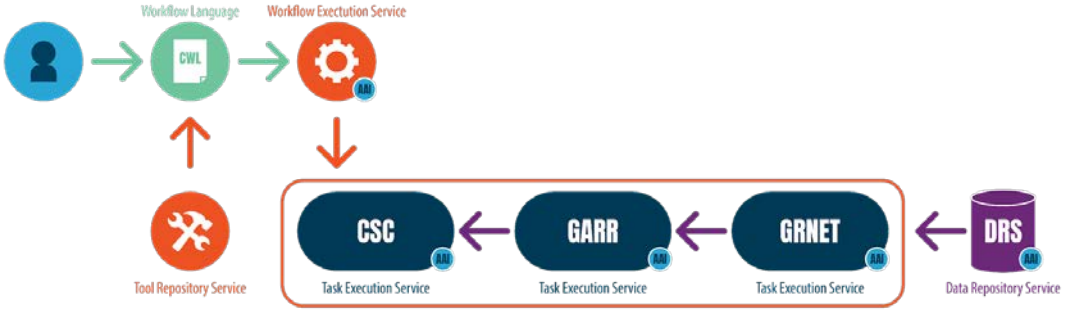
PLAS
Platformed Task

Andrea Detti: andrea.detti@uniroma2.it

Luca Petrucci: luca.petrucci@uniroma2.it

Ludovico Funari: ludovico.funari@uniroma2.it

PLAS – Platformed Tasks



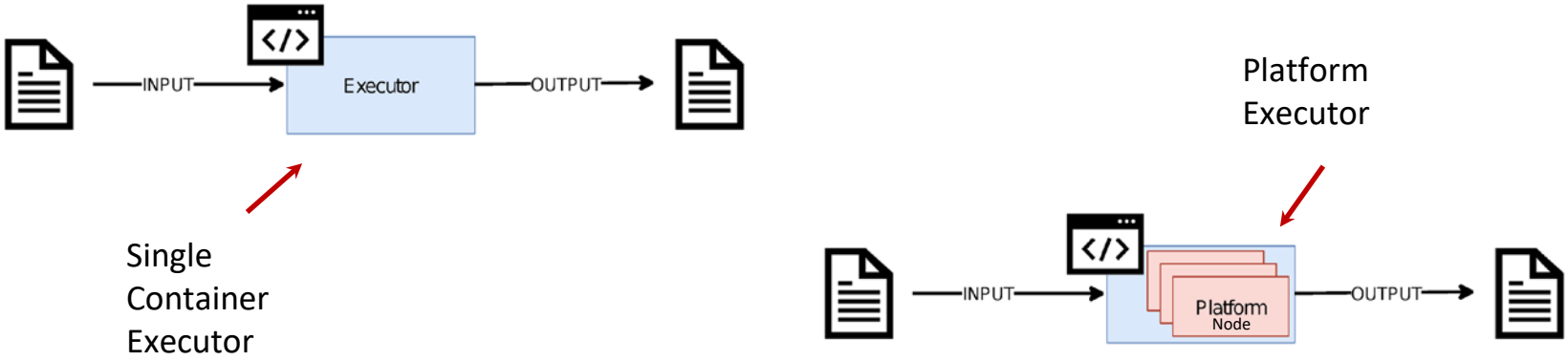
Source: https://clouds.geant.org/community-cloud

- GÉANT Cloud Flow (GCF) provides researchers with the ability to run workflows composed of analysis tasks exploiting GÉANT cloud facilities
- The PLAS project aims to extend GCF services with platformed-tasks

PLAS Goal

Enable researchers to execute single tasks not only within Containers, but also within overlay platforms installed on-demand during the task deployment phase.

