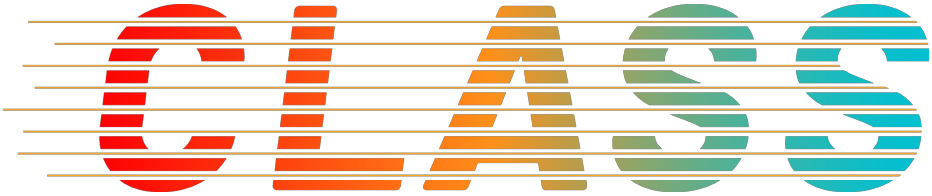

D4.4 First release of the Cloud Data Analytics Service Scalability components

Version 1.0

Document Information

Contract Number	780622
Project Website	https://class-project.eu/
Contractual Deadline	M15, March 2019
Dissemination Level	PU
Nature	DEM
Author(s)	İlknur Chulani (ATOS) Roi Sucasas (ATOS) Orlando Avila (ATOS)
Contributor(s)	
Reviewer(s)	Erez Hadad (IBM)
Keywords	Cloud, WP4, Rotterdam, Container-as-a-Service (CaaS), Docker, Kubernetes, container orchestration, data analytics service scalability, data analytics service management

Notices: *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No "780622".*

Change Log

Version	Author	Description of Change
V0.1	Ilknur Chulani, Roi Sucasas, Orlando Avila (ATOS)	Initial Draft
V0.2	Erez Hadad (IBM)	Deliverable Review
V0.3	Ilknur Chulani (ATOS)	Final version
V1.0	Eduardo Quiñones (BSC)	Ready for EC submission

Table of contents

1 Executive Summary

This document presents the deliverable "D4.4: First release of the Cloud Data Analytics Service Scalability components" for the CLASS project. It is a publicly available summary of "D4.2: First release of the Cloud Data Analytics Service Management components", which is a confidential and more comprehensive report provided by the same work package (WP4). Therefore, this report includes a subset of the content from D4.2 [2], removing some implementation details due to commercial interests and project specific deployment information due to security concerns.

Both deliverables present the cloud standalone environment provided by WP4 Cloud Computing, and specifically the data analytics service management and scalability components developed by ATOS in tasks 4.2. Data Analytics Service Management and 4.3. Data Analytics Service Scalability between months M7-M15.

This first release of the CLASS standalone cloud environment provides an initial version of simplified life-cycle management of data analytic workloads, soft real-time guarantees, performance monitoring of data analytics workloads and data analytics service scalability features for the data analytics service developers.

The cloud environment includes two layers; first a Container-as-a-Service (CaaS) layer called Rotterdam which is developed by ATOS, and Cloud infrastructure layer that is required by Rotterdam to deploy and operate containerized analytics tasks.

Rotterdam provides a REST API interface for deploying and managing data analytics services and can be installed using a Docker [5] image.

A first version of the CLASS cloud standalone environment for data analytics service management and scalability has been deployed in the City of Modena Data Center, and a preliminary demonstration was made to the project team in Haifa F2F meeting.

The progress in this milestone will pave the way for the next phase where the goal will be providing a cloud environment coordinated with the edge layer (WP4), and an analytics scheduler (WP5).

2 Introduction

One of the CLASS project's goals is to provide a cloud computing environment which allows data analytics service developers to focus on the task at hand, and not to worry about cloud computing specific details. This deliverable presents the cloud standalone environment provided for this purpose by WP4 Cloud Computing, and specifically the data analytics service management and scalability components developed by ATOS in tasks 4.2. Data Analytics Service Management and 4.3. Data Analytics Service Scalability between months M7-M15.

This deliverable is linked closely with two other WP4 deliverables; namely "D4.1 – Cloud Requirement Specification and Definition" [1] which provided the specifications for the CLASS cloud computing platform presented in this deliverable, and "D4.2: First release of the Cloud Data Analytics Service management components" which is a confidential and more comprehensive superset of this report. Therefore, most of the content from D4.2 [2] has been replicated here but implementation details have been excluded due to commercial interests. Also project specific deployment details have been removed to avoid any security issues.

The rest of the document is structured as follows:

- Section 3 describes the features and capabilities of the data analytics service management and scalability components.
- Section 4 provides a technical overview on the architecture and the interfaces provided.
- Section 5 presents information on installation and usage.
- Section 6 mentions the CLASS cloud standalone environment deployment in the City of Modena Data Center and the demonstration made in the F2F project meeting in Haifa.

