Demo Session: Deployment and Execution of a Workflow with HPCWaaS

Jorge Ejarque (BSC)

HPC workflows for scientific Applications

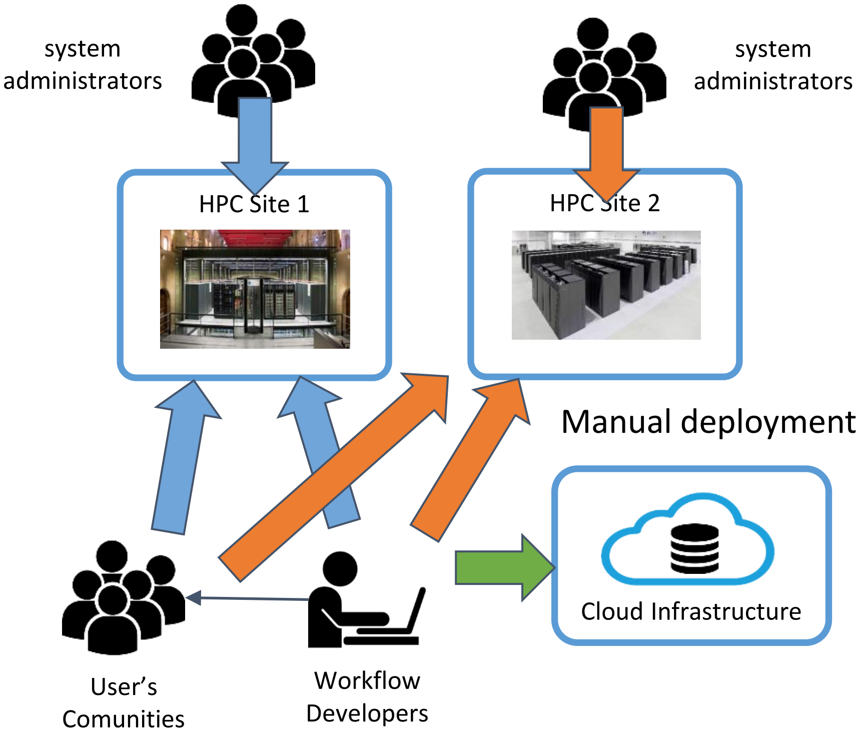
Barcelona, October 25, 2023

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

eFlows4HPC

Current approach



system administrators

system administrators

HPC Site 1

HPC Site 2

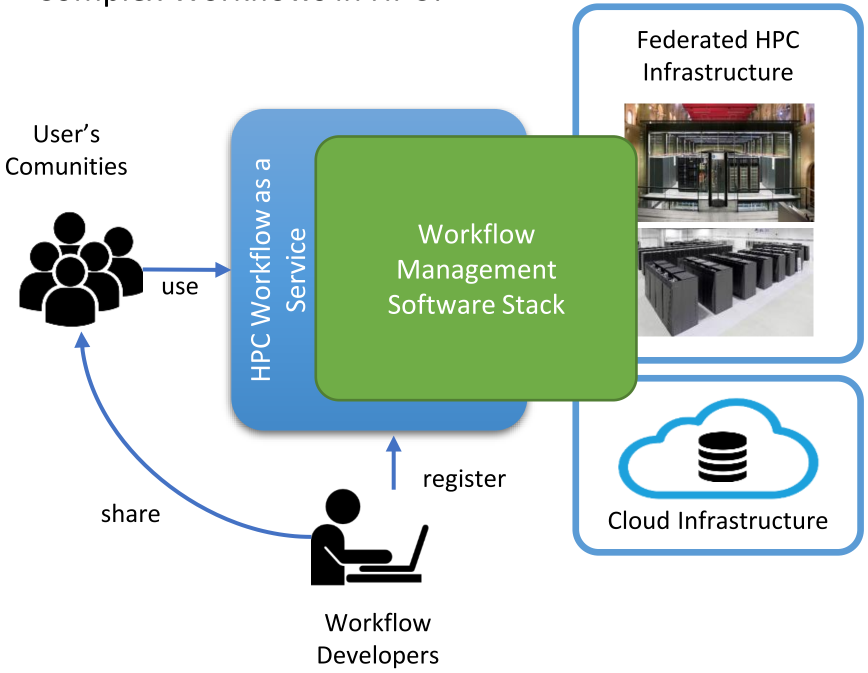
Manual deployment

Cloud Infrastructure

User's Comunities

Workflow Developers

Can we apply something like FaaS for Complex Workflows in HPC?