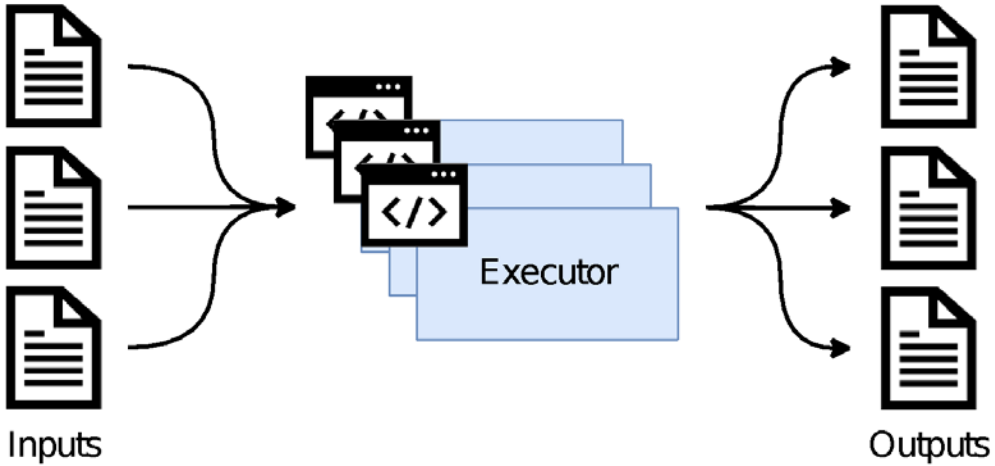
- Parallel computation



TES

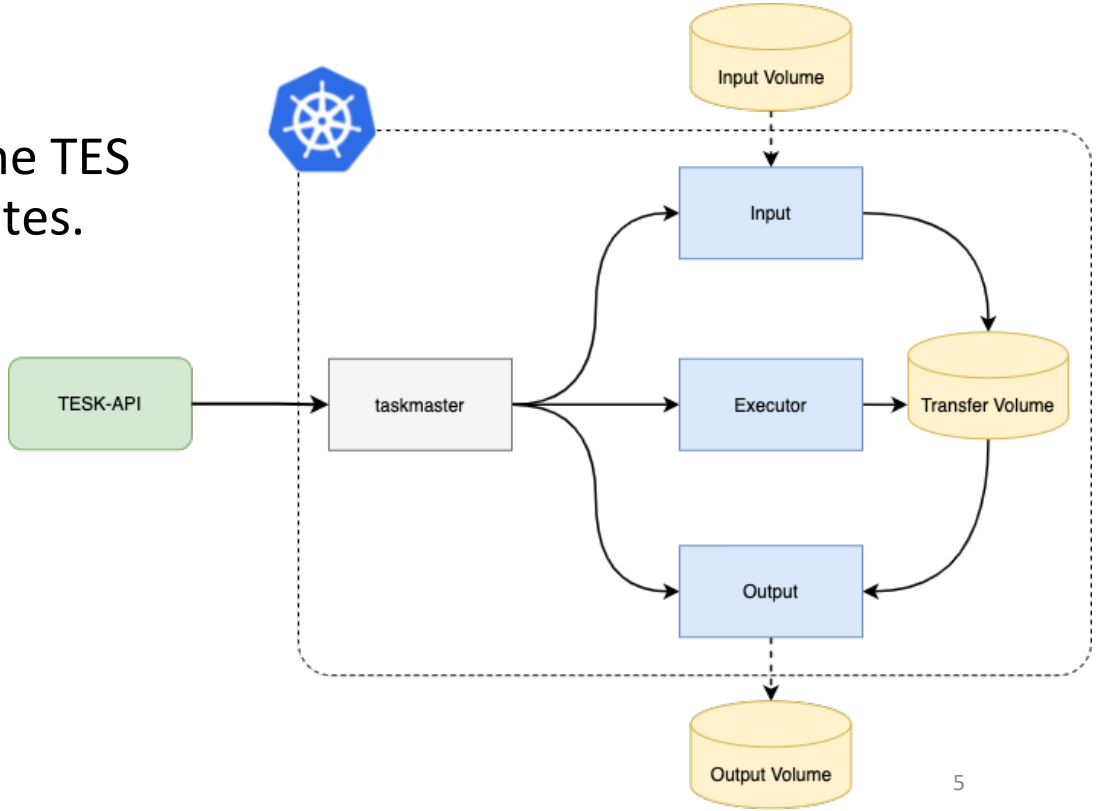
The Task Execution Service (TES) API is a standardized schema and API for describing and executing batch execution tasks.

A task defines a set of input files, a set of containers and commands to run, a set of output files.



TESK - TES on Kubernetes

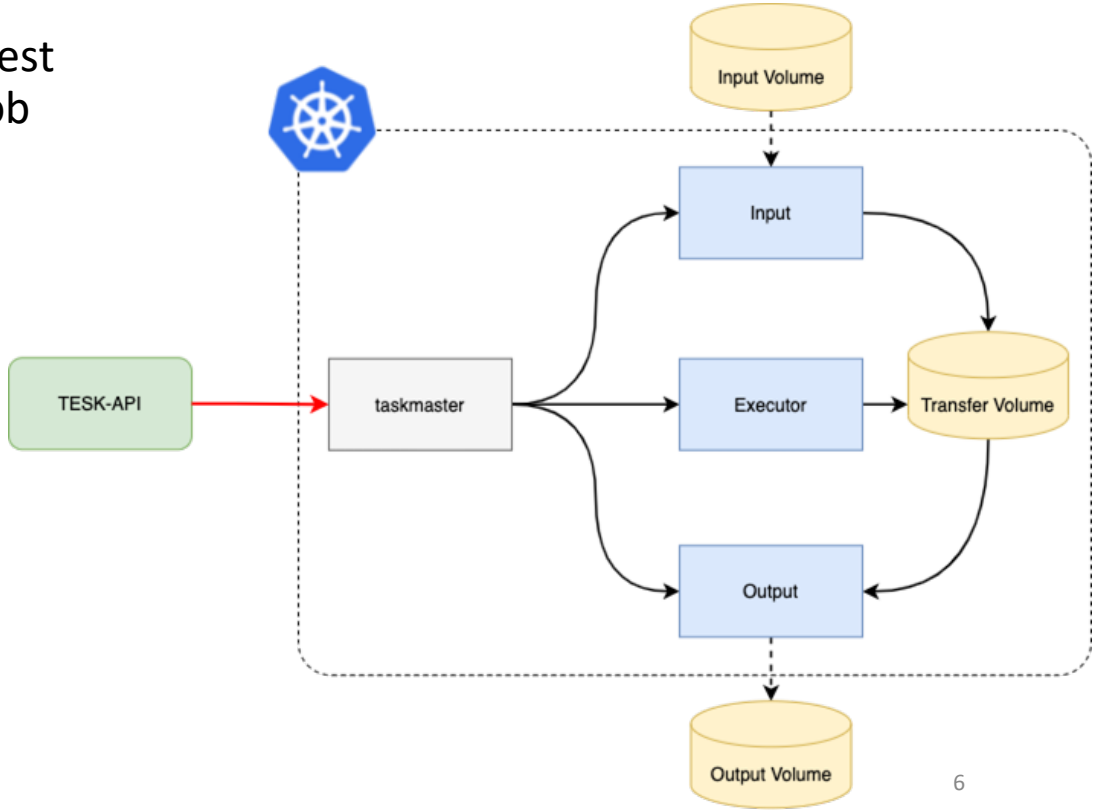
An implementation of a task execution engine based on the TES standard running on kubernetes.