3 Functional Description

This first release of the CLASS standalone cloud environment provides initial implementations of some key service management and scalability features for the data analytics service developers:

- Simplified life-cycle management of data analytic workloads
- Soft real-time guarantees
- Performance monitoring of data analytics workloads
- Data Analytics service scalability

These features have originally been described in D4.1 [1], so some text from that deliverable has been included in the next paragraphs within this section along with new content, for the sake of completeness. Please note that this release includes a preliminary implementation of these features; feature-complete versions will be provided in the next release.

Simplified life-cycle management of data analytic workloads

The main capability of the cloud computing environment for CLASS is the simplified management of the life-cycle of data analytics resources (e.g. VMs, clusters, containers, storage) by providing tools that abstract the details of cloud infrastructures.

So, for instance, to deploy a data analytics service, the analytics developers would just need to take the following steps:

- Build a container image for a single analytic task or workload.
- Specify the CPU and memory requirements.
- Specify the Quality of Service (QoS) constraints.
- Define networking and access (security) policies.

They would not need to be concerned with:

- VMs, servers, clusters, or any other peculiarities of the infrastructure, or the middleware around it.

Soft real-time guarantees

Another key capability is QoS (quality-of-service) regarding timeliness of the analytics tasks; the service provider can specify time constraints (such as throughput and latency) upon the analytics services to be deployed and operated in the cloud. These are considered as SLAs between the service provider and the consumer, and ensures analytics results are served at the right time (i.e. timely) to the consuming decision-making processes.

The platform helps achieve soft real time guarantees by constant monitoring of QoS parameters.

Performance monitoring of data analytics workloads

The platform integrates with well-known frameworks like Grafana [6] and Prometheus [7] to make monitoring data from the distributed computing platform available.

Data Analytics service scalability

The platform also provides service scalability features, i.e. dynamic adaptation and orchestration of cloud resources to meet QoS requirements for the data analytics services deployed in the cloud. A series of quality parameters and specific scalability policies can be defined (and optimized) by the cloud computing platform provider. These parameters include both infrastructure related metrics such as CPU, memory and storage; and specific time performance metrics for the data analytics service, such as throughput and latency. The adaptation actions are transparent to the user of the cloud computing platform as it aims to provide a self-managed self-scaling environment.

It is worth noting that the autoscaling features of the underlying container orchestration platforms such as Kubernetes [3], or related tools such as Istio [4] are not used by this implementation, since in the case of CLASS, self-scaling needs to be done with real-time QoS constraints in mind.

4 Technical description

4.1 Architecture Overview

As previously described in D4.1 [1], and depicted in Figure 1 below, Cloud standalone environment for Data Analytics Service Management and Scalability includes two layers:

Rotterdam is a Container-as-a-Service (CaaS) layer developed by ATOS to manage the life cycle and scalability of cloud data analytics workloads. It abstracts the cloud infrastructure details away from data analytics service developers, which requires not only deployment but also guaranteed performance. By means of the unifying power of Kubernetes and container abstraction, it provides a unified mechanism and fully managed service for the life-cycle management and performance optimization of workloads.

Cloud infrastructure layer is required by Rotterdam to deploy and operate containerized analytics tasks. This infrastructure can go from native cloud (i.e., based on Docker [5] containers) to IaaS providing either a private or public cloud, including "traditional" virtualization solutions in data centers. Thus, Rotterdam aims to abstract away the details of all those infrastructure providers.