Federated HPC Infrastructure

User's Comunities

use

HPC Workflow as a Service

Workflow Management Software Stack

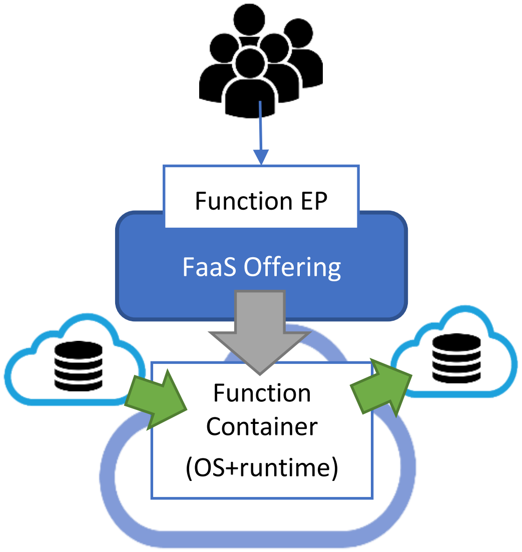
register

share

Cloud Infrastructure

Workflow Developers

FaaS vs. HPCWaaS

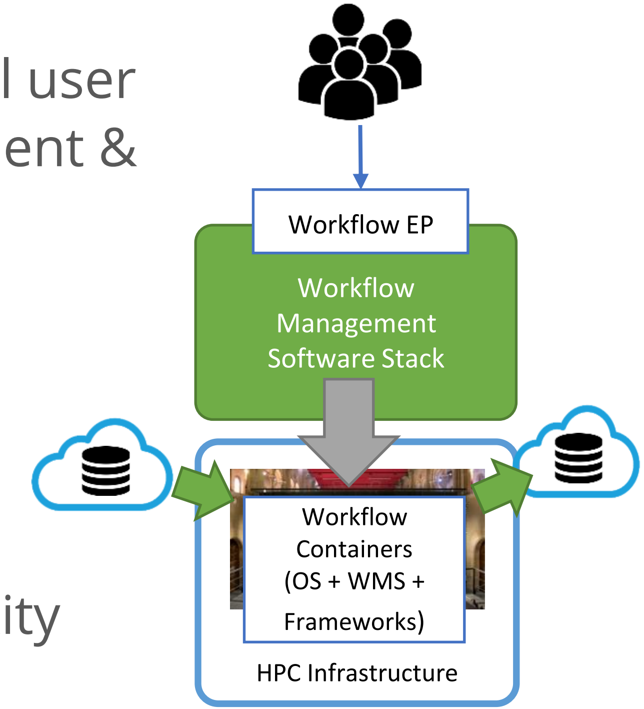


Similarities

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

Differences

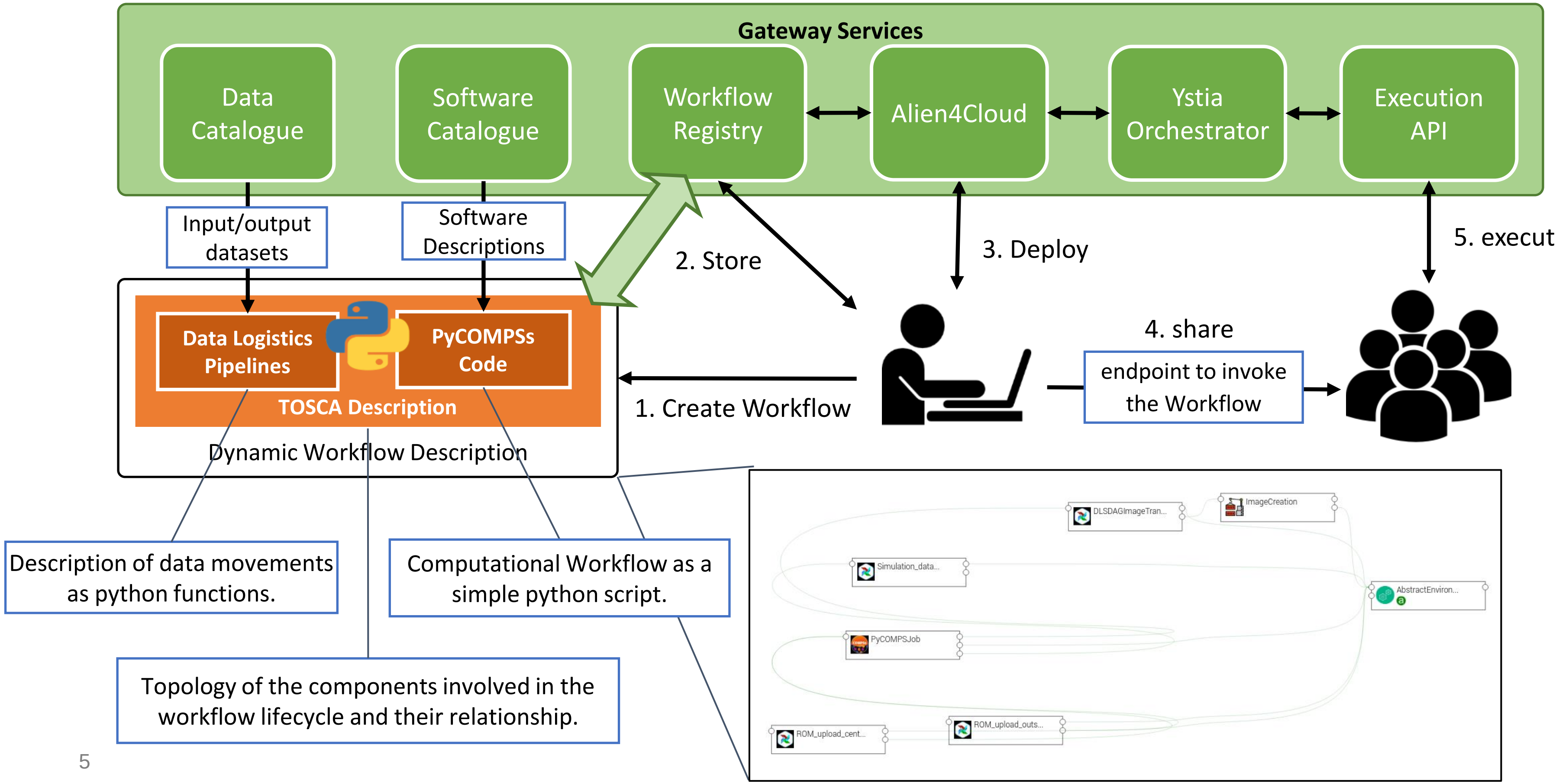
- Restrictions
- Deployment and Execution Complexity
- Performance


Workflow EP
Workflow Management Software Stack
Workflow Containers (OS + WMS + Frameworks)
HPC Infrastructure

ROM Creation Workflow Demo

- Computational Workflow
  - PyCOMPSs Implementation
- Data Management
  - Simulation configuration and meshes 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