https://github.com/elixir-cloud-aai/TESK

TESK Lifecycle

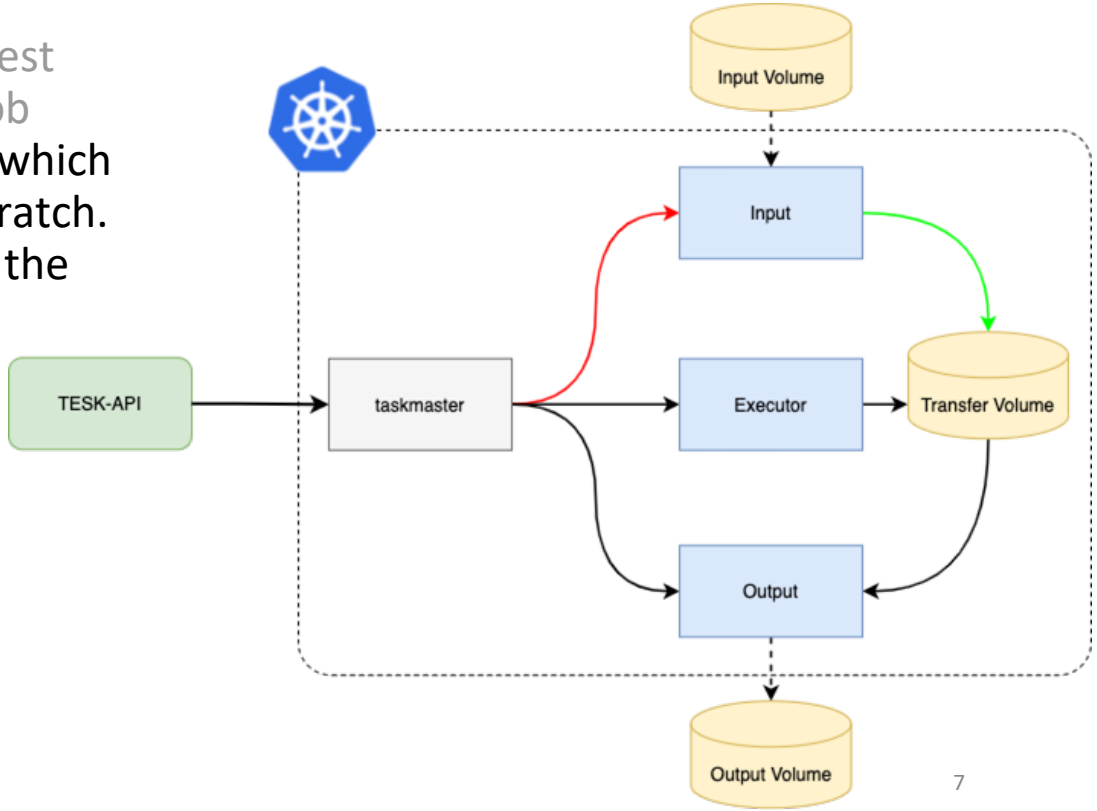
1. TESK-API consumes TES request and translates them to K8s job



https://github.com/elixir-cloud-aai/TESK

TESK Lifecycle

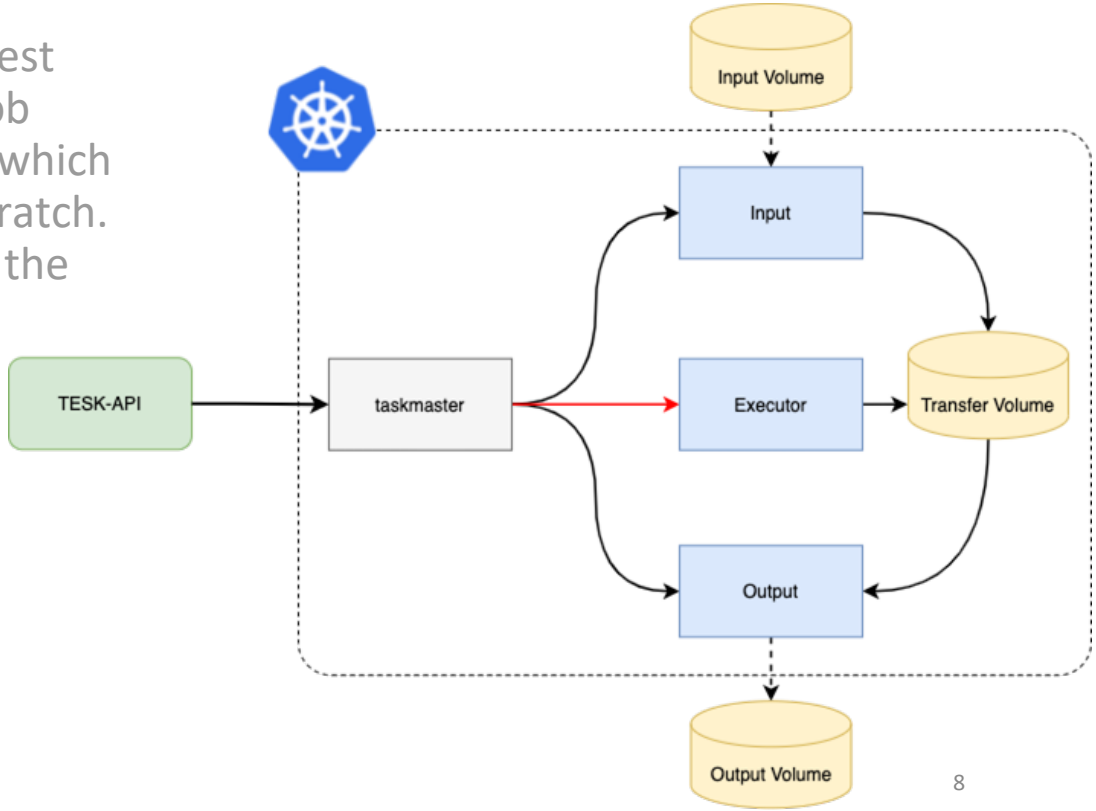
1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request



