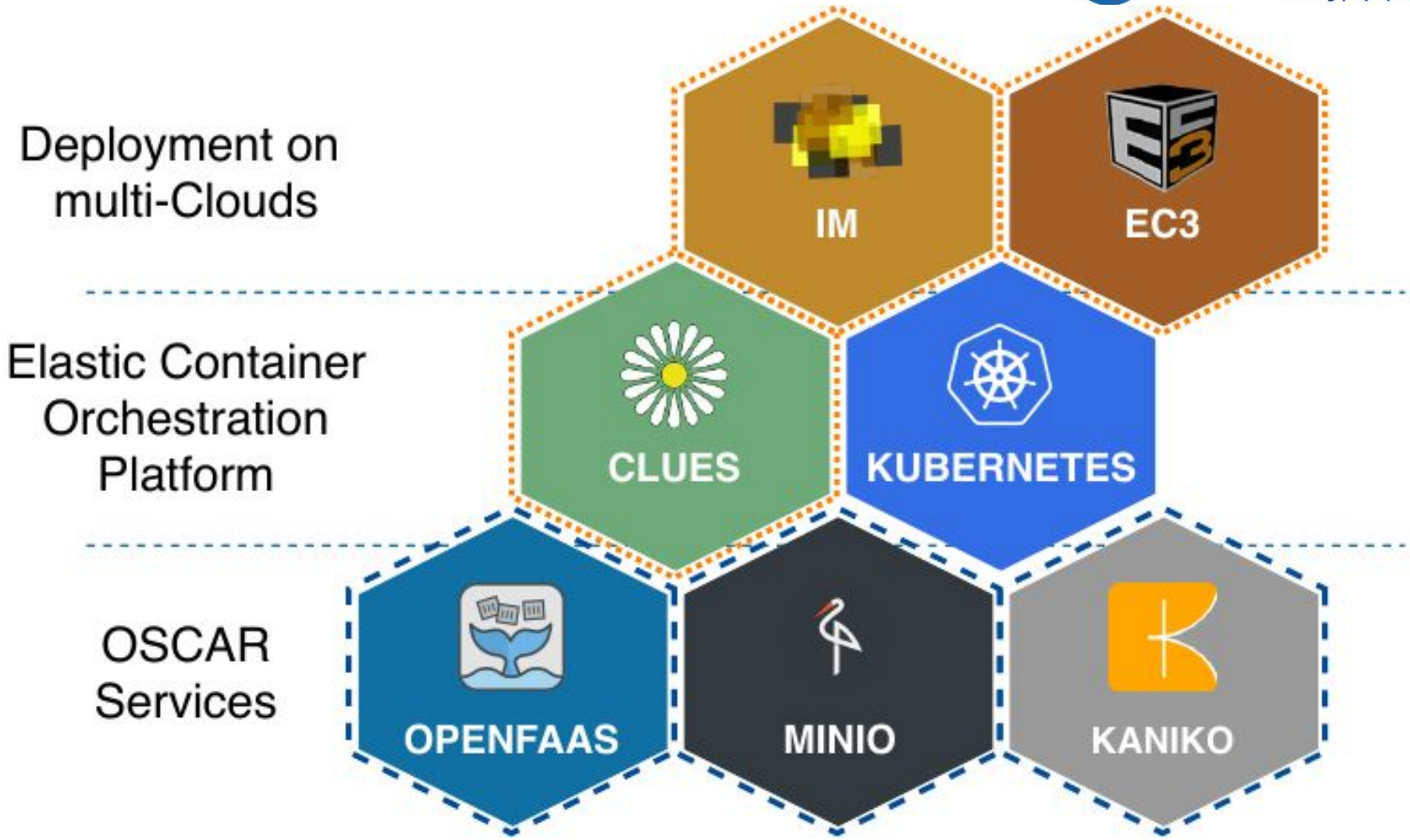
Open Source Serverless Computing for Data-Processing Applications

- Supports the Functions as a Service (FaaS) computing model for file-processing applications.
- It can be automatically deployed on multiple Cloud providers.
- Allows to execute customized runtime environments on an elastic Kubernetes cluster.
- Open-source platform

http://github.com/grycap/oscar



OSCAR COMPONENTS

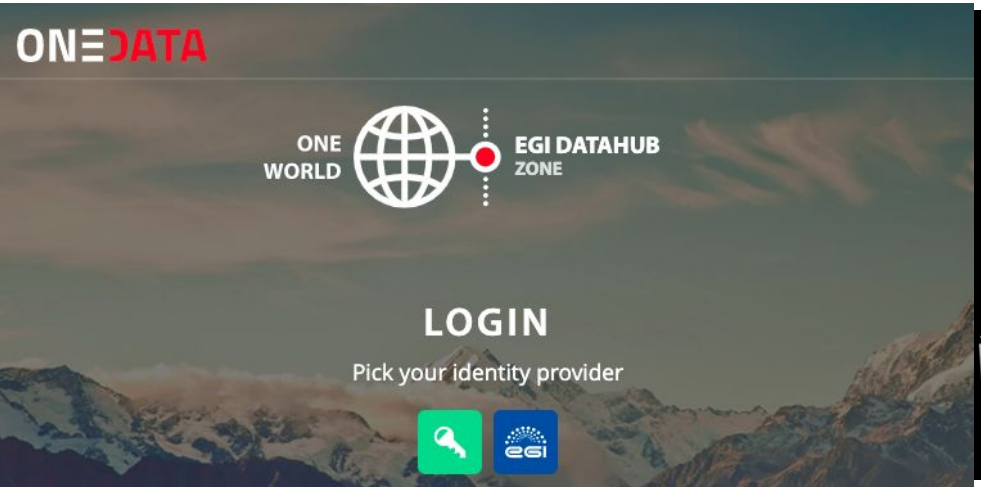


http://github.com/grycap

INTEGRATION WITH EGI

OSCAR is integrated:

- In the EGI Applications on Demand, through the EC3 portal.
- With the EGI DataHub to use Onedata spaces as sources of events and output data storage back-end.
- Demo: https://www.youtube.com/watch?v=ZtAlVc1uLwc



Deploying the infrastructure (with EC3)

```
$ ./ec3 launch oscar oscar_latest ubuntu16 -a auth.dat -u
https://appsgrycap.i3m.upv.es:31443/im
Creating infrastructure
Infrastructure successfully created with ID: XXX-XXX-XXX
Front-end configured with IP XXX.XXX.XXX.XXX
Transferring infrastructure
Front-end ready!
```

```
$ ./ec3 ls
   name      state            IP         nodes   provider
-------------------------------------------------------------
 oscar  configured  XXX.XXX.XXX.XXX    0    OpenNebula
```

```
$ ./ec3 ssh oscar
ubuntu@kubeserver:~$
```

- This can also be done with the web-based GUI provided by the EC3 portal.

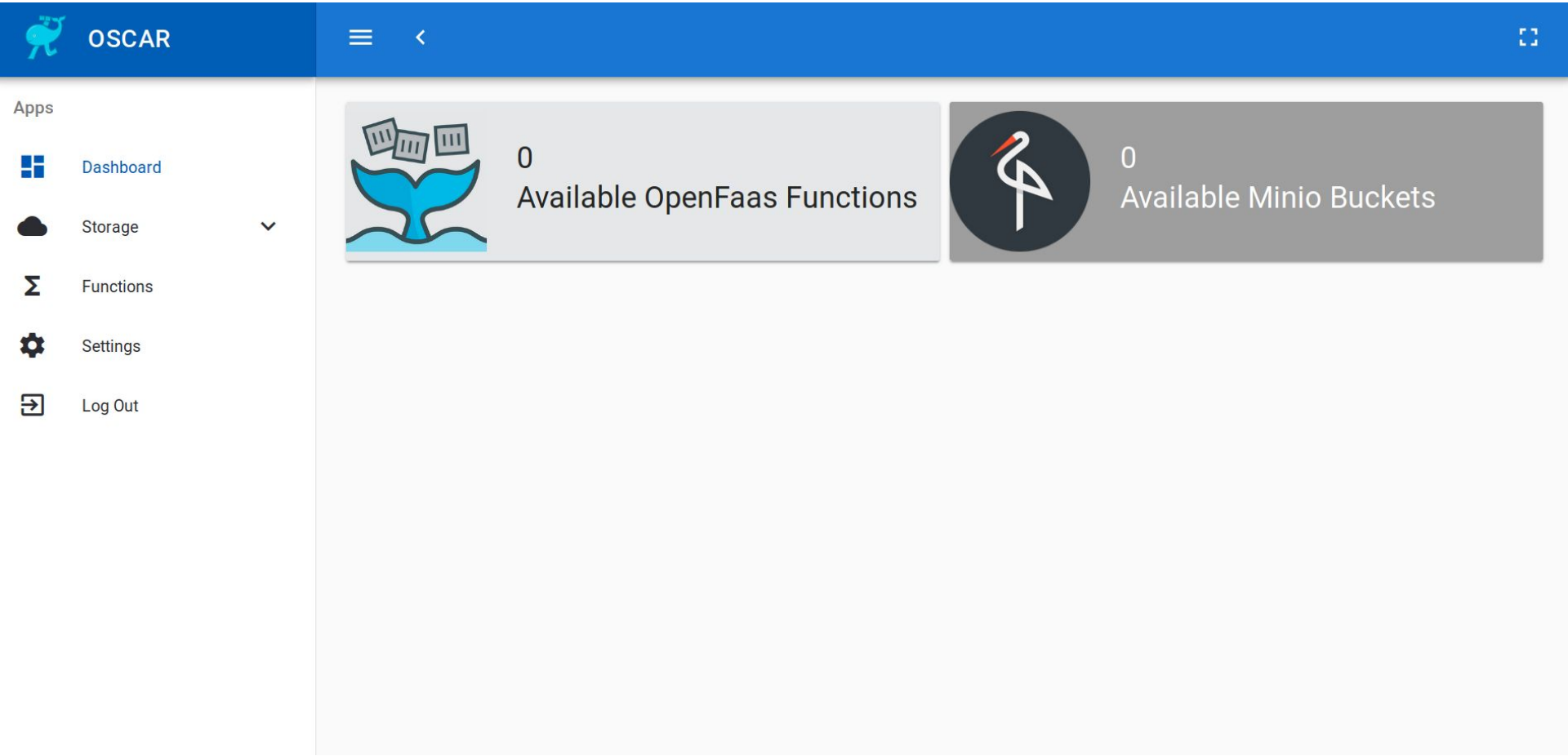
USE CASE 1: Plant Classification

Plant Classification with Lasagne/Theano

- Example extracted from:
  - https://github.com/indigo-dc/plant-classification-theano
- Pretrained convolutional network
- Classifies a plant image among 6K plant species

