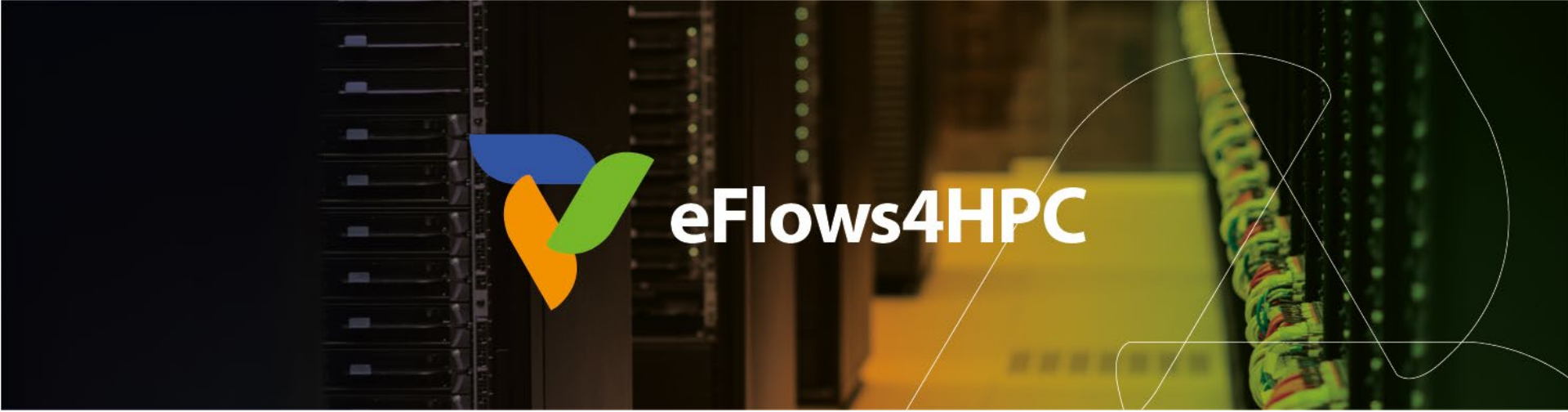


Demo Session: Deployment and Execution the PTF Workflow with HPCWaaS

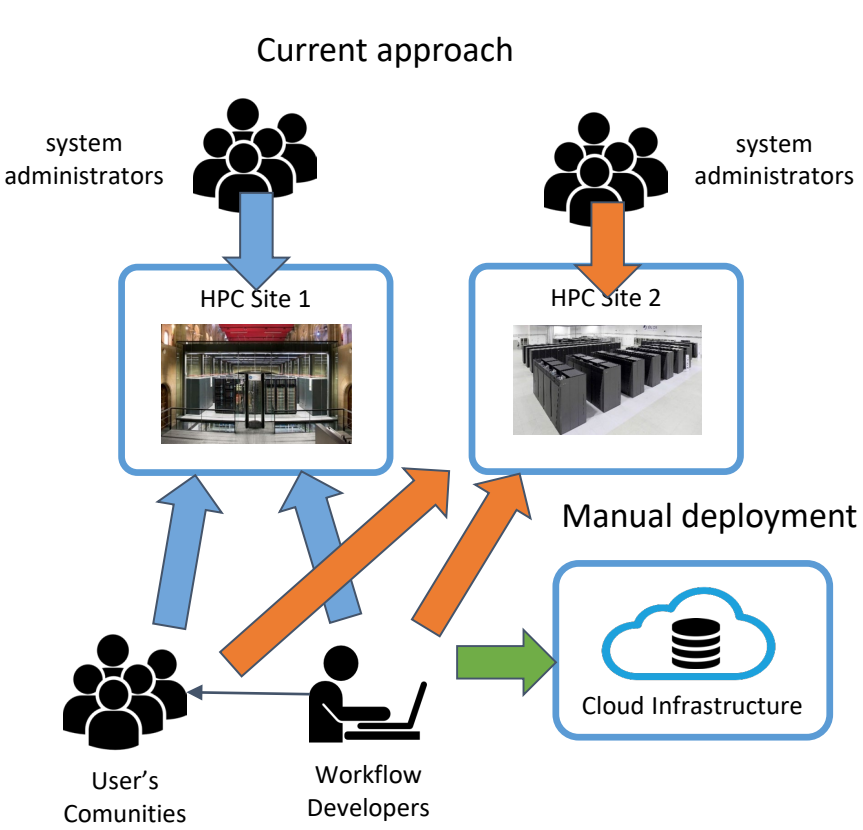
Jorge Ejarque (BSC)

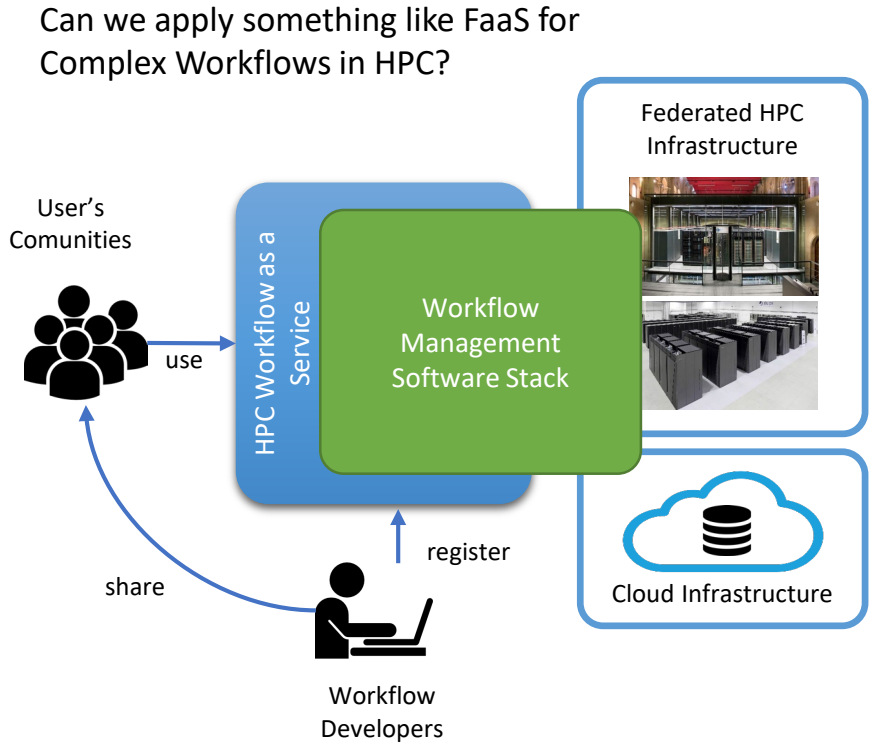
Next-generation HPC workflows for natural hazards