Development Overview



Gateway Services

Data Catalogue

Software Catalogue

Workflow Registry

Alien4Cloud

Ystia Orchestrator

Execution API

Input/output datasets

Software Descriptions

2. Store

3. Deploy

5. execute

Data Logistics Pipelines

PyCOMPSs Code

TOSCA Description

Dynamic Workflow Description

1. Create Workflow

4. share

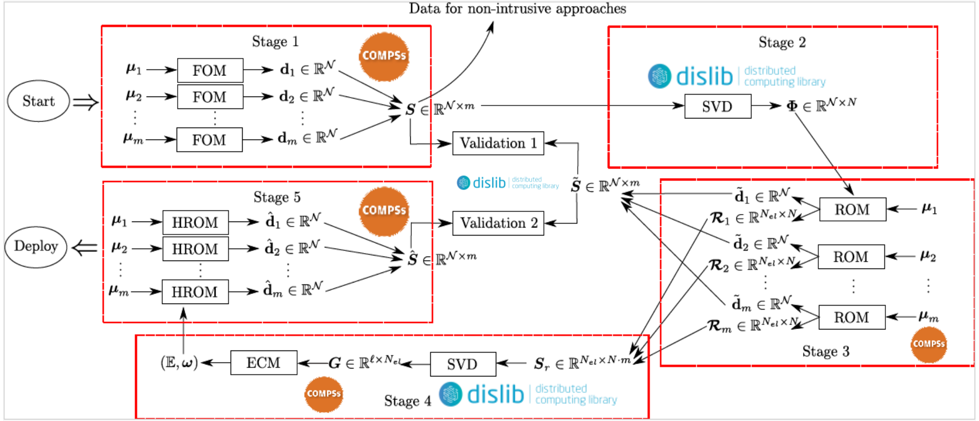
endpoint to invoke the Workflow

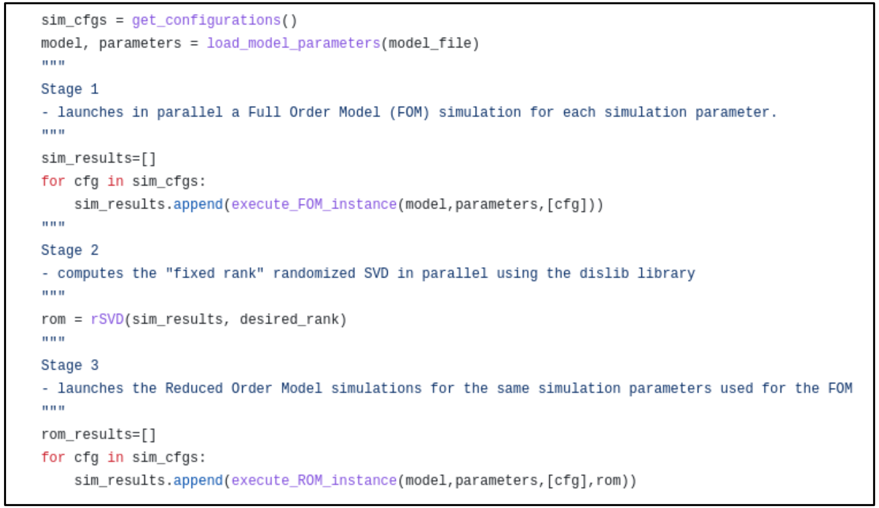
Description of data movements as python functions.

Computational Workflow as a simple python script.

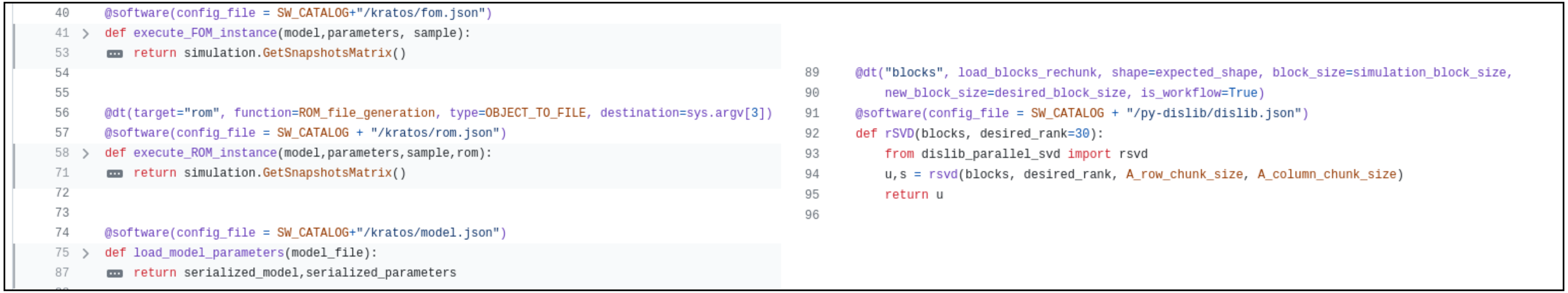
Topology of the components involved in the workflow lifecycle and their relationship.

ROM Creation Workflow Implementation


Data for non-intrusive approaches
Start
Stage 1
COMPSs
μ_1 → FOM → $\mathbf{d}_1 \in \mathbb{R}^{\mathcal{N}}$
μ_2 → FOM → $\mathbf{d}_2 \in \mathbb{R}^{\mathcal{N}}$
μ_m → FOM → $\mathbf{d}_m \in \mathbb{R}^{\mathcal{N}}$
$\boldsymbol{S} \in \mathbb{R}^{\mathcal{N}\times m}$
Validation 1
Stage 2
dislib distributed computing library
SVD
$\boldsymbol{\Phi} \in \mathbb{R}^{\mathcal{N}\times N}$
dislib distributed computing library
$\tilde{\boldsymbol{S}} \in \mathbb{R}^{\mathcal{N}\times m}$
$\tilde{\mathbf{d}}_1 \in \mathbb{R}^{\mathcal{N}}$
$\boldsymbol{\mathcal{R}}_1 \in \mathbb{R}^{N_{el}\times N}$
ROM ← μ_1
$\tilde{\mathbf{d}}_2 \in \mathbb{R}^{\mathcal{N}}$
$\boldsymbol{\mathcal{R}}_2 \in \mathbb{R}^{N_{el}\times N}$
ROM ← μ_2
$\tilde{\mathbf{d}}_m \in \mathbb{R}^{\mathcal{N}}$
$\boldsymbol{\mathcal{R}}_m \in \mathbb{R}^{N_{el}\times N}$
ROM ← μ_m
Stage 3
COMPSs
Stage 5
COMPSs
μ_1 → HROM → $\hat{\mathbf{d}}_1 \in \mathbb{R}^{\mathcal{N}}$
μ_2 → HROM → $\hat{\mathbf{d}}_2 \in \mathbb{R}^{\mathcal{N}}$
μ_m → HROM → $\hat{\mathbf{d}}_m \in \mathbb{R}^{\mathcal{N}}$
$\hat{\boldsymbol{S}} \in \mathbb{R}^{\mathcal{N}\times m}$
Validation 2
Deploy
$(\mathbb{E}, \boldsymbol{\omega})$ ← ECM ← $\boldsymbol{G} \in \mathbb{R}^{\ell\times N_{el}}$ ← SVD ← $\boldsymbol{S}_r \in \mathbb{R}^{N_{el}\times N\cdot m}$
COMPSs
Stage 4
dislib distributed computing library



