CLASS

COORDINATING EDGE AND CLOUD FOR BIG DATA ANALYTICS

Deployment and lifecycle management of containerized applications with Rotterdam

Rut Palmero – Project Director,

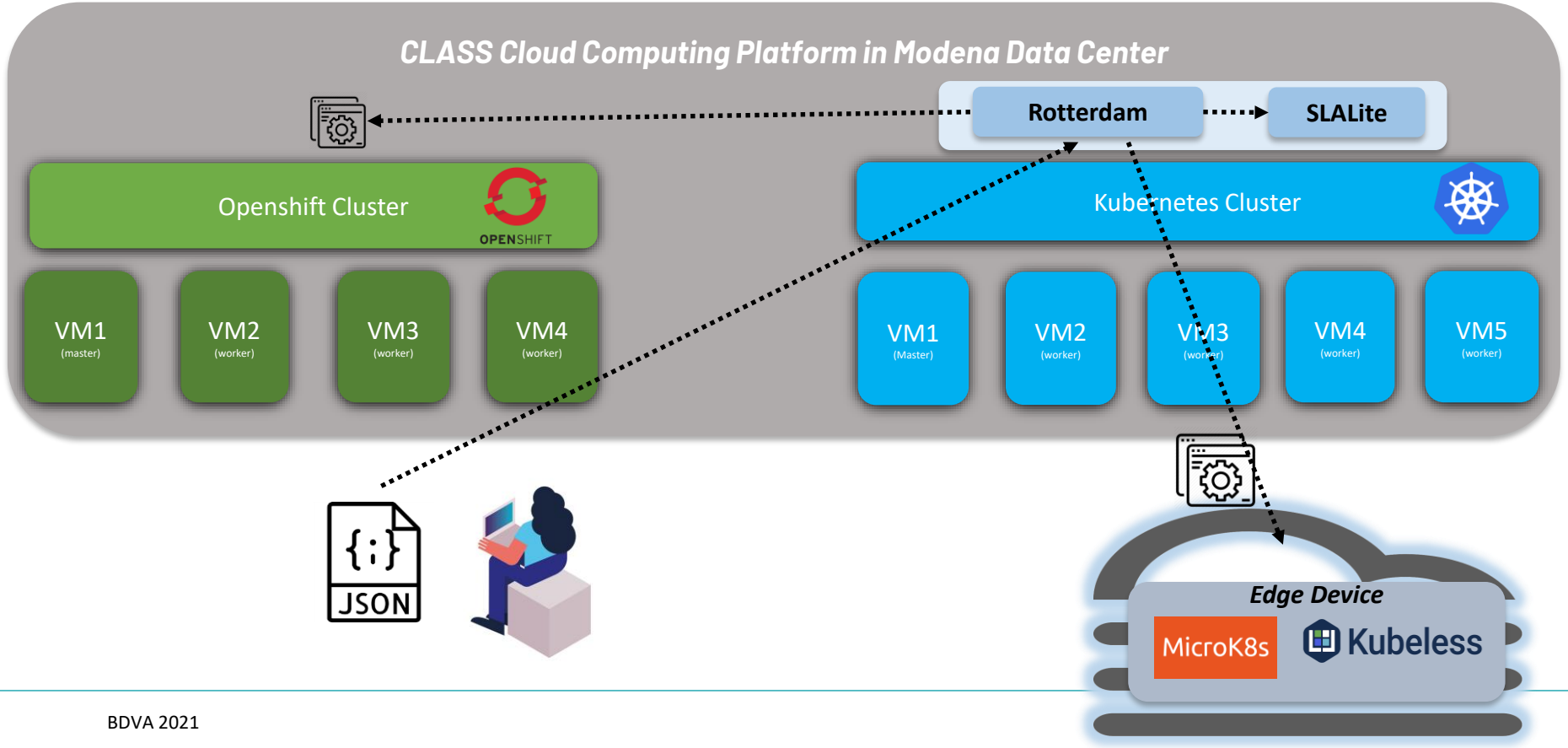
Atos Research and Innovation

The CLASS project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 780622

Topics

- CLASS Computing continuum: Rotterdam CaaS
- SLALite: QoS Enforcement
- SLA Predictor: Time guarantees

CLASS computing continuum



Rotterdam Features

- Rotterdam is a CaaS that allows the deployment and management of containerized applications from the Modena Data Center k8s cluster, including the management of COMPSs workflows.
- It supports the management of multiple infrastructures (K8s, Openshift, MicroK8s and Kubeless) and applications running on them from a simple Rotterdam instance (cloud-to-edge).
- Thanks to the SLALite, it provides applications scalability and elasticity triggered by QoS rules defined by the user.
- Single JSON format for describing general applications, COMPSs workflows and serverless functions definitions.
- Simple Deployment of MicroK8s and applications in additional Edge or Cloud devices through Rotterdam REST API.