USE CASE 1: Plant Classification

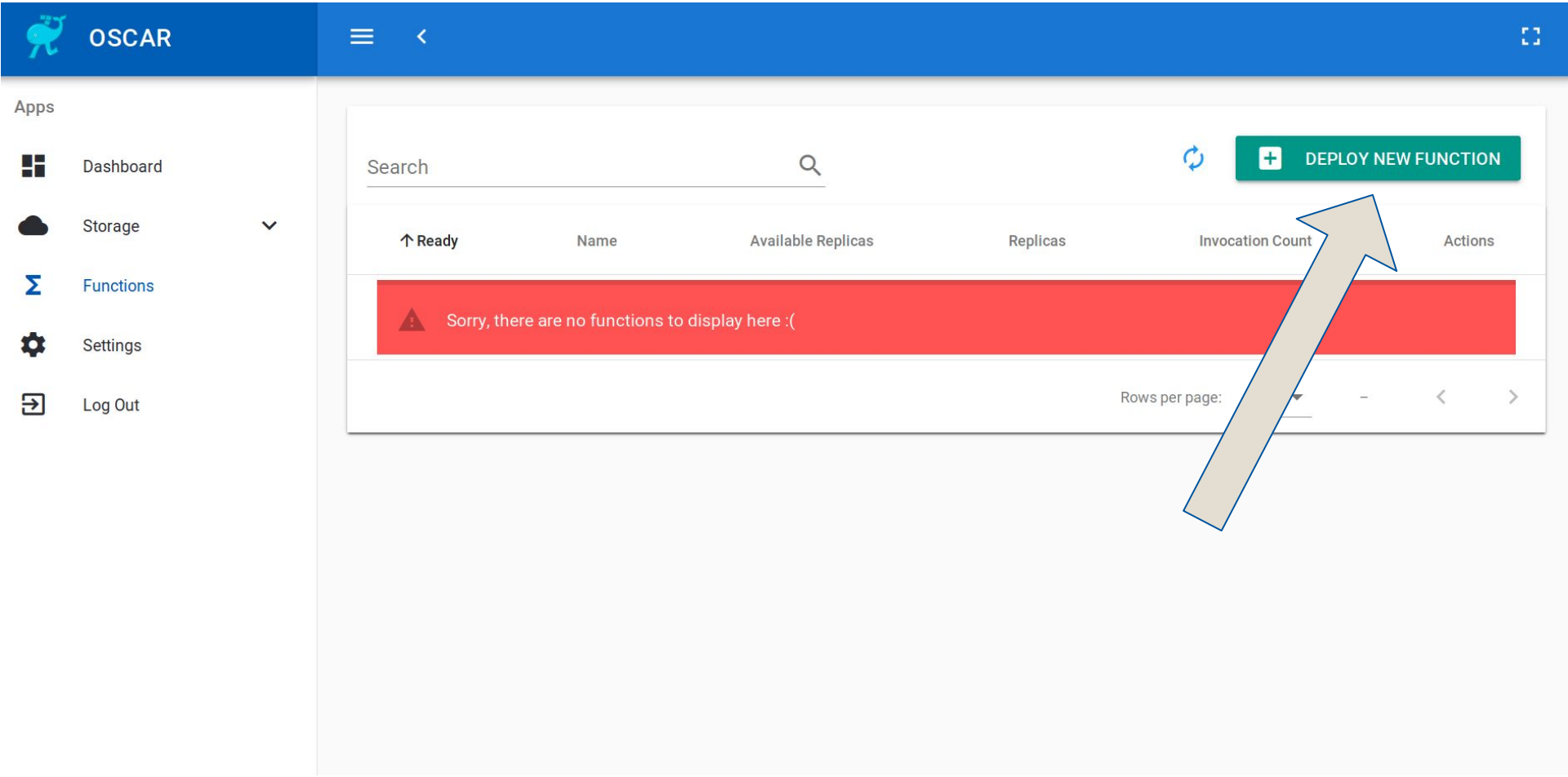
OSCAR user interface



USE CASE 1: Plant Classification

GRyCAP
Grid y Computación de Altas Prestaciones
www.grycap.upv.es