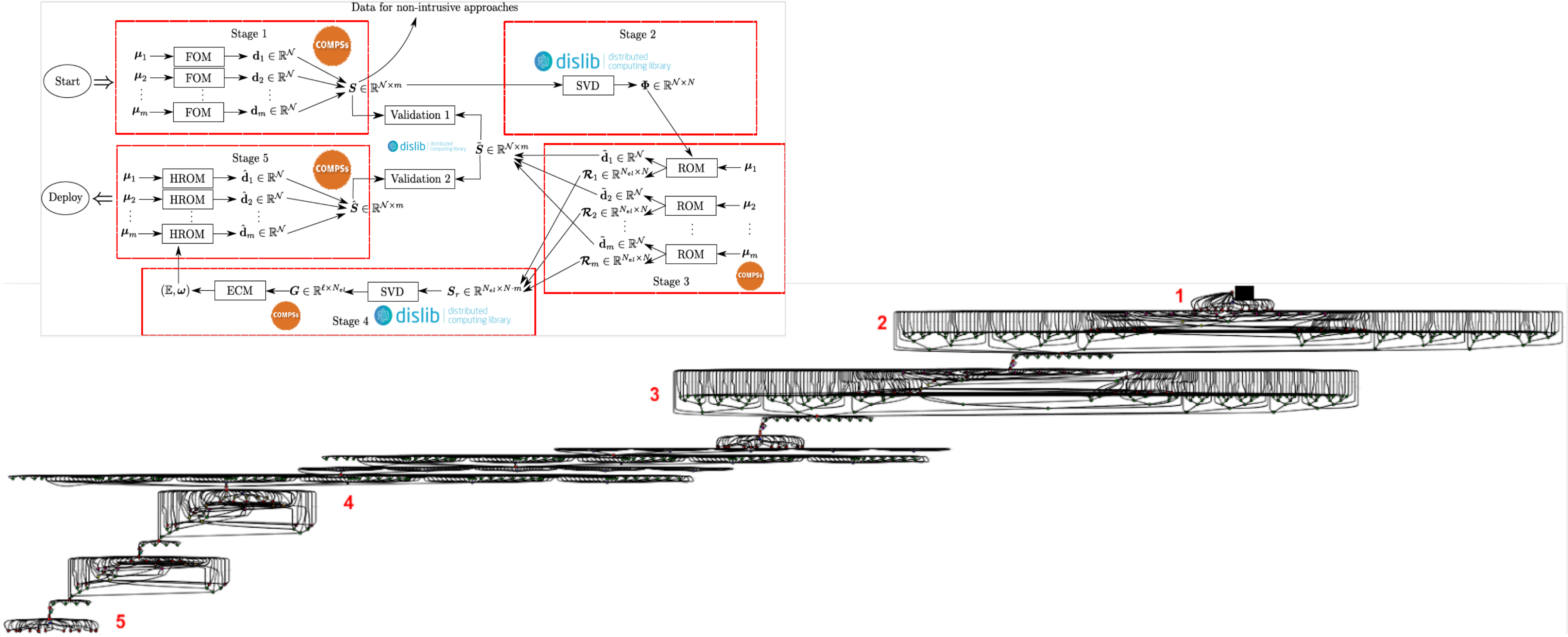
```python
sim_cfgs = get_configurations()
model, parameters = load_model_parameters(model_file)
"""
Stage 1
- launches in parallel a Full Order Model (FOM) simulation for each simulation parameter.
"""
sim_results=[]
for cfg in sim_cfgs:
    sim_results.append(execute_FOM_instance(model,parameters,[cfg]))
"""
Stage 2
- computes the "fixed rank" randomized SVD in parallel using the dislib library
"""
rom = rSVD(sim_results, desired_rank)
"""
Stage 3
- launches the Reduced Order Model simulations for the same simulation parameters used for the FOM
"""
rom_results=[]
for cfg in sim_cfgs:
    sim_results.append(execute_ROM_instance(model,parameters,[cfg],rom))
```



```python
@software(config_file = SW_CATALOG+"/kratos/fom.json")
def execute_FOM_instance(model,parameters, sample):
    return simulation.GetSnapshotsMatrix()


@dt(target="rom", function=ROM_file_generation, type=OBJECT_TO_FILE, destination=sys.argv[3])
@software(config_file = SW_CATALOG + "/kratos/rom.json")
def execute_ROM_instance(model,parameters,sample,rom):
    return simulation.GetSnapshotsMatrix()


@software(config_file = SW_CATALOG+"/kratos/model.json")
def load_model_parameters(model_file):
    return serialized_model,serialized_parameters
```