https://github.com/elixir-cloud-aai/TESK

TESK Lifecycle

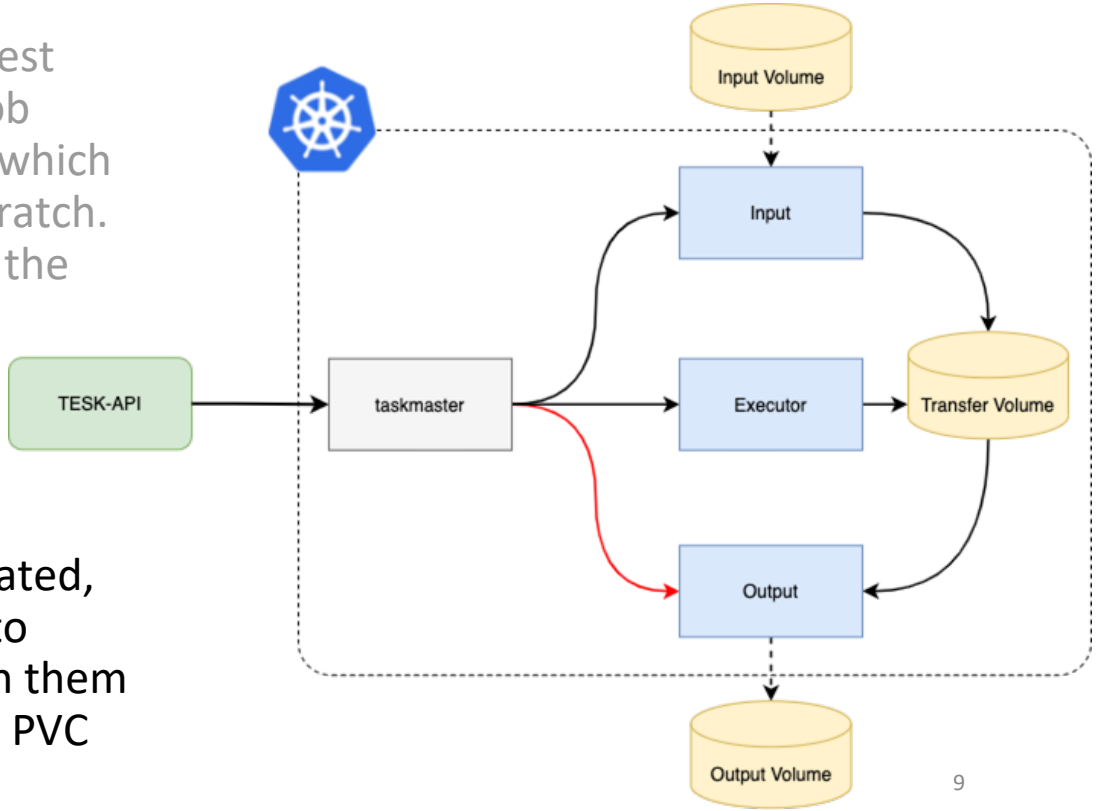
1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request
3. After the *filer* has finished, the taskmaster execute the *executor* as pod



https://github.com/elixir-cloud-aai/TESK

TESK Lifecycle

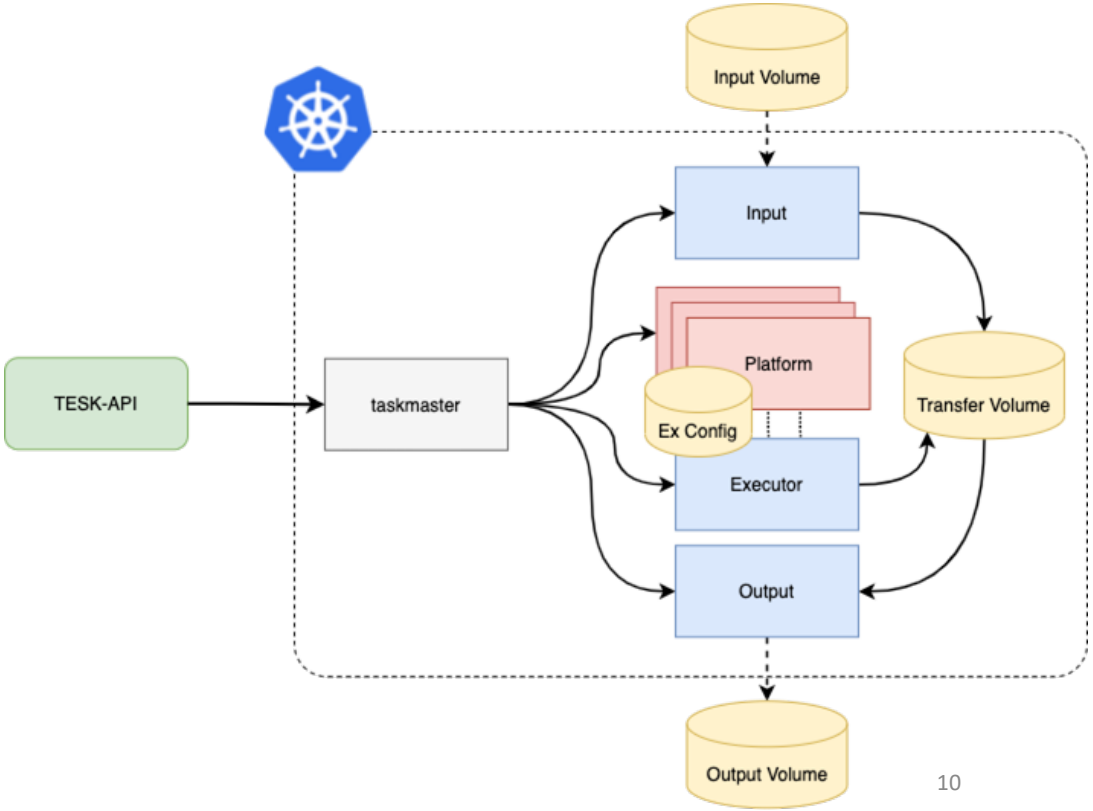
1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request
3. After the *filer* has finished, the taskmaster execute the *executor* as pod
4. When the executor is terminated, the filer is called once more to process the outputs and push them to remote locations from the PVC