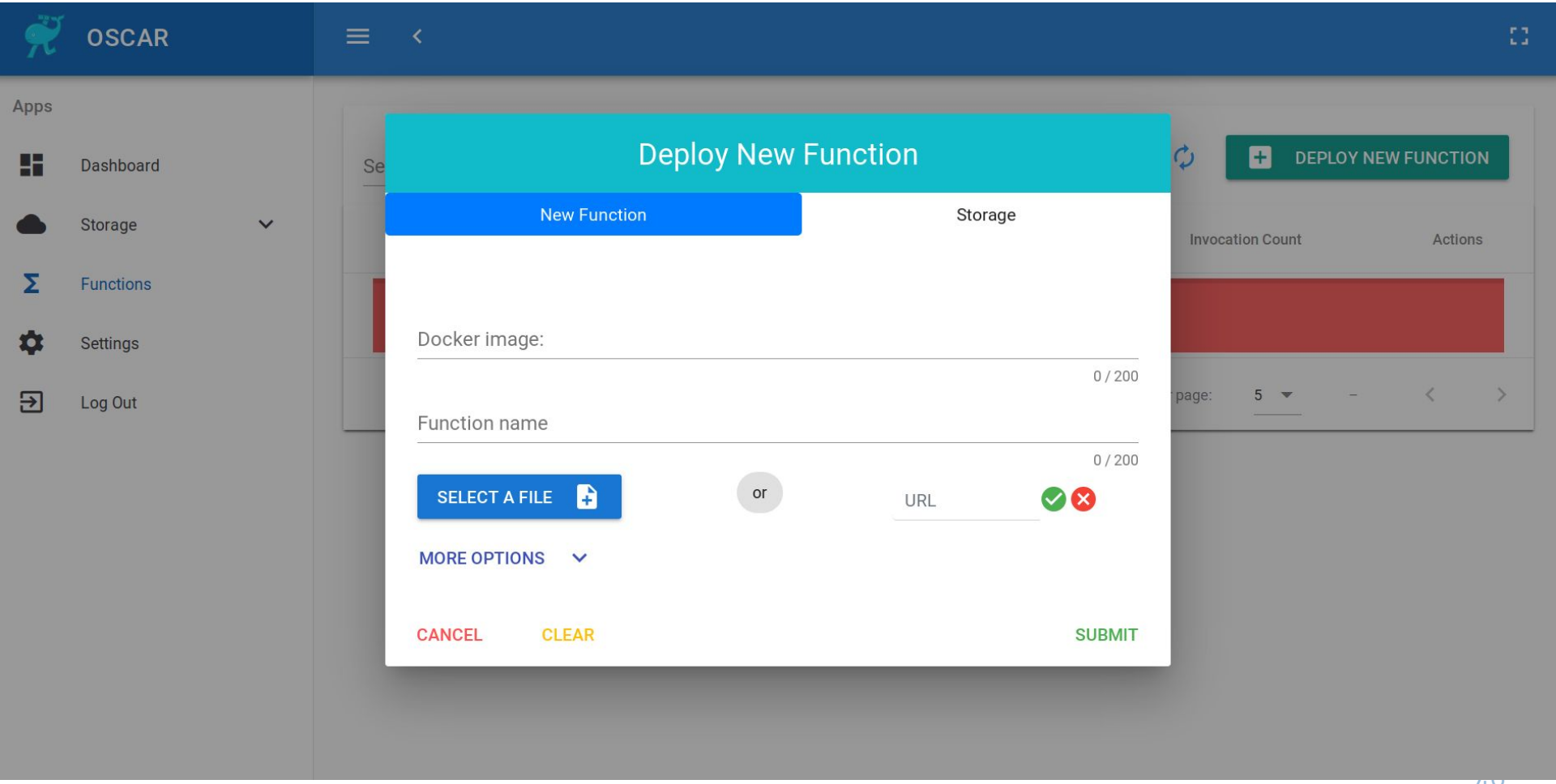
Create new function



USE CASE 1: Plant Classification

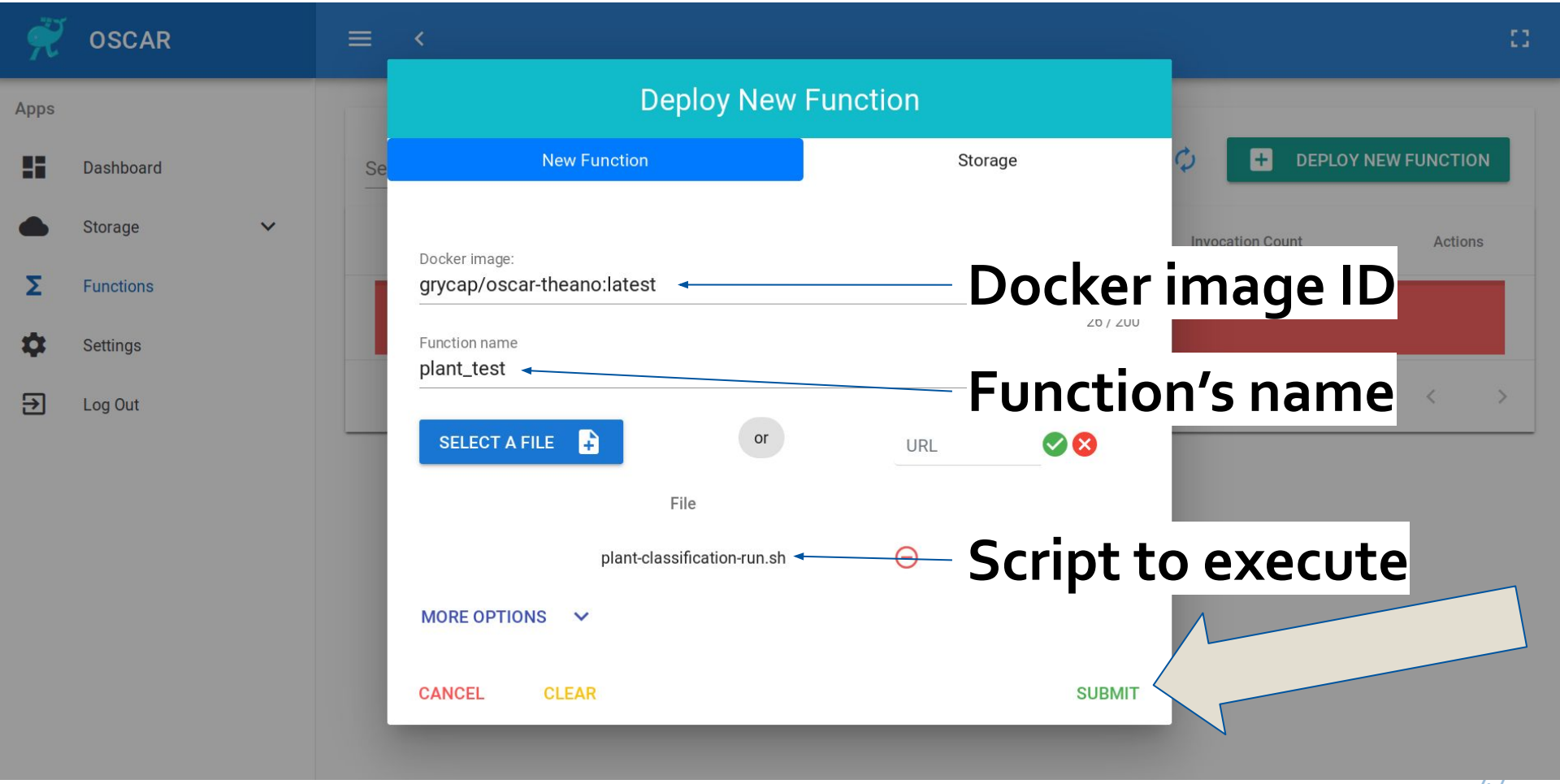
GRyCAP
Grid y Computación de Altas Prestaciones
www.grycap.upv.es