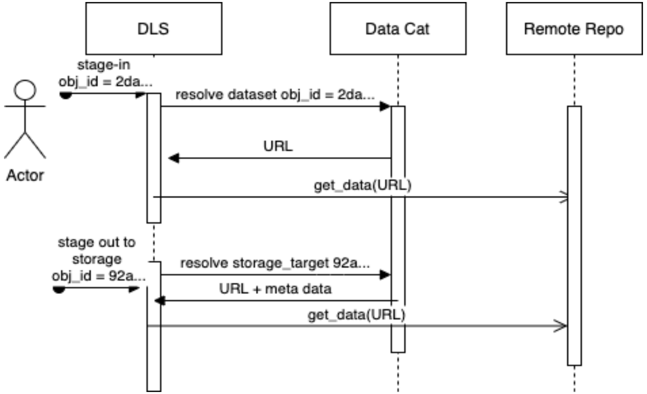
```python
@dt("blocks", load_blocks_rechunk, shape=expected_shape, block_size=simulation_block_size,
    new_block_size=desired_block_size, is_workflow=True)
@software(config_file = SW_CATALOG + "/py-dislib/dislib.json")
def rSVD(blocks, desired_rank=30):
    from dislib_parallel_svd import rsvd
    u,s = rsvd(blocks, desired_rank, A_row_chunk_size, A_column_chunk_size)
    return u

```

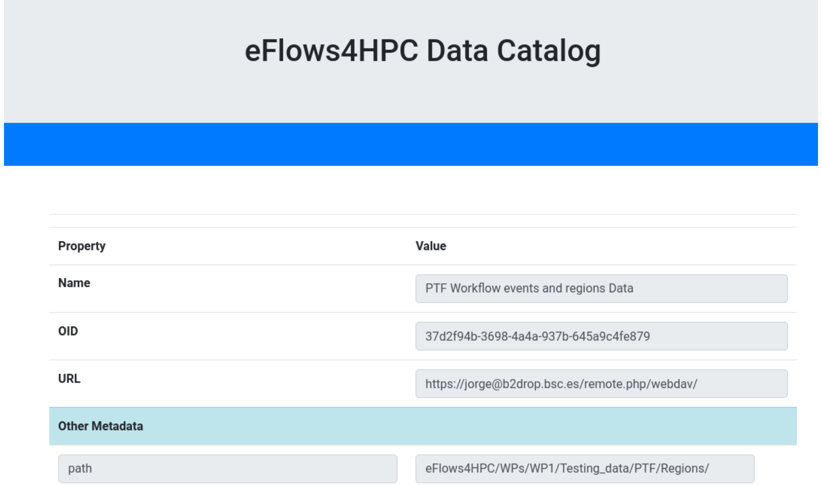
ROM Creation Workflow Implementation


Data for non-intrusive approaches
Start
Stage 1
COMPSs
μ_1 FOM $\mathbf{d}_1 \in \mathbb{R}^{\mathcal{N}}$
μ_2 FOM $\mathbf{d}_2 \in \mathbb{R}^{\mathcal{N}}$
μ_m FOM $\mathbf{d}_m \in \mathbb{R}^{\mathcal{N}}$
$\boldsymbol{S} \in \mathbb{R}^{\mathcal{N} \times m}$
Validation 1
Stage 2
dislib distributed computing library
SVD
$\boldsymbol{\Phi} \in \mathbb{R}^{\mathcal{N} \times N}$
$\tilde{\boldsymbol{S}} \in \mathbb{R}^{\mathcal{N} \times m}$
Stage 3
ROM μ_1 $\tilde{\mathbf{d}}_1 \in \mathbb{R}^{\mathcal{N}}$ $\boldsymbol{\mathcal{R}}_1 \in \mathbb{R}^{N_{el} \times N}$
ROM μ_2 $\tilde{\mathbf{d}}_2 \in \mathbb{R}^{\mathcal{N}}$ $\boldsymbol{\mathcal{R}}_2 \in \mathbb{R}^{N_{el} \times N}$
ROM μ_m $\tilde{\mathbf{d}}_m \in \mathbb{R}^{\mathcal{N}}$ $\boldsymbol{\mathcal{R}}_m \in \mathbb{R}^{N_{el} \times N}$
Stage 4
$\boldsymbol{S}_r \in \mathbb{R}^{N_{el} \times N \cdot m}$
SVD
$\boldsymbol{G} \in \mathbb{R}^{\ell \times N_{el}}$
ECM
$(\mathbb{E}, \boldsymbol{\omega})$
Stage 5
μ_1 HROM $\hat{\mathbf{d}}_1 \in \mathbb{R}^{\mathcal{N}}$
μ_2 HROM $\hat{\mathbf{d}}_2 \in \mathbb{R}^{\mathcal{N}}$
μ_m HROM $\hat{\mathbf{d}}_m \in \mathbb{R}^{\mathcal{N}}$
$\hat{\boldsymbol{S}} \in \mathbb{R}^{\mathcal{N} \times m}$
Validation 2
Deploy
1
2
3
4
5

Data pipelines

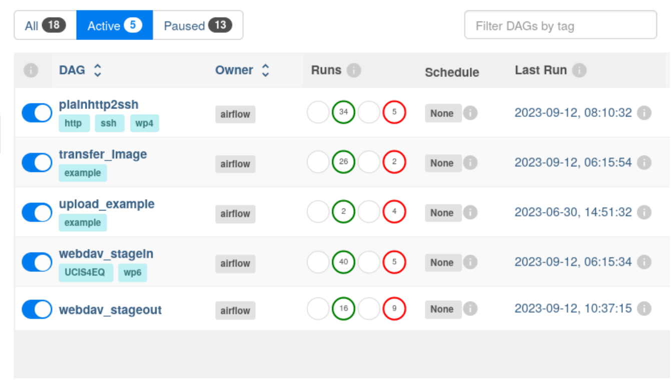
- Implemented in Data Logistics Service
- Reusable for multiple data/workflows
- Configured from Data Catalogue





eFlows4HPC Data Catalog

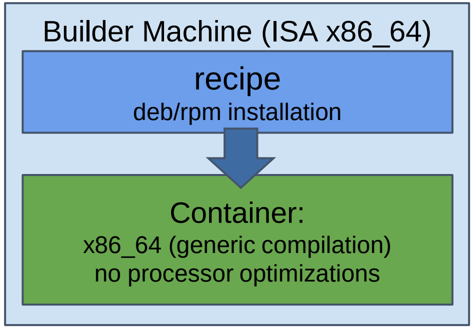
Property	Value
Name	PTF Workflow events and regions Data
OID	37d2f94b-3698-4a4a-937b-645a9c4fe879
URL	https://jorge@b2drop.bsc.es/remote.php/webdav/
Other Metadata	
path	eFlows4HPC/WPs/WP1/Testing_data/PTF/Regions/