https://github.com/elixir-cloud-aai/TESK

TESK - Extension

The figure show our extended architecture of TESK to support Platformed Task

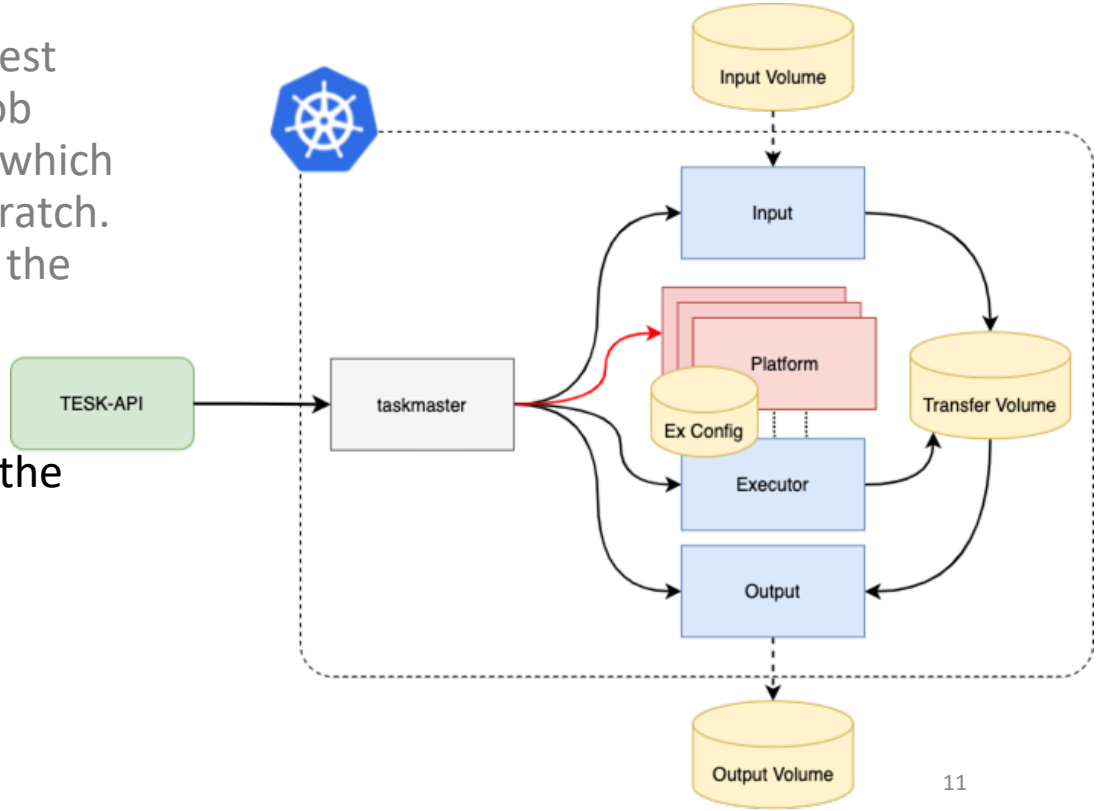


https://github.com/elixir-cloud-aai/TESK

TESK Extended Lifecycle

1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request

3a. After the *filer* has finished, the taskmaster deploys the *platform* with **Helm**



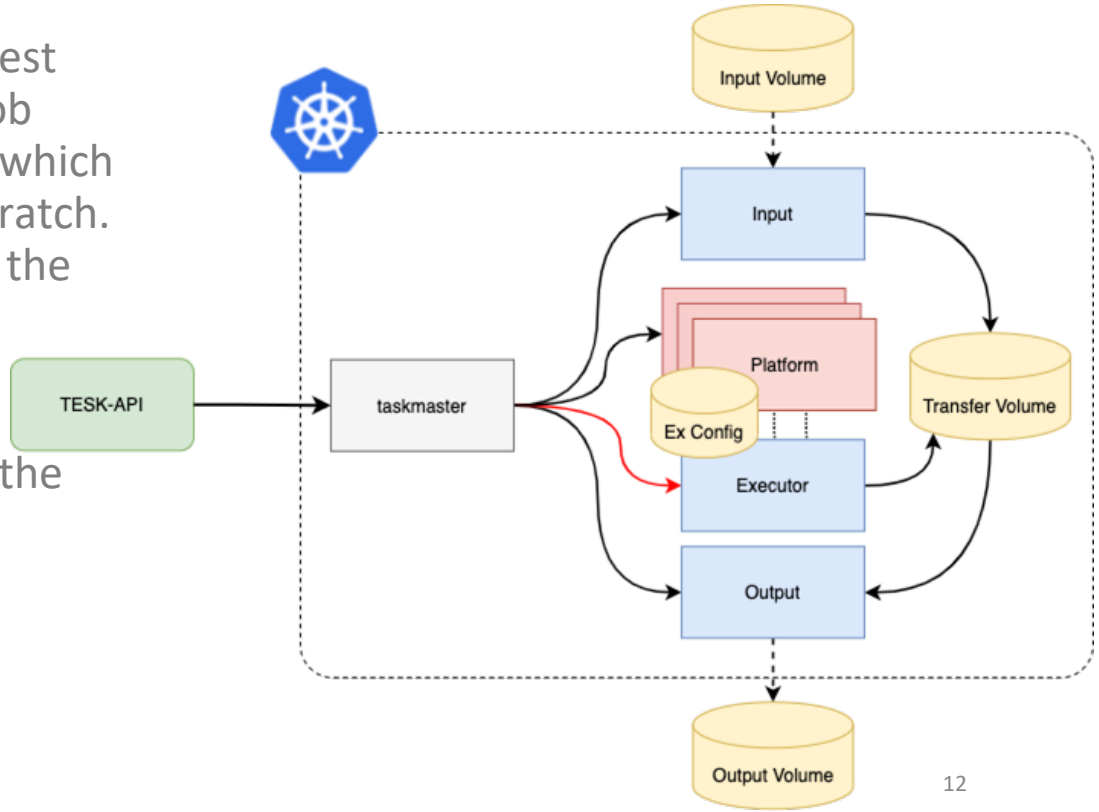
https://github.com/elixir-cloud-aai/TESK

TESK Extended Lifecycle

1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request

3a. After the *filer* has finished, the taskmaster deploys the platform with helm