Barcelona, September 13, 2023

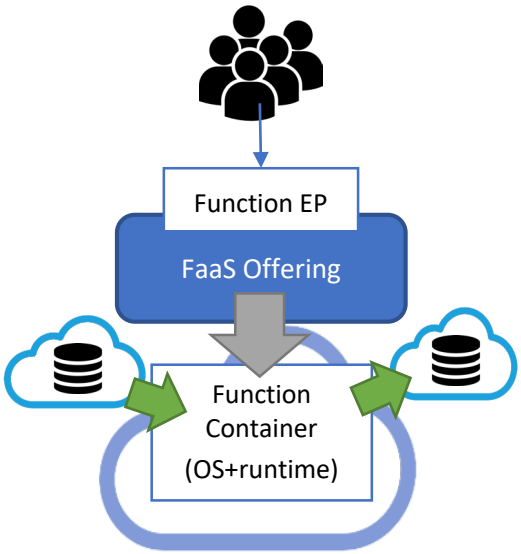
This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Germany, France, Italy, Poland, Switzerland, Norway. MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR (PCI2021-121957)

Deployment in HPC Environments





FaaS vs. HPCWaaS

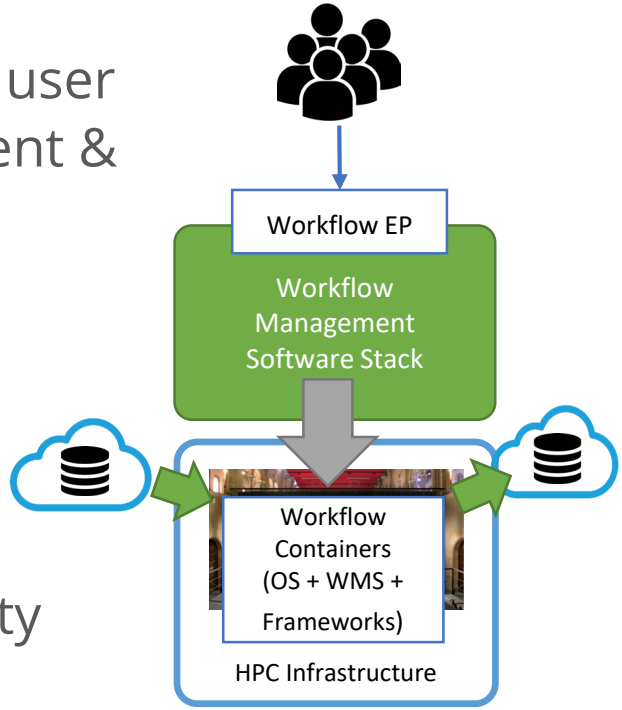


Similarities

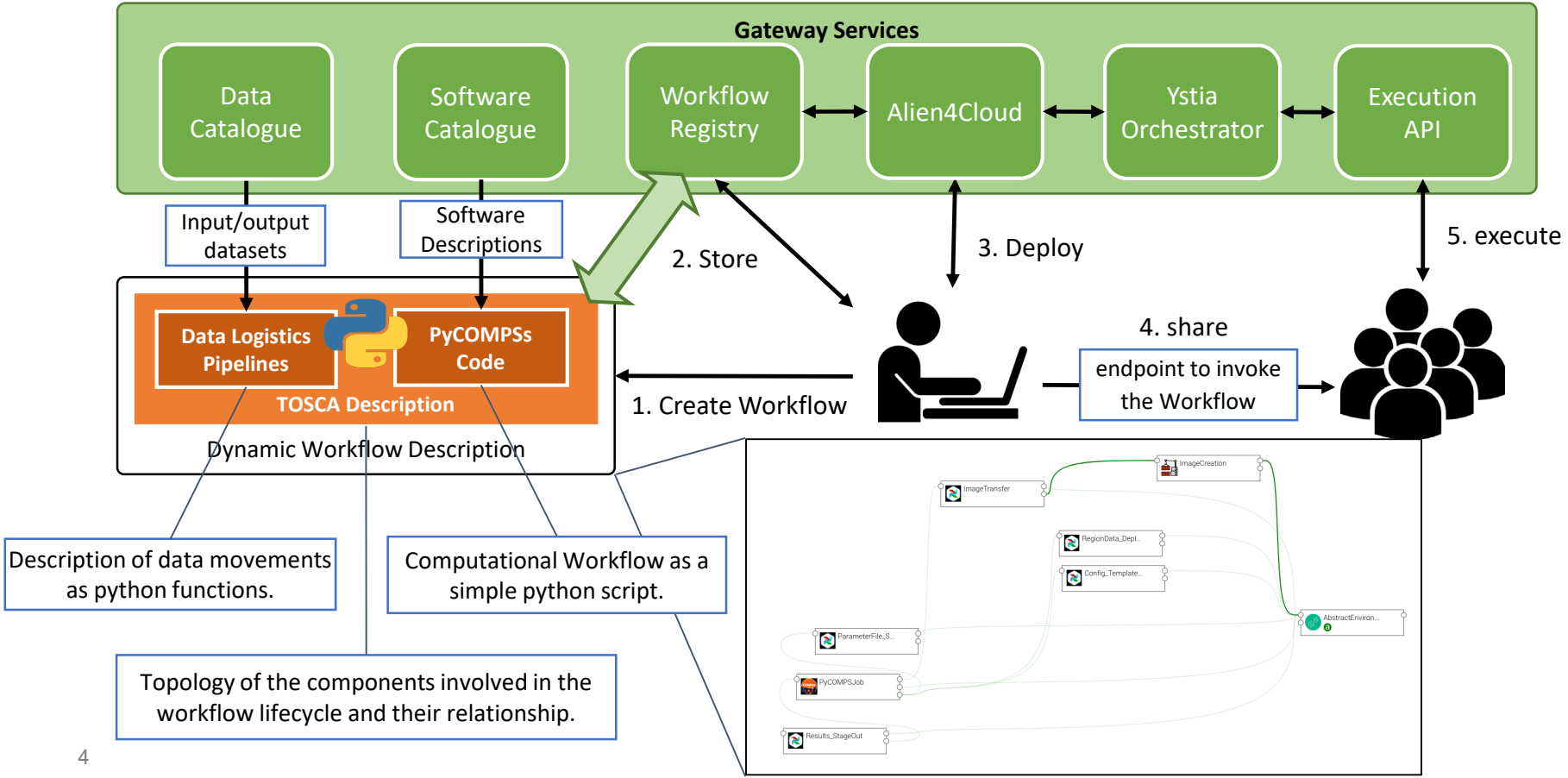
- Easy to use for final user
- Automate deployment & execution
- Data integration
- Containers