DAG	Owner	Runs	Schedule	Last Run
plainhttp2ssh http ssh wp4	airflow	34 5	None	2023-09-12, 08:10:32
transfer_Image example	airflow	26 2	None	2023-09-12, 06:15:54
upload_example example	airflow	2 4	None	2023-06-30, 14:51:32
webdav_stageIn UCIS4EQ wp6	airflow	40 5	None	2023-09-12, 06:15:34
webdav_stageout	airflow	16 9	None	2023-09-12, 10:37:15

Containers and HPC

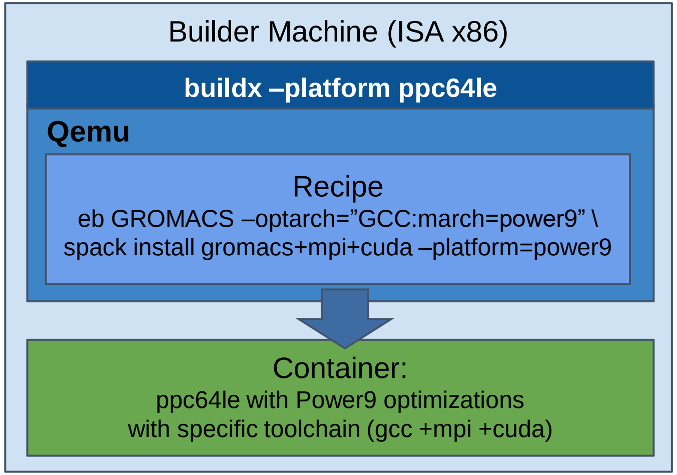
Standard container image creation


Builder Machine (ISA x86_64)
recipe
deb/rpm installation
Container:
x86_64 (generic compilation)
no processor optimizations

- **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**

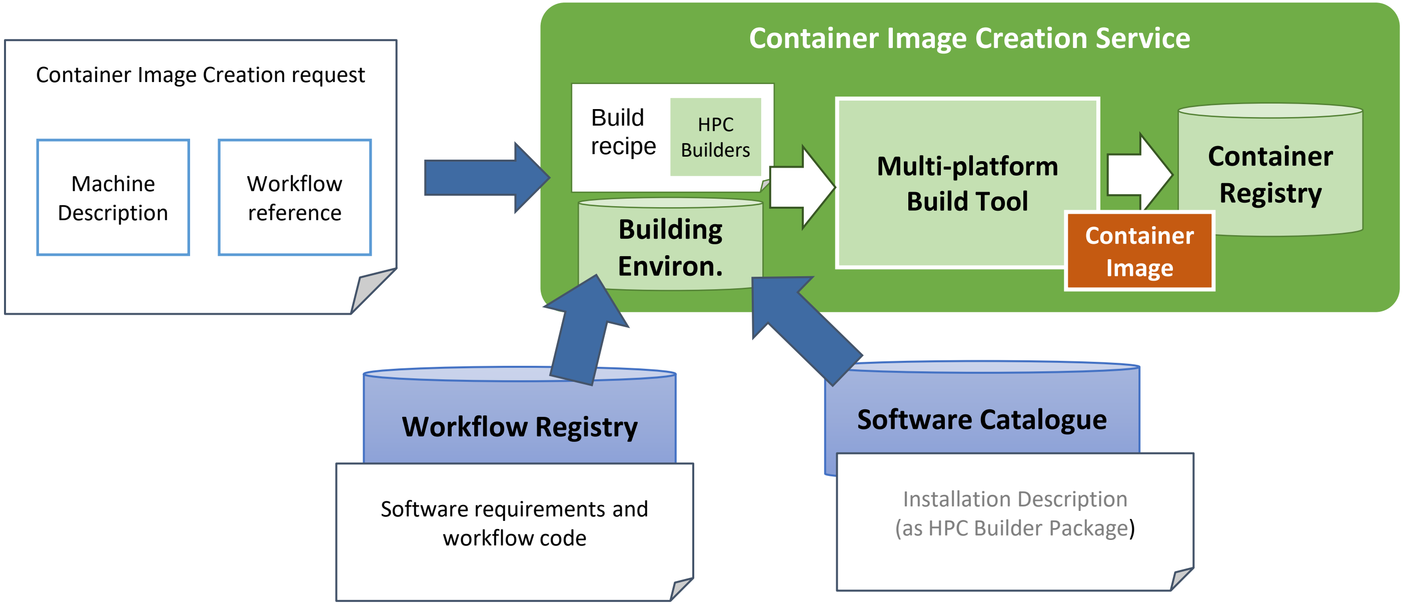
HPC Ready Containers

eFlows4HPC approach


Builder Machine (ISA x86)
buildx –platform ppc64le
Qemu
Recipe
eb GROMACS –optarch=”GCC:march=power9” \
spack install gromacs+mpi+cuda –platform=power9
Container:
ppc64le with Power9 optimizations
with specific toolchain (gcc +mpi +cuda)

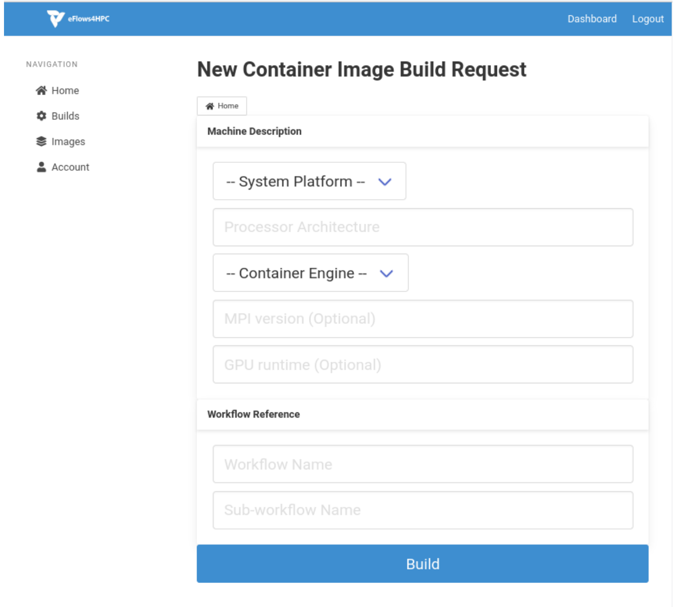
- **Methodology to allow the creation containers for specific HPC system**
  - Leverage HPC and Multi-platform container builders
- **It is tight to do by hand but let’s automate!**