3b. After the *platform* is ready, the taskmaster runs the *executor* as pod



https://github.com/elixir-cloud-aai/TESK

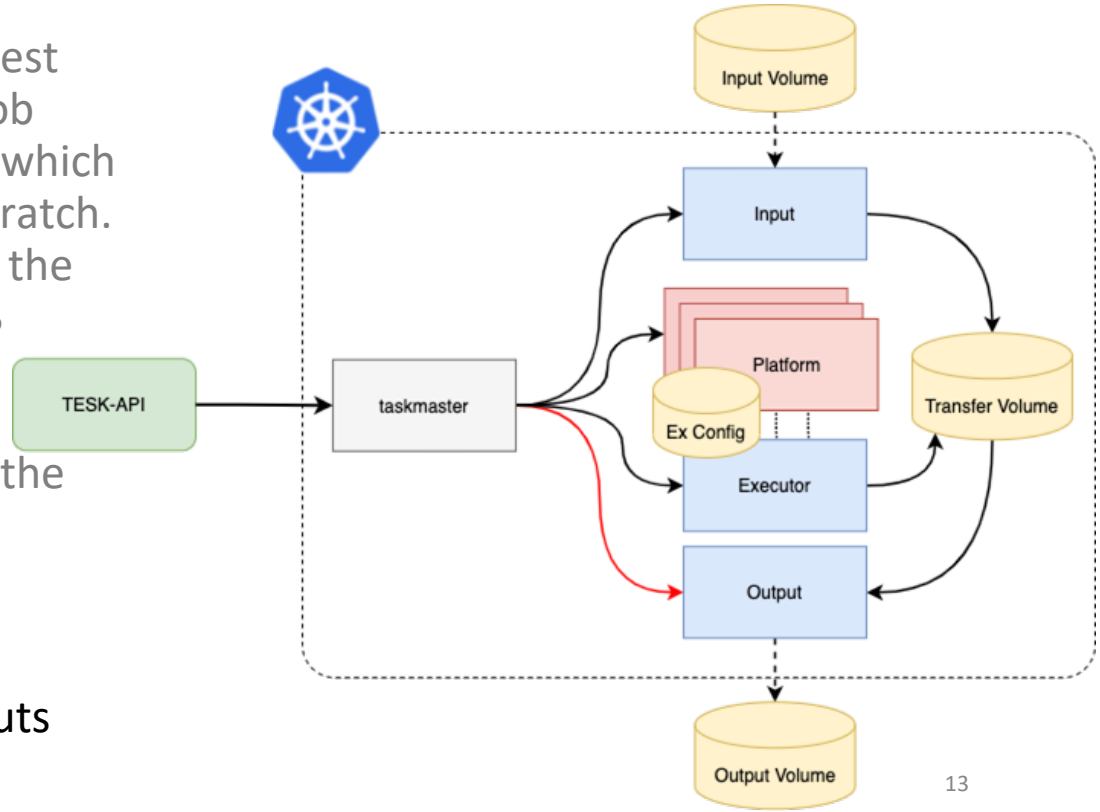
TESK Extended Lifecycle

1. TESK-API consumes TES request and translates them to K8s job
2. Taskmaster creates *filer* pod which creates a *PVC* to mount as scratch. All files are downloaded into the locations specified in the TES request

3a. After the *filer* has finished, the taskmaster deploys the platform with helm

3b. After the *platform* is ready, the taskmaster runs the *executor* as pod

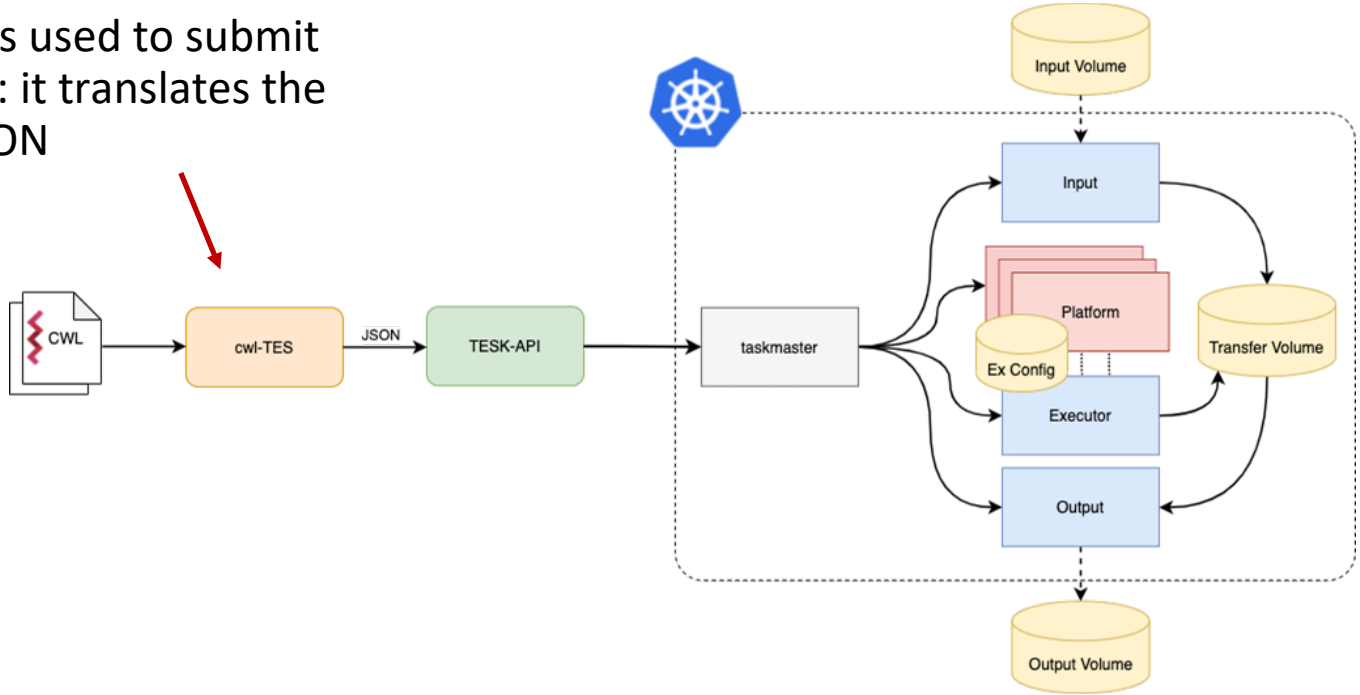
4. The *filer* saves the outputs



https://github.com/elixir-cloud-aai/TESK

TESK - cwl-TES

cwl-tes component is used to submit tasks to a TES server: it translates the *task* from CWL to JSON



https://github.com/elixir-cloud-aai/TESK

CWL – Task

COMMON WORKFLOW LANGUAGE

hashsplitter-sha.cwl.yml

```yaml
cwlVersion: v1.0
class: CommandLineTool
hints:
  - class: DockerRequirement
    dockerPull: kubler/openssl
inputs:
  - id: input
    type: File
    doc: "original content"
    inputBinding:
      position: 1
outputs:
  - id: output
    type: stdout
stdout: sha
baseCommand: ["openssl", "dgst"]
arguments: ["-sha1"]
```