Differences

- Restrictions
- Deployment and Execution Complexity
- Performance



Development Overview



PTF Workflow

- Data Management
  - Required data and Results stored in the B2DROP and must be moved from/to HPC
  - Data Logistics Service and Data Catalogue
- Software Deployment
  - Workflows Code and required software in the HPC with Containers
  - Container Image Creation:
    - ✓ Build a container tailored for the target HPC machine
- Deployment and Execution Automation
  - TOSCA topology in the workflow registry
  - HPCWaaS:
    - ✓ Key management
    - ✓ Orchestration the Image creation, Data pipeline and PyCOMPS executions

PyCOMPSs

```
@binary(binary=config_bin)
@task(config_file=FILE_OUT)
def build_config(config_template, config_file, data_dir, files_step2, par_file, kag, tsu, event_id):
    pass
```

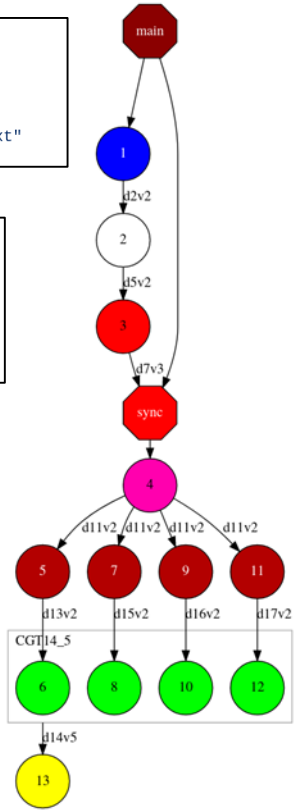
```
@task(config_file=FILE_IN, returns=1)
def step1_func(args, config_file, seistype, sim_files_step1):
    args.cfg=config_file
    run_step1_init(args,sim_files_step1)
    return sim_files_step1 + "/Step1_scenario_list_"+seistype+".txt"
```

```
@binary(binary=simulBS_bin, working_dir="{{wdir}}")
@task(sim_files_step2=FILE_OUT)
def build_structure(seistype, grid, hours, group, sim_files_step2, load_balancing, pois_ts_file,
    pass
```

```
@constraint(processors=[{'processorType':'CPU', 'computingUnits':'1'},
                        {'processorType':'GPU', 'computingUnits':'1'}])
@mpi(binary=tsunamiHySEA_bin, args="{{file_in}}", runner="mpirun", processes=gpus_per_exec, processes_per_node=gpus_per_node,working_dir="{{wdir}}")
@task(file_in=FILE_IN,returns=1)
def mpi_func(file_in, wdir):
     pass
```

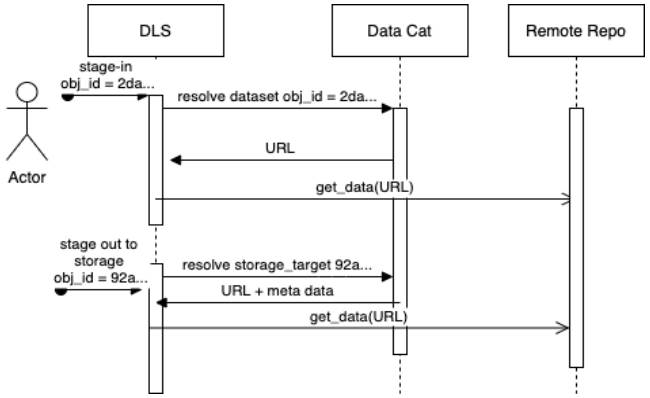
```
@task(ptf_files=COMMUTATIVE, config_file=FILE_IN)
def append_and_evaluate(ptf_files, ptf_file, args, config_file, sim_files_step1, out_step2_path, out_update_path, out_final, depth_file, log_file, sim_pois_ts, num_sims, kag, tsu,
    args.cfg = config_file
    ptf_files.append(ptf_file)
    if (num_sims != 0) and (len(ptf_files) % num_sims == 0):
        step2_create_ptf_input(ptf_files, out_step2_path, depth_file, log_file)
        if kag>0:
            run_step_kagan(args,sim_files_step1,out_update_path)
            sim_files_input=out_update_path
        elif tsu>0:
            run_step_mare(args, sim_files_step1, out_update_path, sim_pois_ts, ptf_files)
            sim_files_input=out_update_path
        else:
            sim_files_input=sim_files_step1
        run_step3_init(args, sim_files_input, out_final, sim_pois_ts, ptf_files)
```