Creating the function



USE CASE 1: Plant Classification

Creating the function



USE CASE 1: Plant Classification

Uploading the file to process

OSCAR

Apps

Dashboard

Storage

plants-in

plants-out

CREATE BUCKET

Functions

Settings

Log Out

Bucket **plants-in**

SELECT FILES

UPLOAD

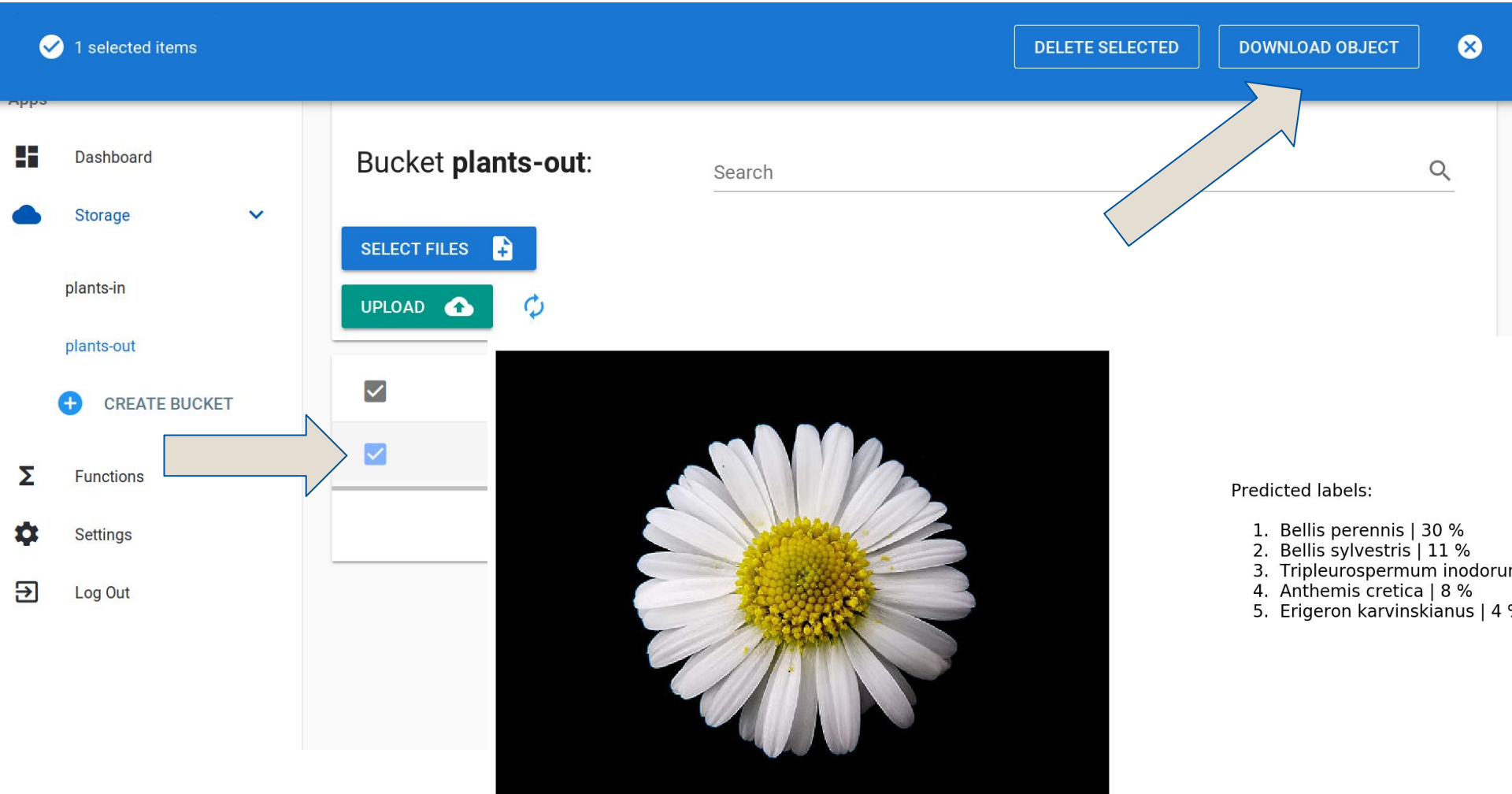
Name			Actions
daisy.jpg	37.63 KB	2019-09-04 15:02	

Rows per page: 5 1-1 of 1

USE CASE 1: Plant Classification

GRyCAP
Grid y Computación de Altas Prestaciones
www.grycap.upv.es

Downloading the output



USE CASE 1: Plant Classification

Conclusions:

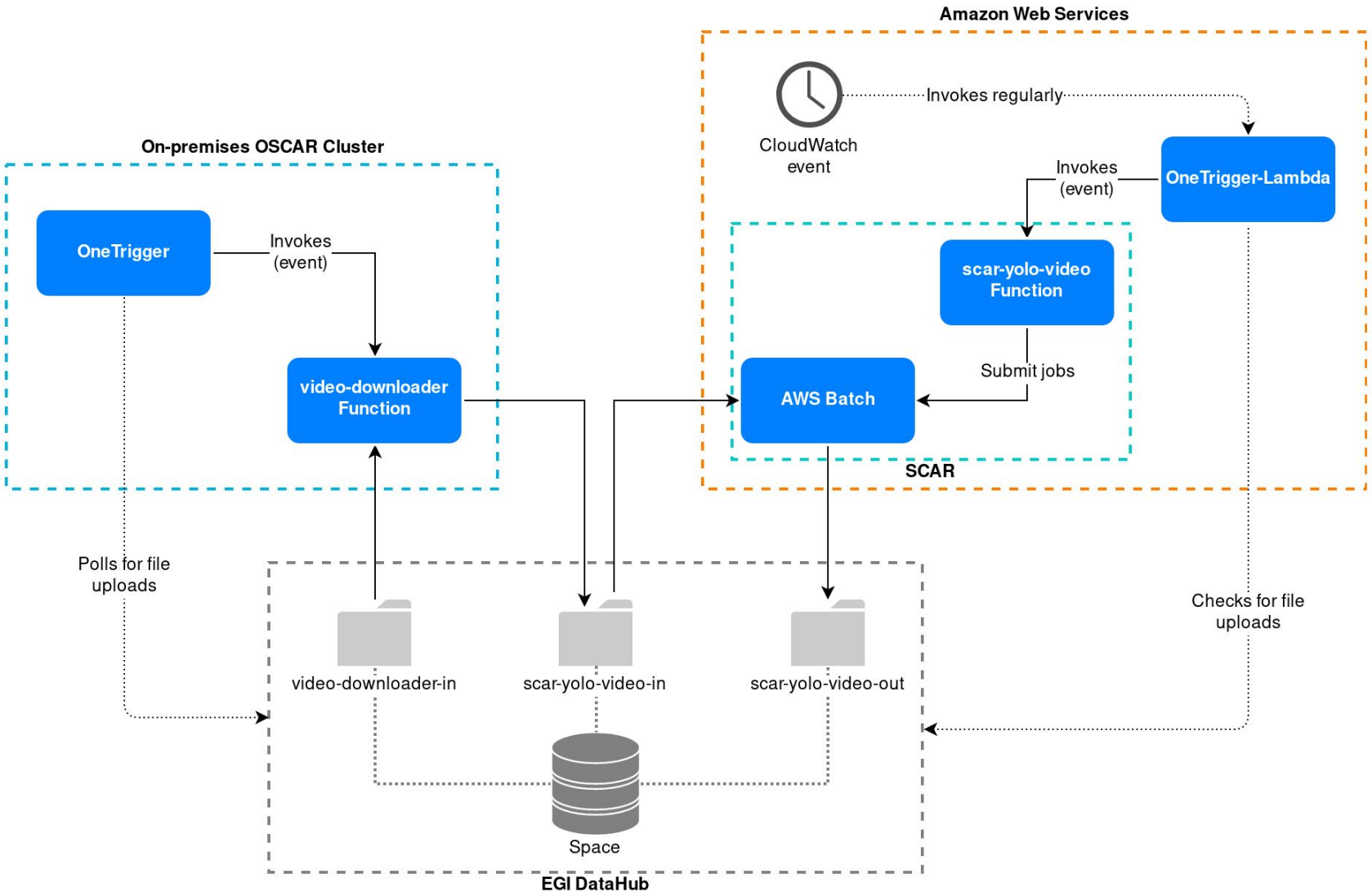
- Execute container image stored in Docker Hub
- Execute customized user script inside container
- Automatically create and link Minio buckets
- Launch image processing through the user interface
- On-premises cluster with two levels of elasticity
  - Pod Level: managed by kubernetes
  - Node Level: managed by CLUES

USE CASE 2

Multi-cloud workflow for video processing:

- Combining on-premises, public and federated Cloud resources.
- OSCAR and SCAR function composition using EGI DataHub as a storage provider.
- Download a bunch of videos inside a **Onedata space** (EGI DataHub) from a text file in an **on-premises or federated Serverless platform** (OSCAR) and process them in **AWS Batch** leveraging its accelerated computing capability (SCAR).