Container Image Creation Service


Container Image Creation request
Machine Description
Workflow reference
Container Image Creation Service
Build recipe
HPC Builders
Building Environ.
Multi-platform Build Tool
Container Image
Container Registry
Workflow Registry
Software requirements and workflow code
Software Catalogue
Installation Description (as HPC Builder Package)

Container Image Creation Service

- Web Interface

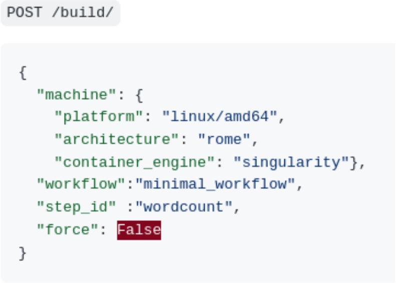


New Container Image Build Request

Machine Description

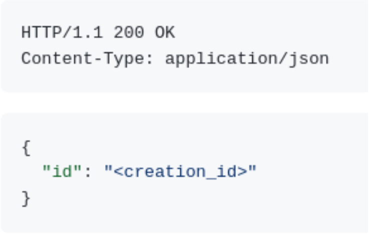
Workflow Reference

- REST Interface and CLI



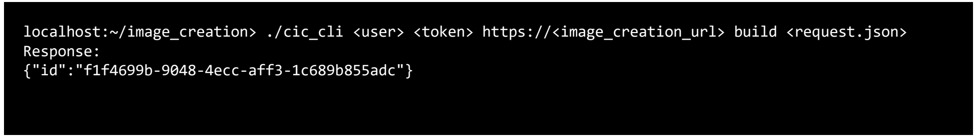
```
POST /build/
```

```
{
  "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>"
}
```

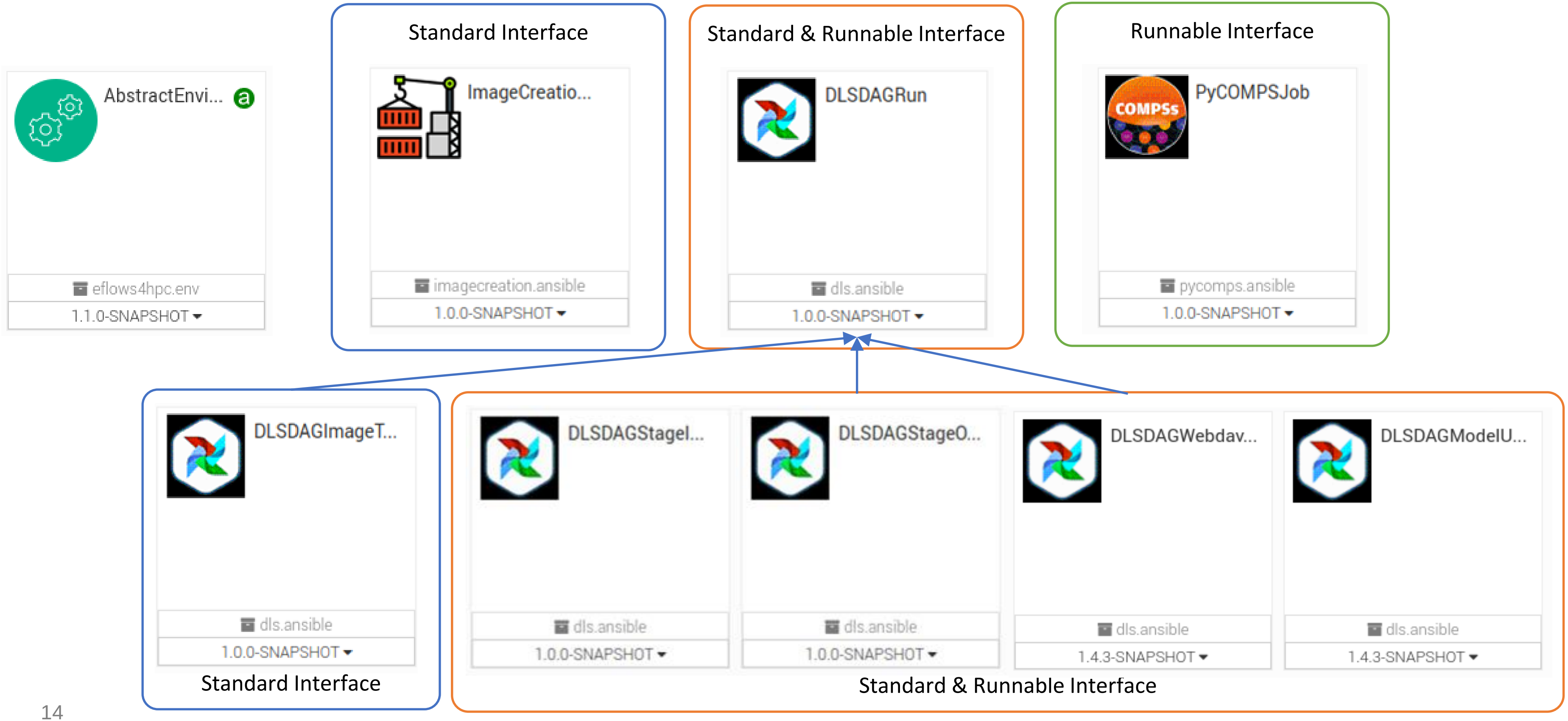


```
localhost:~/image_creation> ./cic_cli <user> <token> https://<image_creation_url> build <request.json>
Response:
{"id":"f1f4699b-9048-4ecc-aff3-1c689b855adc"}
```