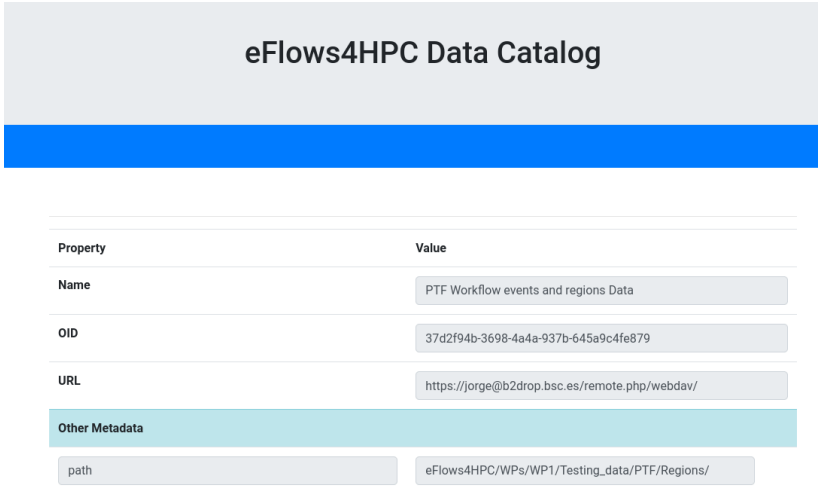
```
@binary(binary="tar", args="zcvf {{outfile}} {{folder}}")
@task(outfile=FILE_OUT)
def compress(folder, outfile, ptf_files):
    pass
```

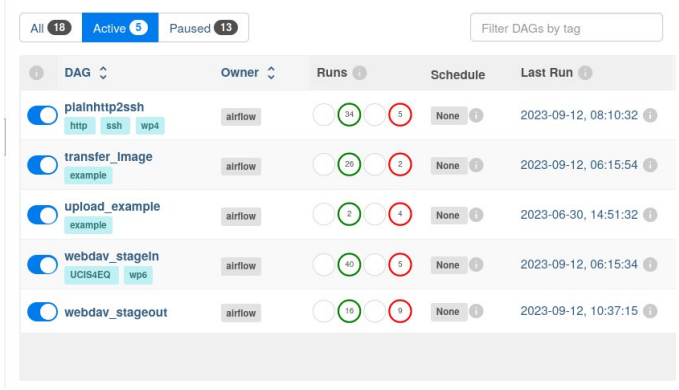


Data pipelines

- Implemented in Data Logistics Service
- Reusable for multiple data/workflows
- Configured from Data Catalogue

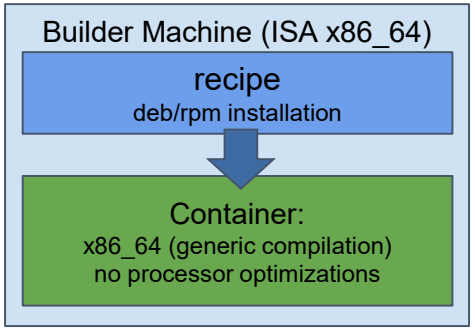






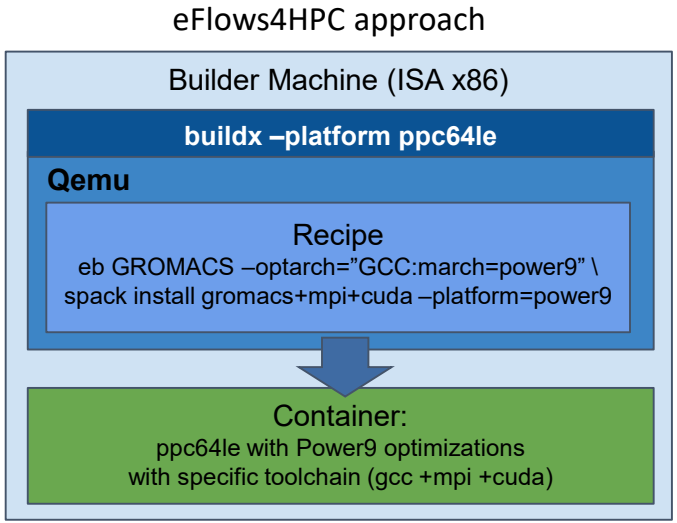
Containers and HPC

Standard container image creation



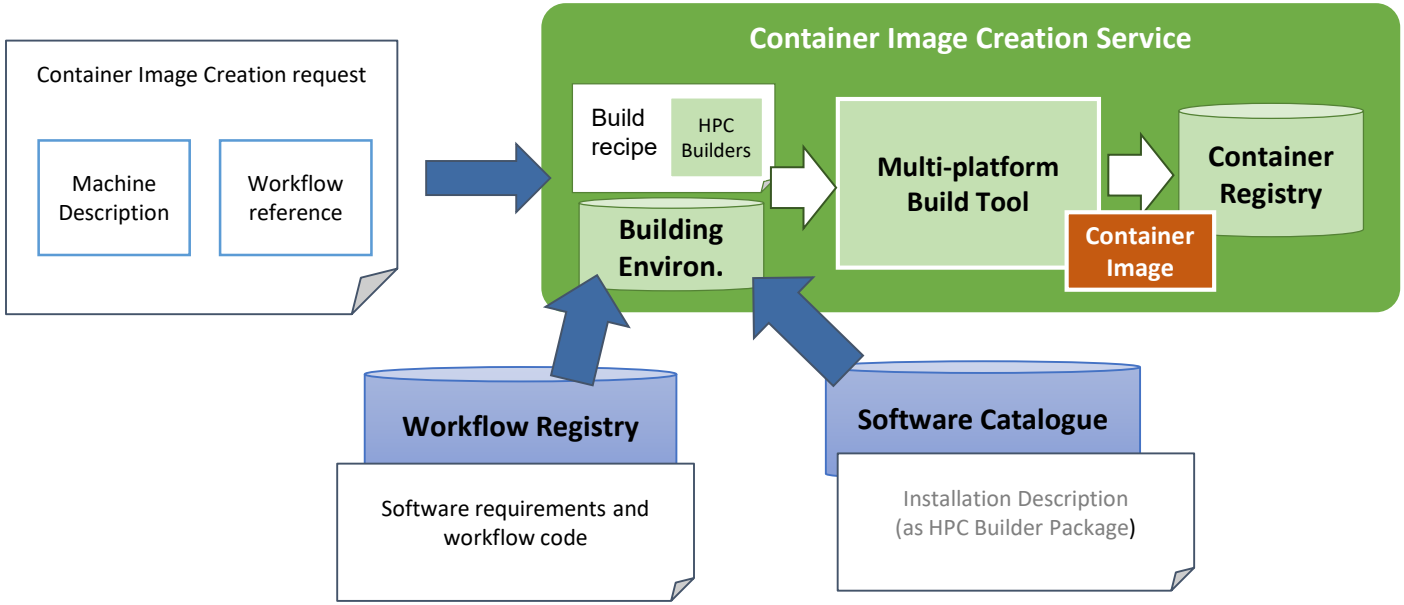
- **Simplicity for deployment**
  - Just pull or download the image
- **Trade-Off performance/portability**
  - Architecture Optimizations
- **Accessing Hardware from Containers**
  - MPI Fabric /GPUs
- **Host-Container Version Compatibility**