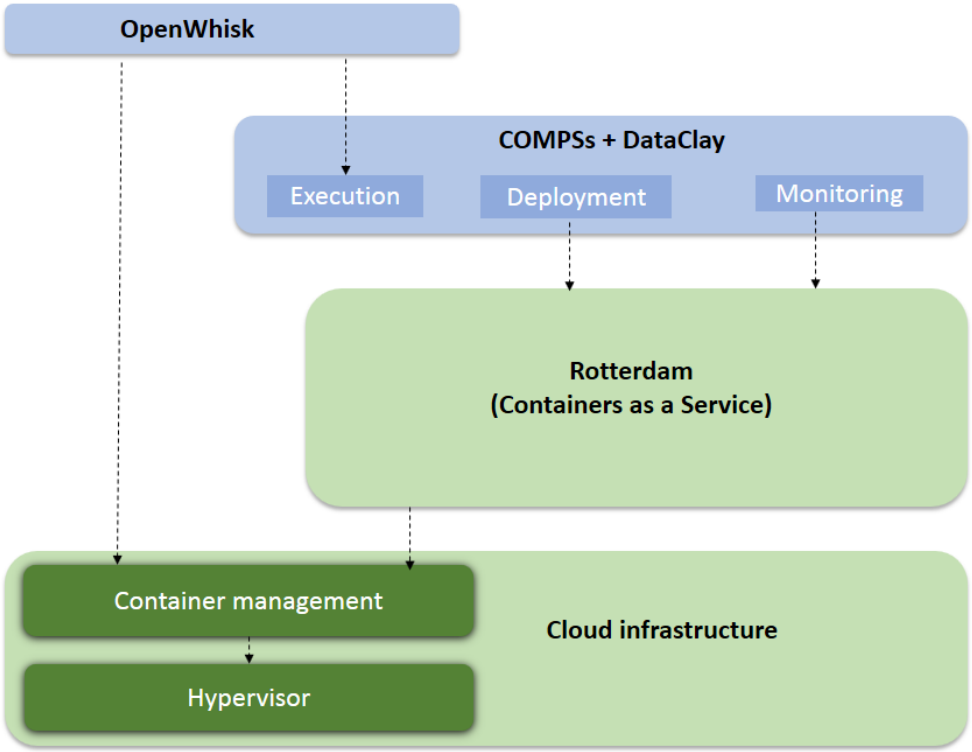

Figure 1: CLASS Cloud computing platform shown in green, in the context of the overall CLASS architecture.

Rotterdam internally includes four sub-components to provide Data Analytics Service Management and Scalability features:

- A **CaaS API gateway** for providing REST APIs to developers for deploying data analytics tasks easily
- A **Deployment Engine** for optimal placement of data analytics services on cloud resources
- An **SLA Manager** for monitoring and enforcing real time QoS parameters
- An **Adaptation Engine** for self-managed and elastic scalability actions based on SLA Manager's inputs.

The **cloud infrastructure layer** is based on the following technologies:

- **Docker** [5] for container technology
- **OpenShift** [8] flavor of Kubernetes for Container Management
- **VMware** [9] as Hypervisor

4.2 Interfaces Provided

Rotterdam exposes a REST API interface to other users and applications to manage the lifecycle of applications or tasks in a Kubernetes cluster. A Rotterdam task includes all the Kubernetes elements involved in the execution of an application, like the service, routes, deployments (including the pods), volumes, etc. And the dock is the corresponding namespace in a Kubernetes cluster. In its current state, Rotterdam implements the following main operations:

- Create a new task in a specific dock.

Method	URI
POST	/docks/<dock>/tasks

- Get the status of a task, including the description and other information from the corresponding Kubernetes elements.

Method	URI
GET	/docks/<dock>/tasks/<name>

- Remove a task

Method	URI
DELETE	/docks/<dock>/tasks/<name>

Rotterdam also includes other operations for getting information about the pods (execution units from Kubernetes) executed by a specific task, and a list of all the active tasks managed by Rotterdam:

Method	URI	Description
GET	/docks /tasks/	*Get all tasks managed by Rotterdam*
GET	docks/<dock>/tasks/<name>/containers	*Get the list of containers of a given task.*

Finally, there are other internal interfaces used to connect the different subcomponents involved in the Rotterdam functionalities, like the SLA Manager. The following method gets requests from the SLA Manager and redirect them to the Scalability Management component, also included in the Rotterdam application:

Method	URI
POST	/rules-engine/docks/<dock>/tasks/<name>/sla

5 Installation and Usage

5.1 Installation

This section gives an overview of the installation process for Rotterdam; complete steps can be found in D4.2 [2], which can be accessed by making a request to the CLASS project team.

The Rotterdam prototype is composed of two main builds. The first one is what we call in this section Rotterdam, and it includes the CaaS API Gateway, the Deployment Engine and the Adaptation Engine. The other build contains the SLA Manager.