A minimal TES task representation

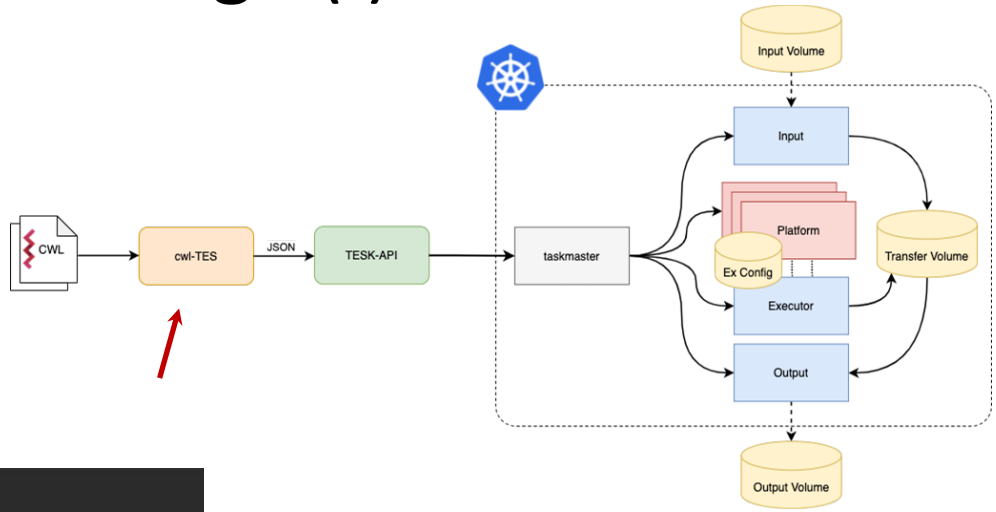
input-file.json

```json
{
    "input": {
        "class": "File",
        "location": "ftp://10.0.0.10/files/input.txt"
    }
}
```

Extension Challenges

cwl-TES - Extension Challenge (I)

Extend the cwl-tes component to add the support to new Common Workflow Language (CWL) `HelmRequirement` class.

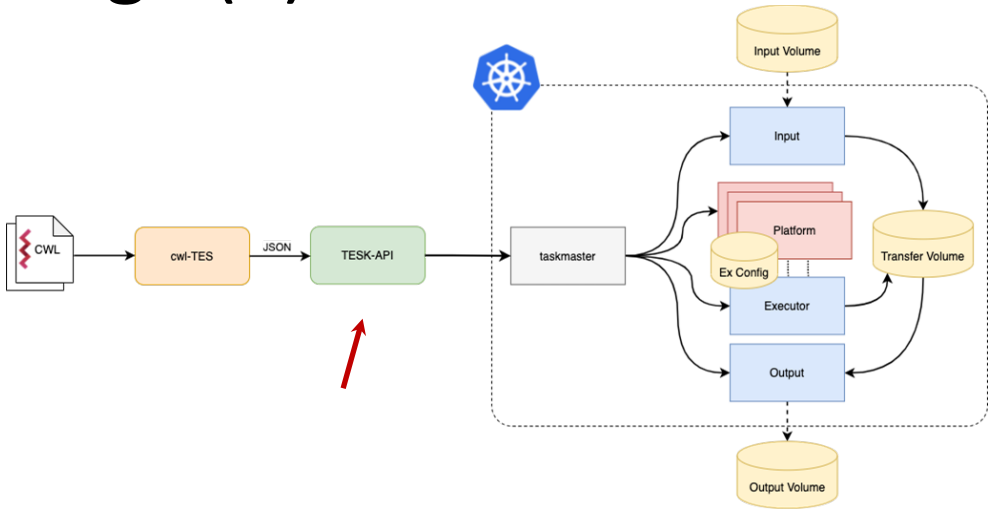



```
cwlVersion: v1.0
class: CommandLineTool
requirements:
  - class: HelmRequirement
    chartRepo: "https://link_to_horovod_chart_repo/charts"
    chartVersion: "3.0.0"
    chartName: "horovod"
    executorImage: "horovod:latest"
```

Note: Supported chart repository link and his specific executor image is needed

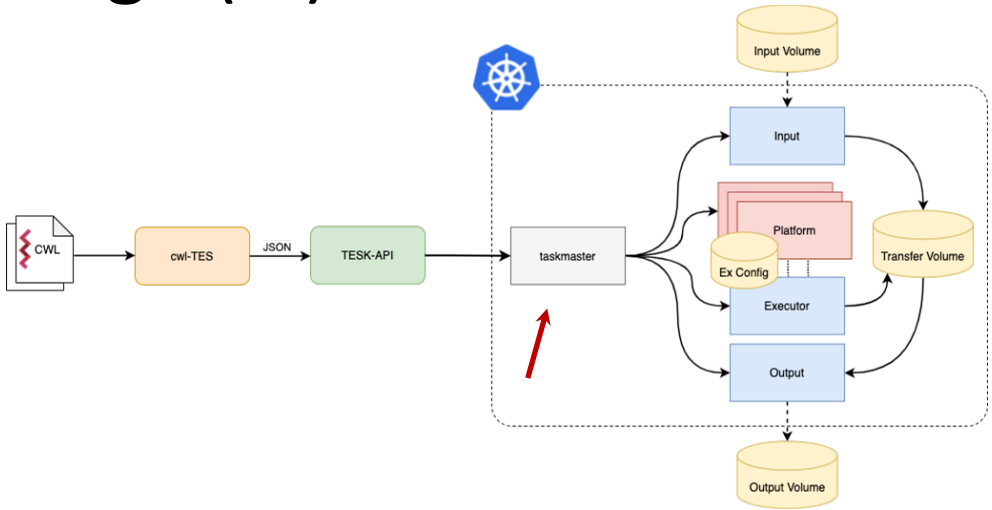
TESK - Extension Challenge (II)

Extend the TESK-API component to forward changes and generate the taskmaster pod with additional volume and extra necessary details.