HPC Ready Containers



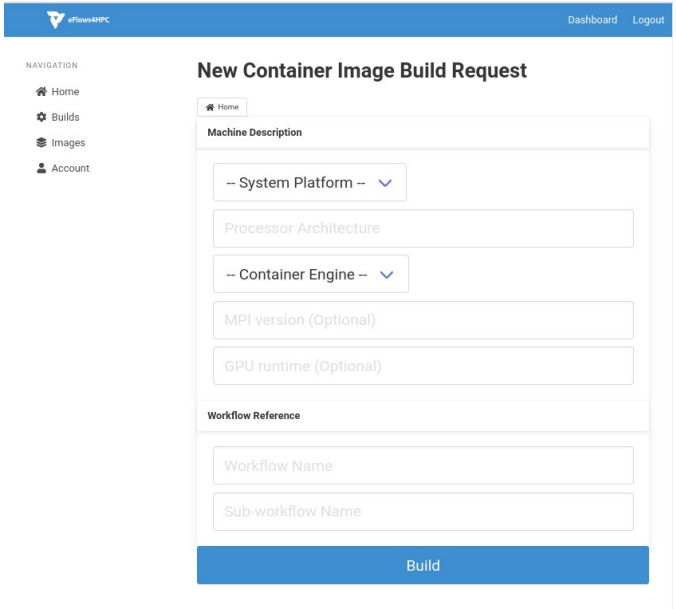
- **Methodology to allow the creation containers for specific HPC system**
  - Leverage HPC and Multi-platform container builders
- **It is tight to do by hand but let’s automate!**

Container Image Creation Service



Container Image Creation Service

- Web Interface



- REST Interface and CLI

`POST /build/`

```
{
  "machine": {
    "platform": "linux/amd64",
    "architecture": "rome",
    "container_engine": "singularity"},
  "workflow":"minimal_workflow",
  "step_id" :"wordcount",
  "force": False
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json
```

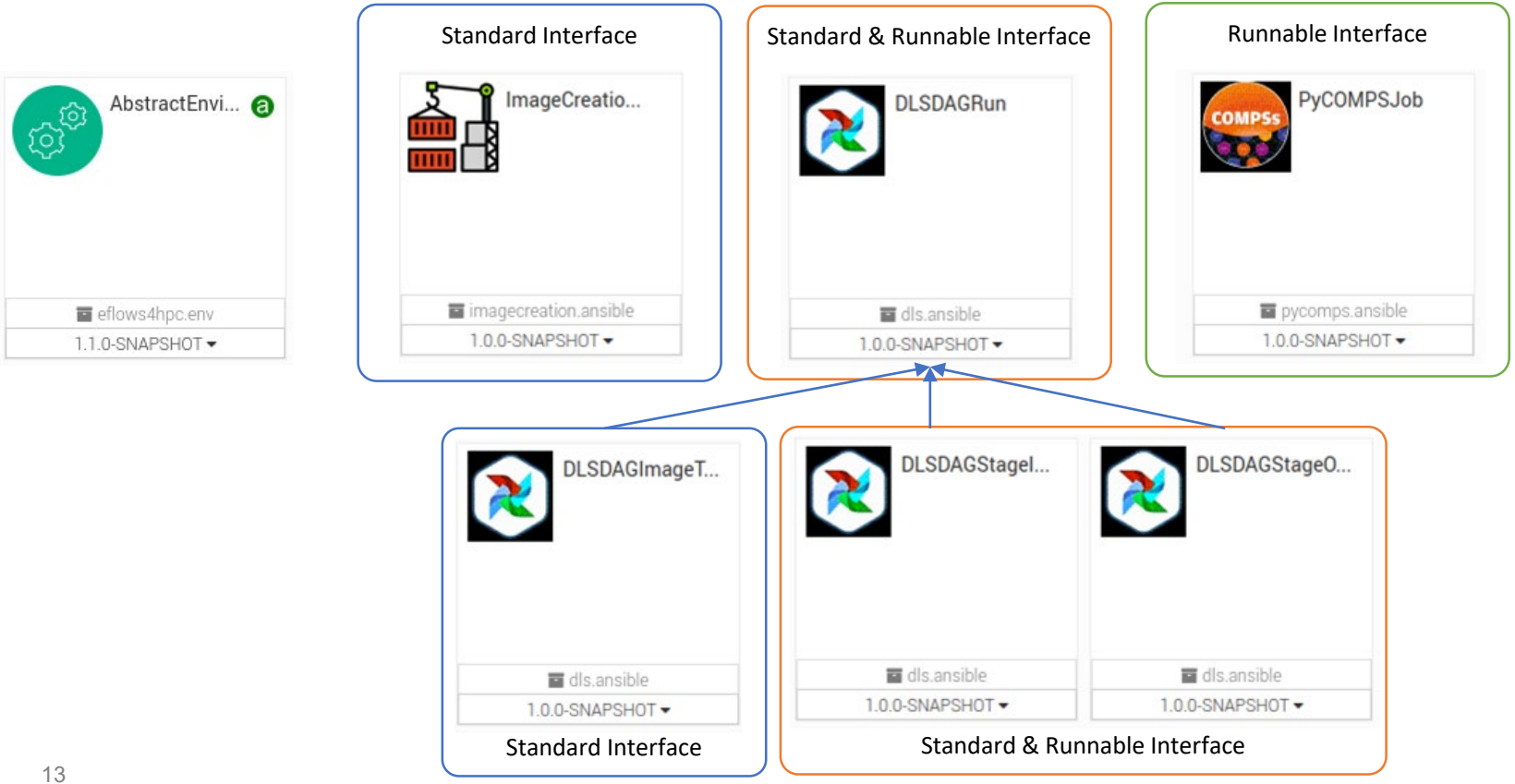
```
{
  "id": "<creation_id>"
}
```

```
localhost:~/image_creation> ./cic_cli <user> <token> https://<image_creation_url> build <request.json>
Response:
{"id":"f1f4699b-9048-4ecc-aff3-1c689b855adc"}
```