Both builds are provided as Docker images, so they only require a Docker or Kubernetes environment to be executed. The images can be found in the following URLs:

https://cloud.docker.com/u/atosclass/repository/docker/atosclass/rotterdam-caas

https://cloud.docker.com/u/atosclass/repository/docker/atosclass/slalite

Rotterdam Docker image (***atosclass/rotterdam-caas:0.0.9.4***) can be installed in different ways depending on the execution environment. This prototype was installed and tested successfully in Docker, Kubernetes and OKD (Openshift Kubernetes) [10].

After the installation, Rotterdam should be accessible from the following URL or similar: http://rotterdam-caas.*SERVER-IP-ADDRESS*.xip.io

The SLA Manager docker image (***atosclass/rotterdam-slalite:0.0.1)*** can be installed via docker commands or via the Kubernetes / Openshift web user interface.

The SLA Manager takes part in the scalability management process, needs to have access to Prometheus, and it also needs to have access to the CaaS API Gateway component. In this first prototype some manual steps are required. Finally, the best way to install all third-party applications, like Grafana or Prometheus, is to use the Openshift OKD user interface, which facilitates not only the installation of these components but also the configuration.

The resulting URL for the SLA Manager application is the following (or similar): http://rotterdam-slalite.*SERVER-IP-ADDRESS*.xip.io

5.2 Usage

Rotterdam also includes a visual user interface based on **swagger** [11], which should be accessible from "http://rotterdam-caas.*SERVER-IP-ADDRESS*.xip.io/swaggerui/".

The following picture shows this user interface:

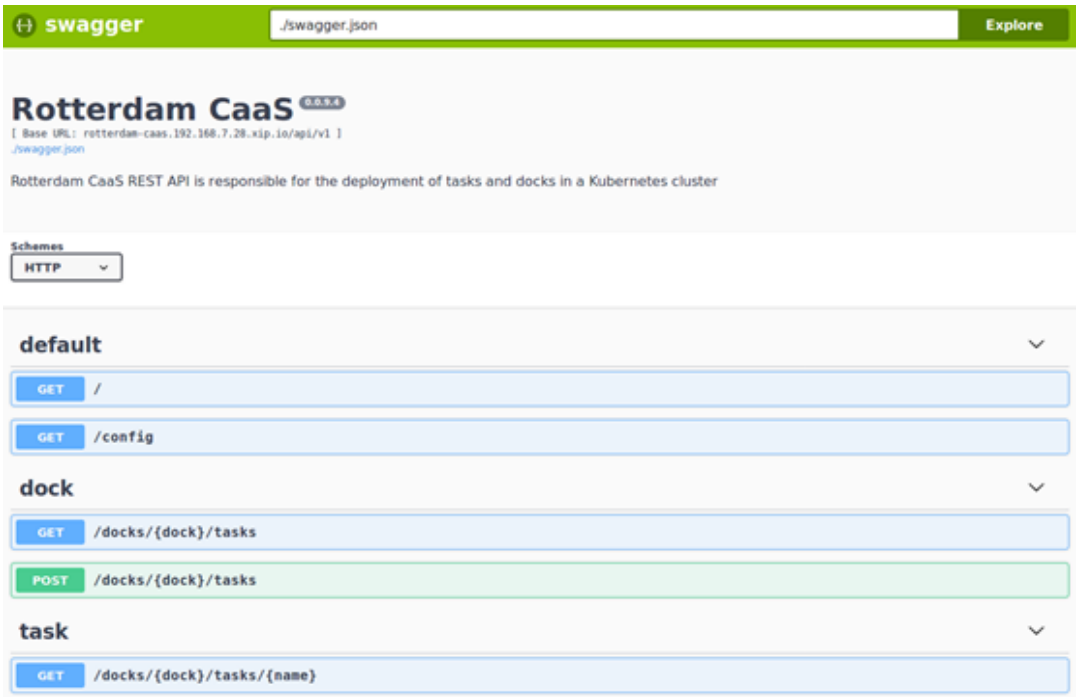

Figure 2: Rotterdam (swagger) UI

From this window, a user can perform all the deployment and management operations: deployment, management and termination of one or more tasks.

Deployment of a task:

Request: *POST* http://rotterdam-caas.*IP-ADDRESS*.xip.io/api/v1/docks/class/tasks

Request Body:

```
{
 "name": "my-application",
 "qos": {},
 "containers": [
          {
   "name": "my- application",
   "image": "helloworld",
   "essential": true,
   "ports": [],
   "volumes": {},
   "environment": []
          }
 ]
}
```

Response:

```
{
 "response": "ok",
 "message": "task my- application' created ['class']",
 "content": {
  "name": "my- application",
  "dock": "class",
  "status": {
   "desc": "deploying",
   "replicas": 0,
   "readyReplicas": 0
  },
          "urls": "http://my- application.IP-ADDRESS.xip.io",
          "created": "2019-03-14T14:08:19Z"
 }
}
```

The response includes the URL where the application is being exposed. Replicas is 0 because the application is still being deployed.