Rotterdam REST API

swagger ./swagger.json Explore

Rotterdam CaaS 1.3.3

[Base URL: rotterdam-caas.192.168.7.28.xip.io/api/v1]
./swagger.json

Rotterdam CaaS REST API is responsible for the deployment of tasks and docks in a Kubernetes cluster
Find out more about the CLASS project

Schemes
HTTP

Authorize

status Information about the status and configuration of Rotterdam

task Rotterdam Tasks: creation, deletion and management

- GET `/tasks/{id}` Gets a Rotterdam Task
- DELETE `/tasks/{id}` Deletes a Rotterdam Task
- GET `/tasks/{id}/all` Gets a Rotterdam Task, including deployment info
- GET `/tasks` Returns all the current Rotterdam tasks (from all infrastructures / clusters)
- POST `/tasks` Creates a new Rotterdam Task

```
{
"name": " compss-workflow ",
"replicas": 2,
"type": " app-compss ",
"qos": [{"qosid": " DeadlinesMissed_1 "}],
"image": " example/compss-test ",
"ports": [44240]
}
```

COMPS Workflows (data analytics workloads)

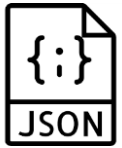
```
{
 "name": "helloworld1",
 "type": "function",
 "qos": [{"qosid": "KubeletTooManyPods"}],
 "runtime": "python3.7",
 "functionType": "url+zip",
 "function":
"https://github.com/kubeless/kubeless/blob/master/examples/nodejs/helloFunctions.zip}
```

Serverless functions

SLALite

```
{
"name": " compss-workflow ",
"replicas": 4,
"type": " app-compss ",
"qos": [{"qosid": " DeadlinesMissed_1 "}],
"image": " example/compss-test ",
"ports": [44240]
}
```

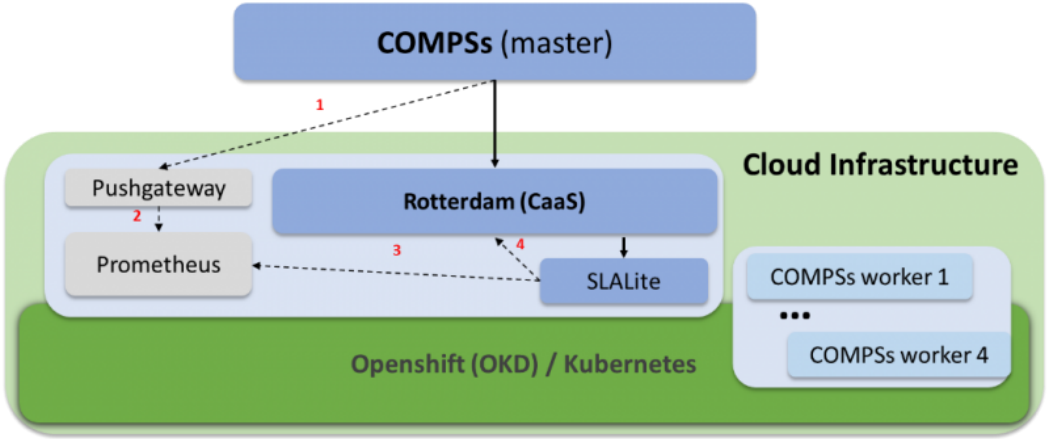
Rotterdam JSON

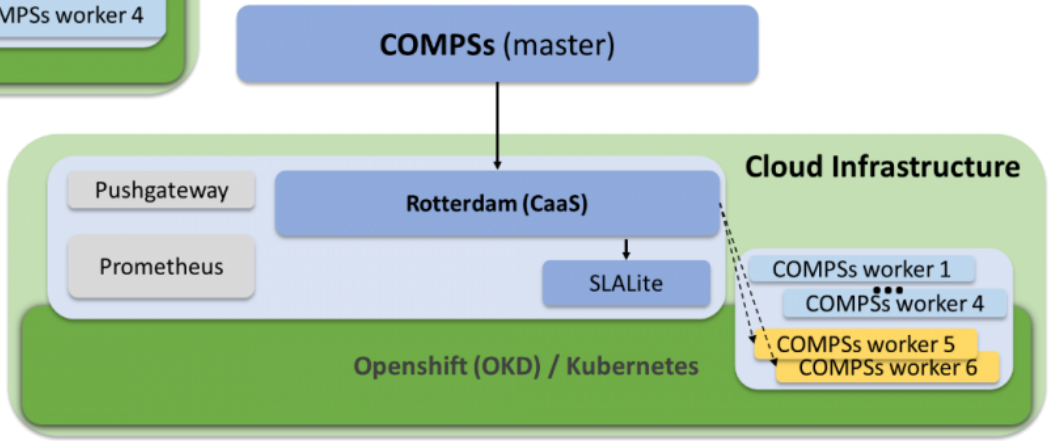



```
{"type": "app-compss",
   "guaranteeName": "DeadlinesMissed_1",
   "maxAllowed": 0,
   "action": "scale_out",
   "scaleFactor": 1.5,
   "guarantees": [{
      "name": "deadlines_missed",
         "constraint": "deadlines_missed < 1"}]
}
```