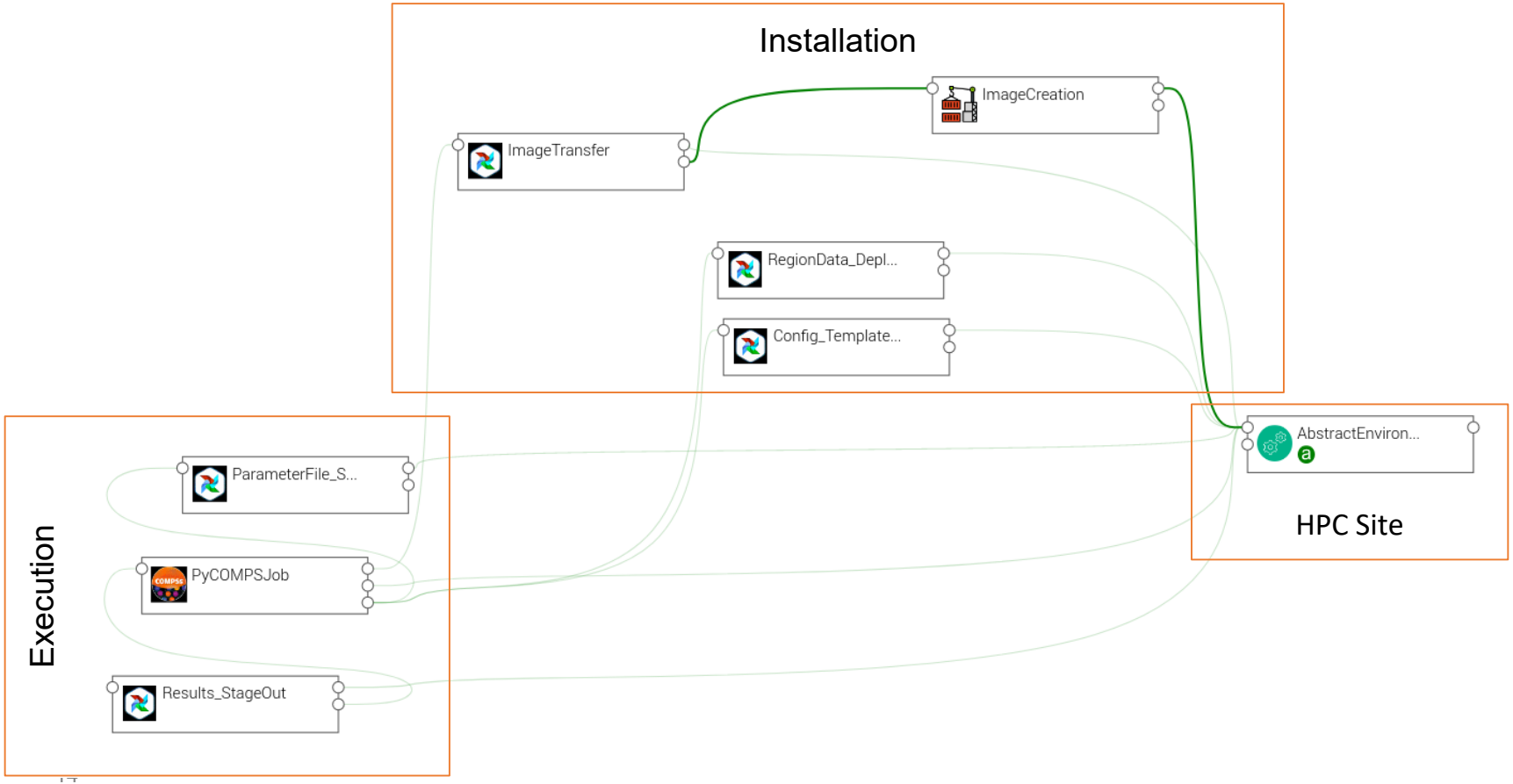
TOSCA Model

- **Describe the orchestration of the application lifecycle management**
- **Topology of components with dependencies**
  - Application Component:
    - Describe what to do in every lifecycle step
      - ✓ Standard tosca steps (start, stop, delete,…)
      - ✓ Extended runnable (submit, run, cancel,…) Integrate jobs in Tosca.
    - The required input data and properties
  - Dependencies:
    - Describe the data exchanged between components.
- **Workflows**
  - Topology generate the standard TOSCA workflows to deploy/undeploy the application
  - Custom workflows