Get the status of a task:

Request: *GET* http://rotterdam-caas.*IP-ADDRESS*.xip.io/api/v1/docks/class/tasks/my-application

Response:

```
{
 "response": "ok",
 "content": {
  "name": "my-nginx",
  "dock": "class",
  "status": {
   "desc": "ready",
   "replicas": 1,
   "readyReplicas": 1
  },
  "urls": "http://my-nginx.192.168.7.28.xip.io",
```

```
  "created": "2019-03-14T14:08:19Z"
 }
}
```


Termination of a task:

Request: *DELETE* http://rotterdam-caas.*IP-ADDRESS*.xip.io/api/v1/docks/class/tasks/my-application

Response:


```
{
 "response": "ok",
 "message": "task my-nginx deleted",
 "content": {
  "dock": "class",
  "name": "my-nginx",
  "res": "deleted",
  "res-db": "..."
 }
}
```


Some usage examples have been included in D4.2 [2] which can be accessed by making a request to the CLASS project team.

6 Deployment and Demonstration in CLASS project

A first version of the CLASS cloud standalone environment for data analytics service management and scalability has been installed and integrated in the City of Modena Data Center.

Also, a preliminary demo was made to the project team in Haifa F2F meeting, demonstrating the CLASS integrated cloud environment and some features of the cloud service management and scalability components:

- How analytic services can be easily deployed using the rest APIs of Rotterdam
- How developers don't need to worry about clusters, VMs, instances etc.
- How services will be automatically scaled based on the monitoring of real time QoS objectives
- How platform metrics can be viewed on Grafana/Prometheus
- The integration of Rotterdam and monitoring frameworks

A video of this demonstration can be found at the following link: https://drive.google.com/file/d/1lGUZMpVQqt2fYrlS4kVE3CkCBDDhAhbY/view.

Please note that some of these features are still work in progress, i.e. to be continued in the third phase of the project, so the demo required some manual steps.

7 Conclusions

This deliverable reported on the work done in WP4 Cloud Computing from M7 to M15. It presented a publicly available summary of "D4.2: First release of the Cloud Data Analytics Service Management components", which is a confidential and more comprehensive report provided by the same work package in the project, so this report included a subset of the content from that deliverable.

The target at milestone MS2 of Task 4.2 Data Analytics Service Management and Task 4.3 Data Analytics Service Management has been successfully achieved and documented in this deliverable: A cloud standalone environment for data analytics service management and scalability.

The previous sections of this document provided a functional description of the platform and an overview of the technical details such as the architecture and interfaces provided. Installation and usage details have also been included.

An instance of the cloud standalone environment has already been installed and integrated in the city of Modena Data Center and made available to the other layers of the CLASS software architecture. Also, a demonstration has been made to the project team in Haifa F2F meeting.

The progress in this milestone will pave the way for the next phase where the goal will be providing a cloud environment coordinated with the edge layer (WP4), and an analytics scheduler (WP5).

Acronyms and Abbreviations

- D – deliverable
- EC – European Commission
- M – Month
- MS – Milestones
- WP – Work Package
- OKD – Openshift Kubernetes distribution
- SLA – Service Level Agreement

References

[1] D4.1 Cloud Requirement Specification and Definition, Deliverable D4.1 of CLASS project

[2] D4.2: First release of the Cloud Data Analytics Service Management components, Deliverable D4.2 of CLASS project

[3] Kubernetes: https://kubernetes.io/

[4] Istio: https://istio.io/

[5] Docker: https://www.docker.com/

[6] Grafana: https://grafana.com/

[7] Prometheus: https://prometheus.io/

[8] OpenShift: https://www.openshift.com/

[9] VMware: https://www.vmware.com/

[10] OKD, Openshift Kubernetes distribution: https://www.okd.io/

[11] Swagger: https://swagger.io/tools/swagger-ui/

[12] Nginx, a HTTP and reverse proxy server: https://nginx.org/en/