USE CASE 2

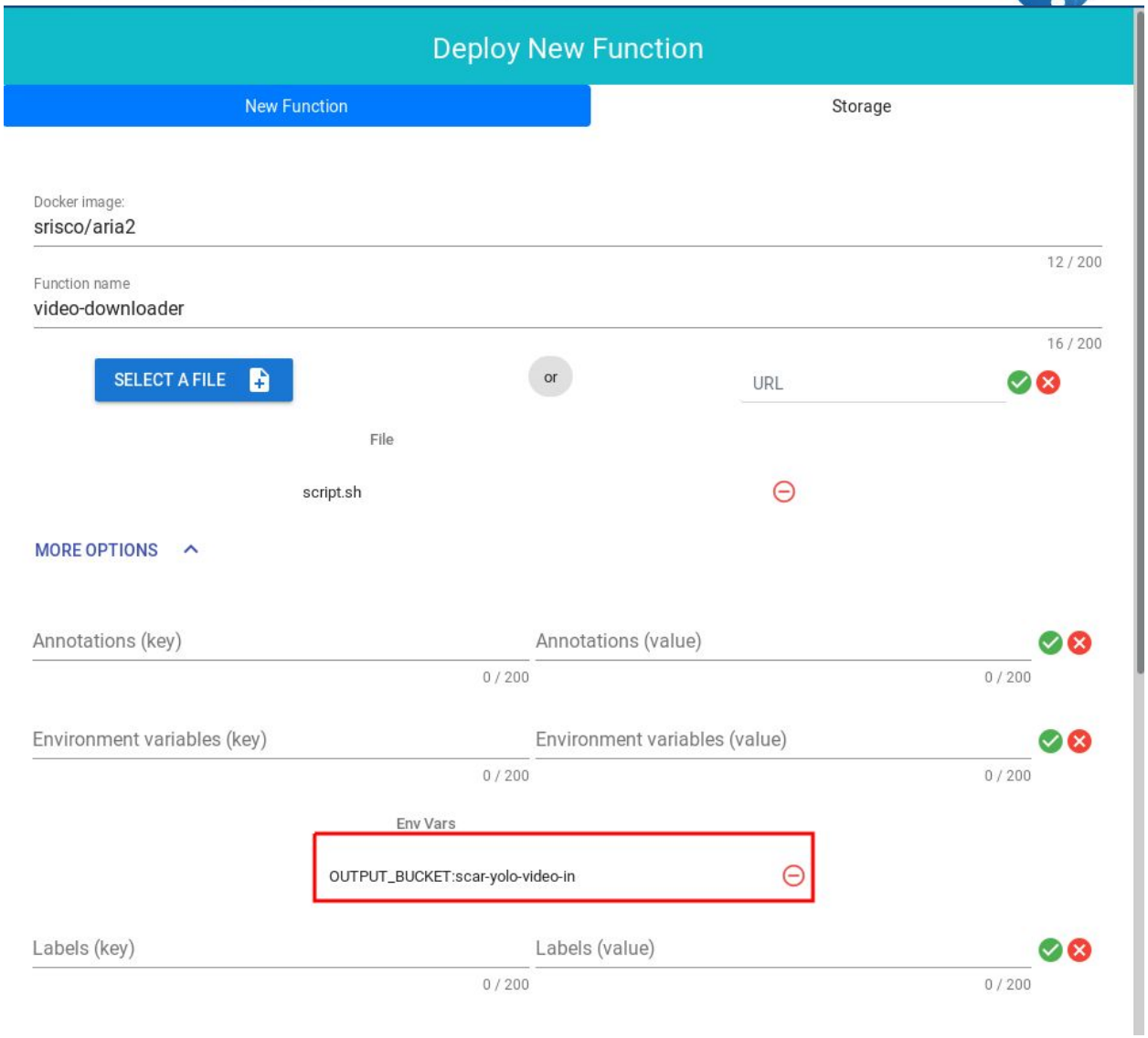

Amazon Web Services
CloudWatch event
Invokes regularly
OneTrigger-Lambda
Invokes (event)
scar-yolo-video Function
Submit jobs
AWS Batch
SCAR
On-premises OSCAR Cluster
OneTrigger
Invokes (event)
video-downloader Function
Polls for file uploads
Checks for file uploads
video-downloader-in
scar-yolo-video-in
scar-yolo-video-out
Space
EGI DataHub