eFlows4HPC TOSCA Components

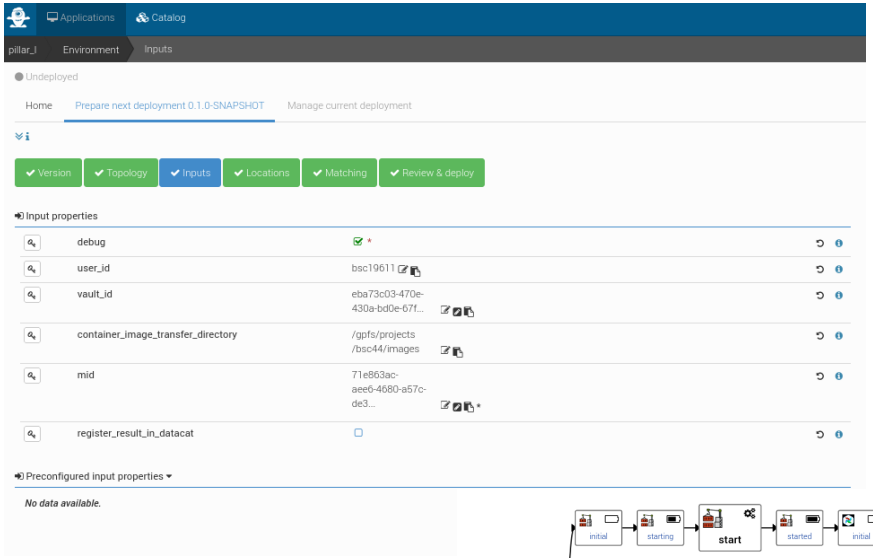


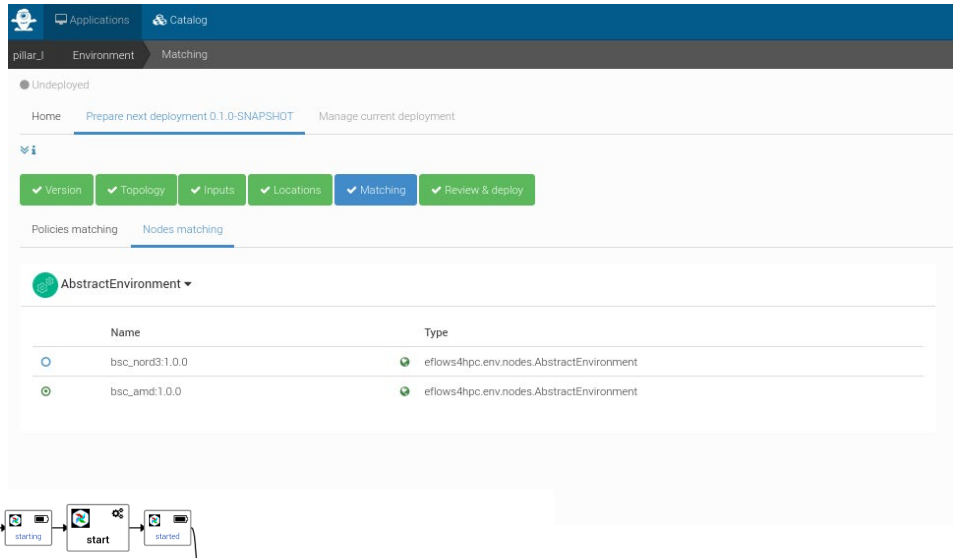
TOSCA Modelization

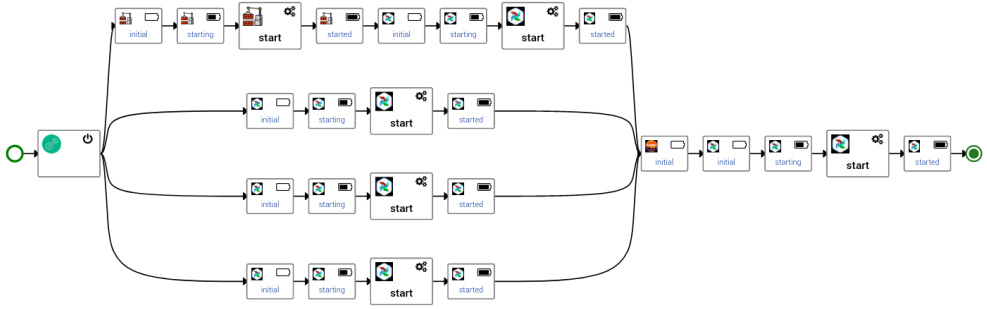


Workflow Deployment (done once per HPC site)

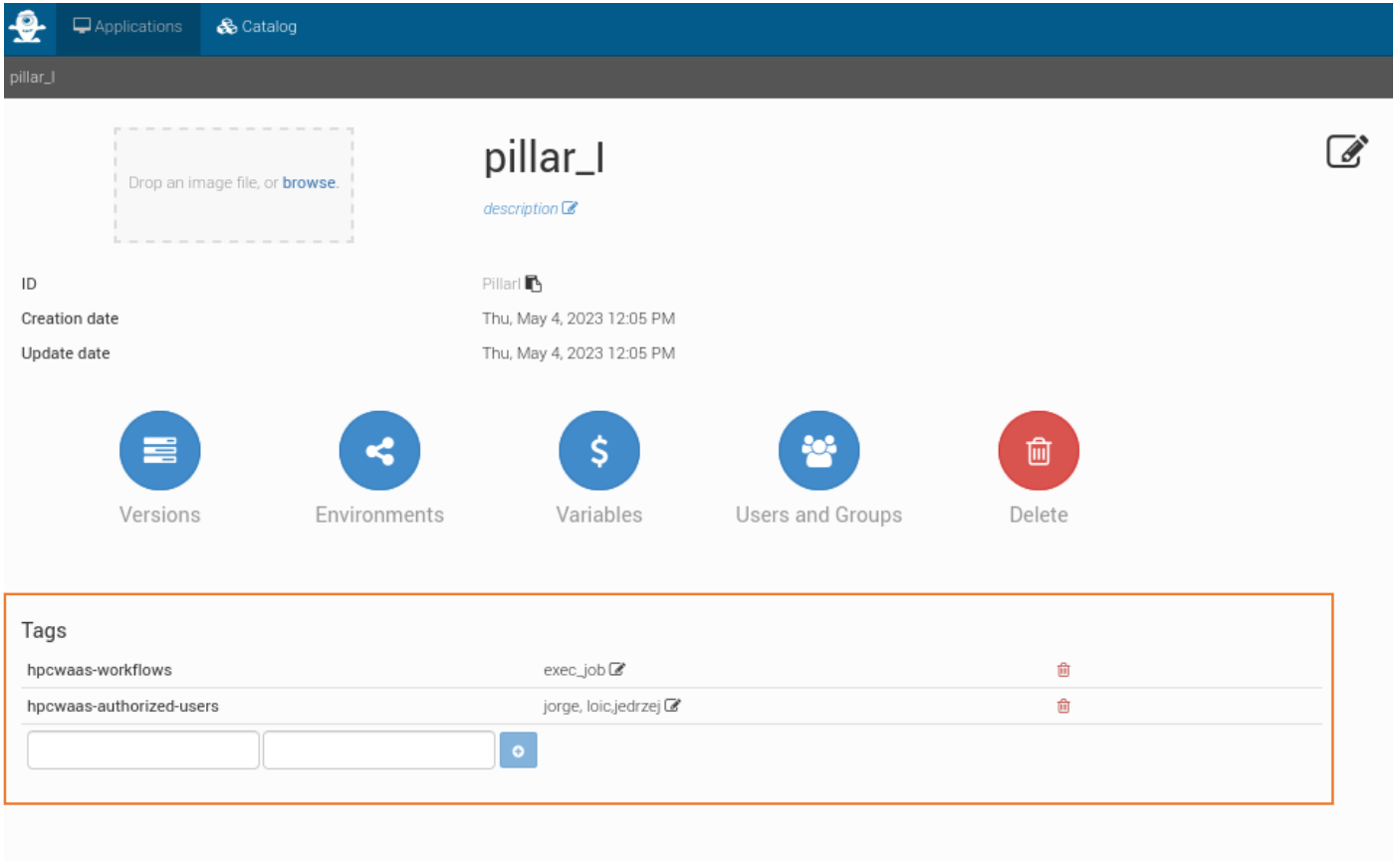
- Set deployment input parameters (user, credential, select HPC location)