SLA Agreement

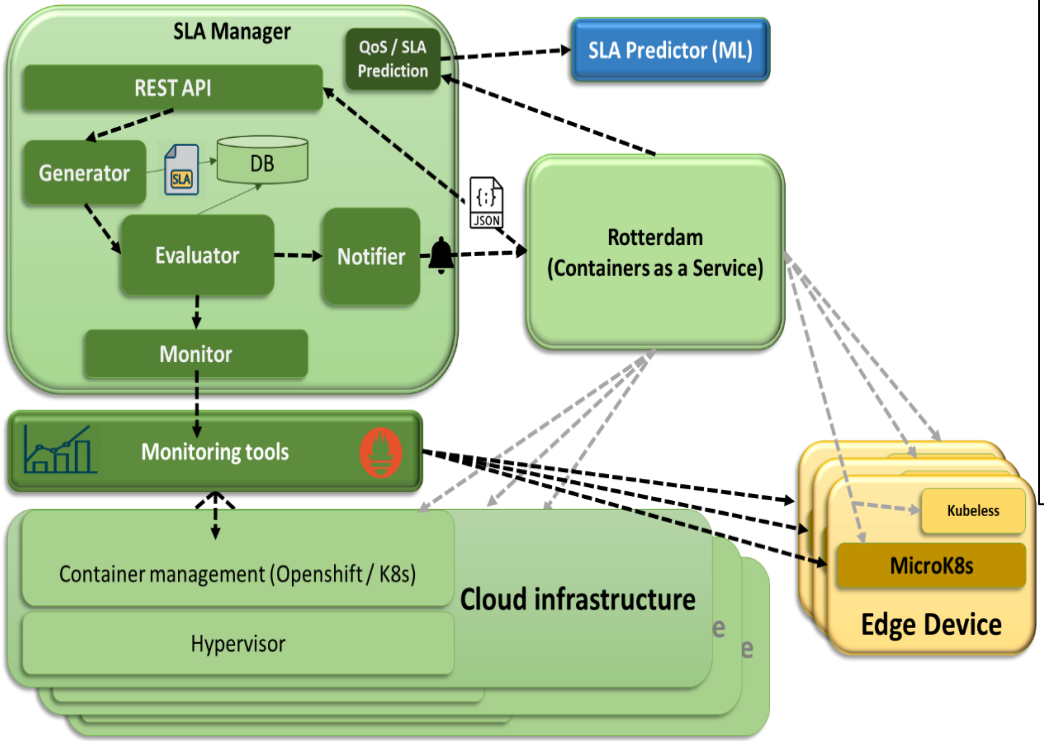
Scalability overhead problem





Time Guarantees from SLA Predictor

Rotterdam + SLA Predictor




```
{"name": " compss-workflow ",
"replicas": 4,
"type": " app-compss ",
"qos": [{"qosid": "exectime", "metric":
"execution_time", "comparator": "<",
"value": 520000}],
"image": " example/compss-test ",
"ports": [44240]
}
```


/predictSLA?workers=4&exectime=520000

Output: 6

SLA Predictor Design

- First, classify how stressed is our system by analysing the metrics data extracted in the Exploratory Data Analysis (EDA). It can be classified in 3 classes: low, normal, or high.
- SLA Predictor has been exposed as a microservice to be accessible using REST calls. It return an estimated execution time for that level of stress and the number of workers.
  - It performs a query to Prometheus to retrieve the last hour of *our* selected metrics (EDA), classify it to know how much the system is stressed, and return the most similar number of workers that fits the input data based on the rules defined for "stress→workers→execution time".

Workers	Stress	Execution time
1	Low	(550000,580000)
1	Normal	(580001,610000)
1	High	(610001,1000000)
3	Low	(550000,570000)
3	Normal	(570001,600000)
3	High	(600001,1000000)
6	Low	(430000,450000)
6	Normal	(450001,470000)
6	High	(470001,1000000)
9	Low	(390000,410000)
9	Normal	(410001,420000)
9	High	(420001,1000000)

Table: Rules definition for linking workers, stress, and execution time

CLASS Cloud-to-edge continuum - Summary

- Deployment and lifecycle management of containerized applications:
  - Multiple infrastructures: Microk8s, K8s, Kubeless… in cloud or edge.
  - REST API and a single JSON format for describing all applications.
  - Applications scalability and elasticity triggered by QoS rules defined by the user.
- Rotterdam with Time guarantee with SLA Predictor
  - Applications scalability and elasticity triggered by informed QoS rules (from the ones provided by the user).
  - Initial EDA process to classify how stressed is our system by analysing the metrics data that better describes it.

CLASS

www.class-project.eu
Twitter: **@EU_CLASS**
LinkedIn: **linkedin.com/company/classproject**