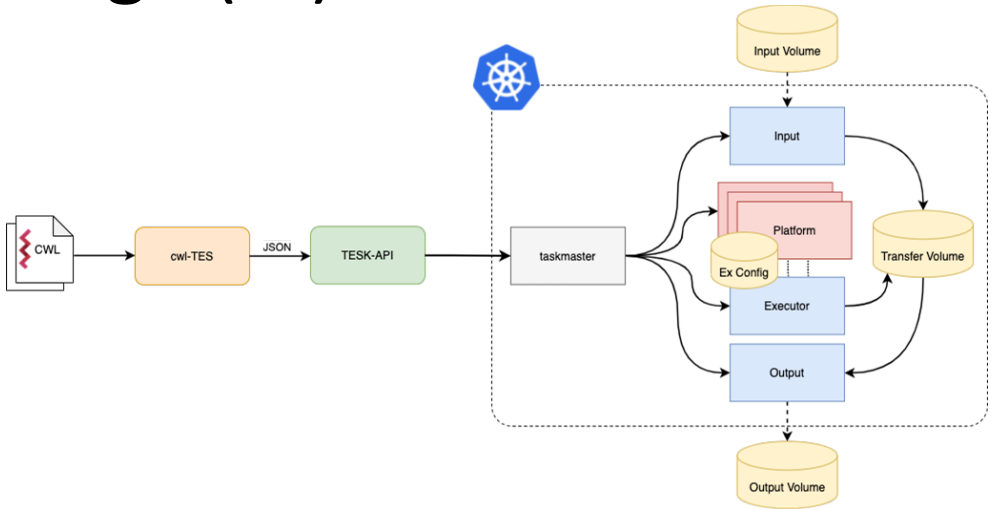
TESK - Extension Challenge (III)

Improve the capability of the *taskmaster* to deploy the platform using **Helm**, in addition to the creation of input output and executor.



TESK - Extension Challenge (IV)

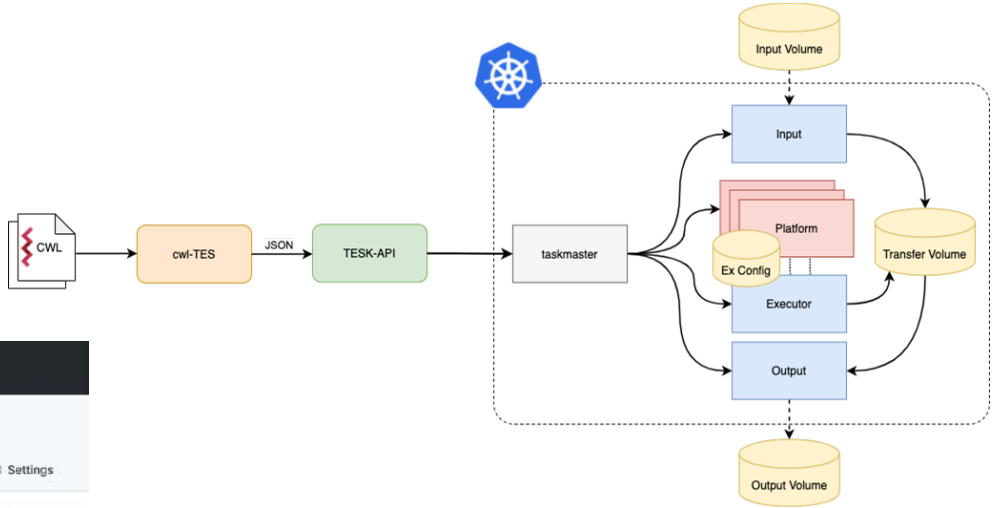
- Knowledge of the *horovod* platform itself before the implementation with CWL workflow
- Developing the specific *horovod* executor image
- Integrate all with the extended components

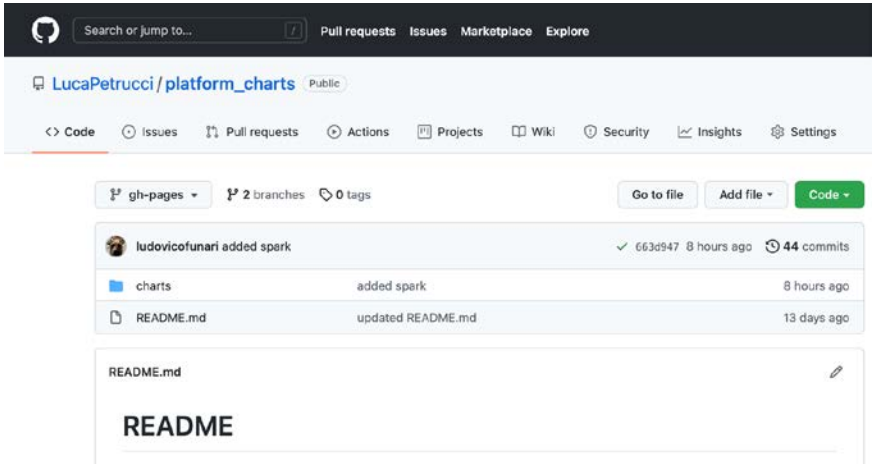



```
NAME                                    READY   STATUS
task-a963dd6a--1-8zhrq                  1/1     Running
task-a963dd6a-ex-00--1-57c9w            0/1     Completed
task-a963dd6a-inputs-filer--1-hmmqd     0/1     Completed
task-a963dd6a-outputs-filer--1-phhc6    0/1     Completed
task-a963dd6a-platform-horovod-0        1/1     Terminating
task-a963dd6a-platform-horovod-1        1/1     Terminating
tesk-api-d4f8579c8-jjkv5                1/1     Running
```

PLAS - Repositories

Make a *github* chart repository with the compatible charts (at the moment *horovod* and *spark*) and the corresponding executor.





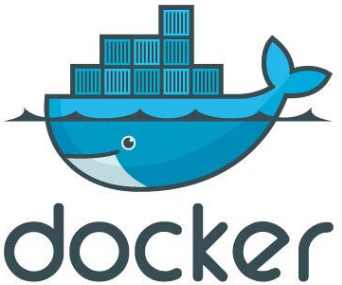
Repositories

We have a GitHub repository for each module:

- cwl-TES
- TESK-API
- Taskmaster
- Platform helm charts:
  - horovod
  - spark

We also have developed the Docker Images of:

- TESK-API
- Taskmaster
- horovod
- spark



Conclusion and Future works

- We are able to perform a platformed task
- We have implement the horovod and spark use cases
- Platform security issues need to be evaluated

DEMO