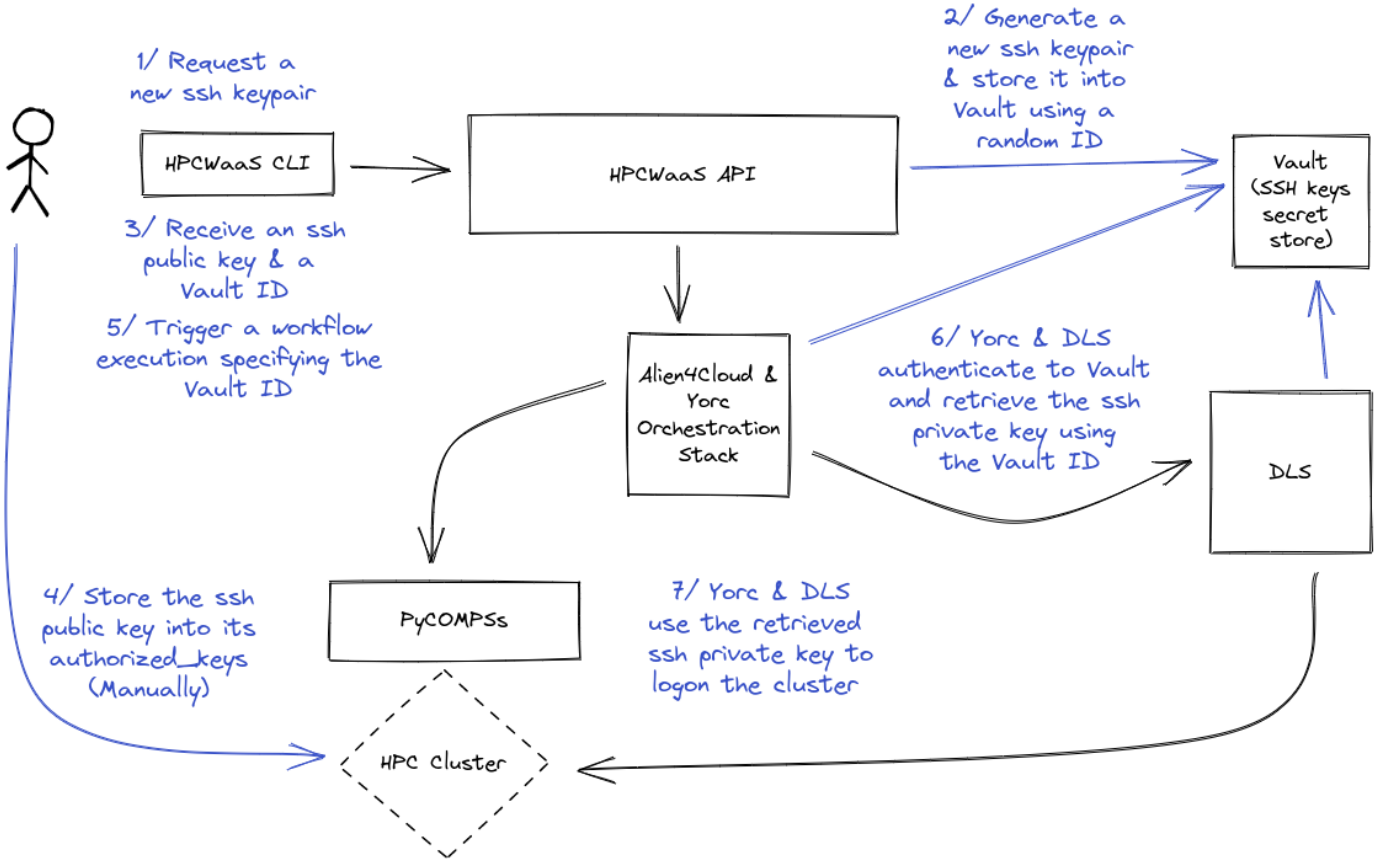




Publish workflow and authorize users



Workflow Execution End user



Thank you

eFlows4HPC

Enabling dynamic and Intelligent workflows
in the future EuroHPC ecosystem

www.eFlows4HPC.eu

@eFlows4HPC eFlows4HPC Project

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Germany, France, Italy, Poland, Switzerland, Norway.