USE CASE 2

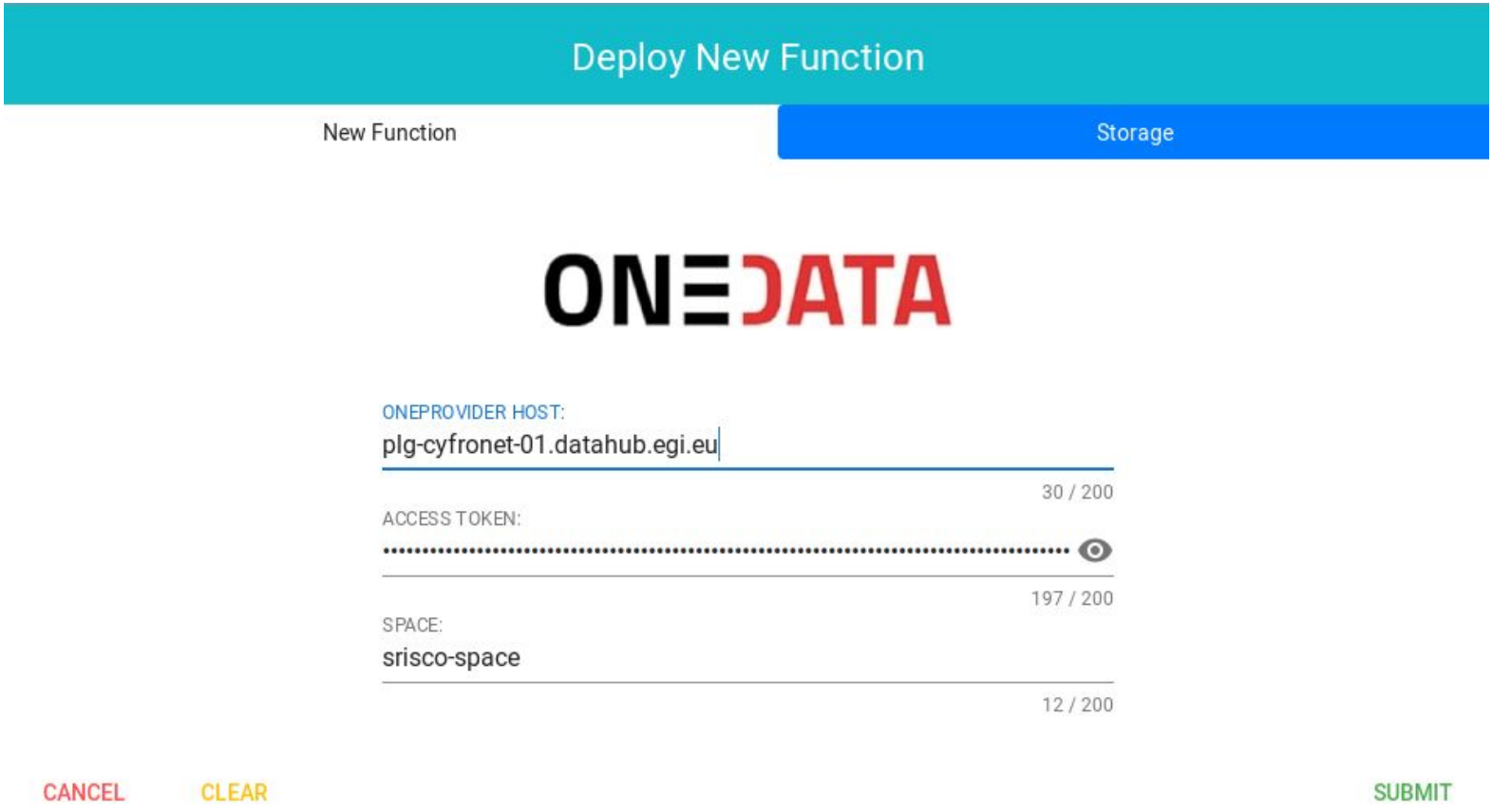
Function 1: **video-downloader**

- Using a Docker image with the *aria2* download utility.
- Users upload a text file with a list of URLs in the input folder and the function download the content in the output folder.

USE CASE 2


Deploy New Function
New Function
Storage
Docker image:
srisco/aria2
12 / 200
Function name
video-downloader
16 / 200
SELECT A FILE
or
URL
File
script.sh
MORE OPTIONS
Annotations (key)
Annotations (value)
0 / 200
0 / 200
Environment variables (key)
Environment variables (value)
0 / 200
0 / 200
Env Vars
OUTPUT_BUCKET:scar-yolo-video-in
Labels (key)
Labels (value)
0 / 200
0 / 200

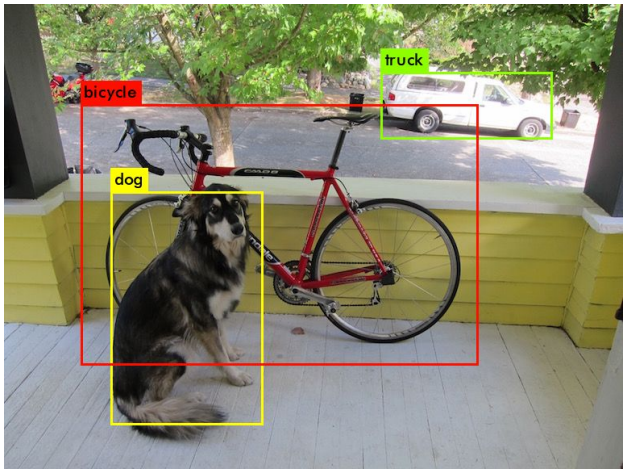
USE CASE 2


Deploy New Function
New Function
Storage
ONEDATA
ONEPROVIDER HOST:
plg-cyfronet-01.datahub.egi.eu
30 / 200
ACCESS TOKEN:
197 / 200
SPACE:
srisco-space
12 / 200
CANCEL
CLEAR
SUBMIT

USE CASE 2

Function 2: **scar-yolo-video**

- Uses YOLO, a real-time object detection system.
- Supports CPU and GPU computation to process images or videos.
- With SCAR can be deployed in batch-mode in order to support GPU-based computing to dramatically accelerate object recognition.



USE CASE 2


```yaml
functions:
  scar-yolo-video:
    image: srisco/yolov3:opencv-cudnn
    init_script: user-script.sh
    s3:
      input_bucket: scar-yolo-video
    api_gateway:
      name: scar-yolo-video
    execution_mode: batch
    batch:
      enable_gpu: true
      compute_resources:
        max_v_cpus: 4
        instance_types:
        - p2.xlarge
    lambda_environment:
      STORAGE_AUTH_ONEDATA_SPACE_1: my-onedata-space
      STORAGE_AUTH_ONEDATA_HOST_1: plg-cyfronet-01.datahub.egi.eu
      STORAGE_AUTH_ONEDATA_TOKEN_1: my-secret-token
      STORAGE_PATH_INPUT_1: scar-yolo-video-in
      STORAGE_PATH_OUTPUT_1: scar-yolo-video-out
```

USE CASE 2

OneTrigger-Lambda

- Check for file uploads in a Onedata space.
- Needed to automatically invoke the *scar-yolo-video* function.
- Can be deployed as a Serverless function using AWS Lambda and CloudWatch Events.

CONCLUSIONS

- Event-driven computing allows to perform computing in response to events (such as file uploads) on a serverless platform which provides automated elasticity for dynamic resource provisioning.
- SCAR allows to execute generic applications (that fit) on AWS Lambda and automated extension to AWS Batch (for those apps that do not fit in AWS Lambda) to allow elastic CPU/GPA batch computing on the Cloud.
- OSCAR implements the event-driven computing model of SCAR in on-premises Clouds, integrated with EGI services (EGI DataHub and EGI Federated Cloud).

CONTACT & ACKNOWLEDGEMENTS

Germán Moltó - gmolto@dsic.upv.es
Sebastián Risco - serisgal@i3m.upv.es
Alfonso Pérez - alpegon3@upv.es
Miguel Caballer - micafer1@upv.es
Diana María Naranjo - dnaranjo@i3m.upv.es
Instituto de Instrumentación para Imagen Molecular
Universitat Politècnica de València
Camino de Vera s/n
46022, Valencia
SPAIN

The authors would like to thank the Spanish "Ministerio de Economía, Industria y Competitividad" for the project "BigCLOE" with reference number TIN2016-79951-R.

This work has been partially funded through the EGI Strategic & Innovation Fund.