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



Standard Interface

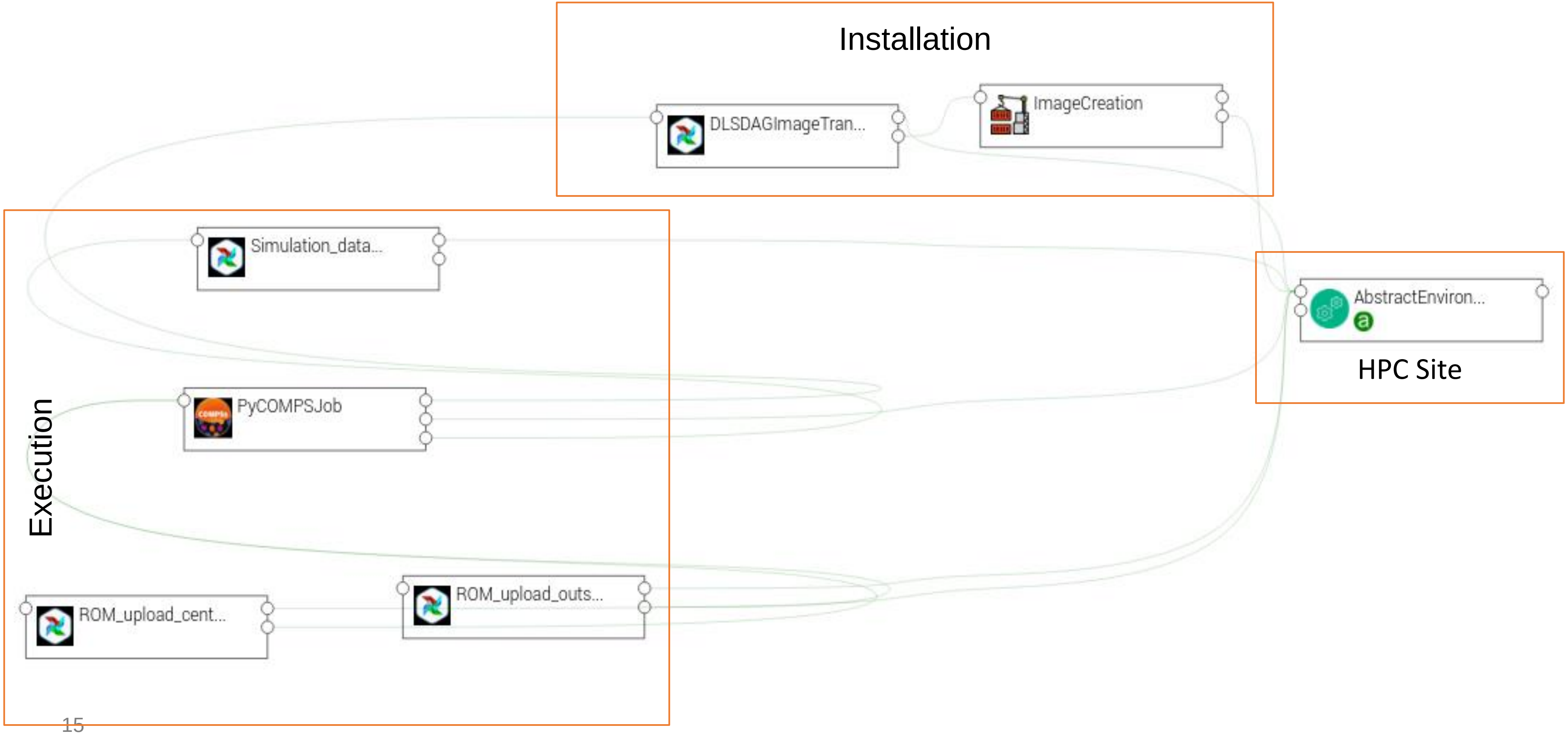
Standard & Runnable Interface

Runnable Interface

Standard Interface

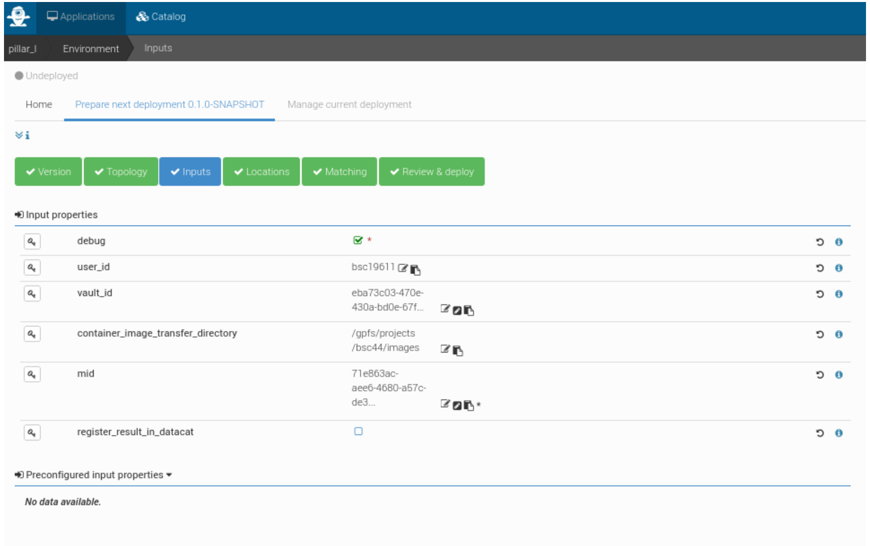
Standard & Runnable Interface

TOSCA Modelization



Workflow Deployment (done once per HPC site)

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



Applications | Catalog

pillar_I | Environment | Inputs

Undeployed

Home | Prepare next deployment 0.1.0-SNAPSHOT | Manage current deployment

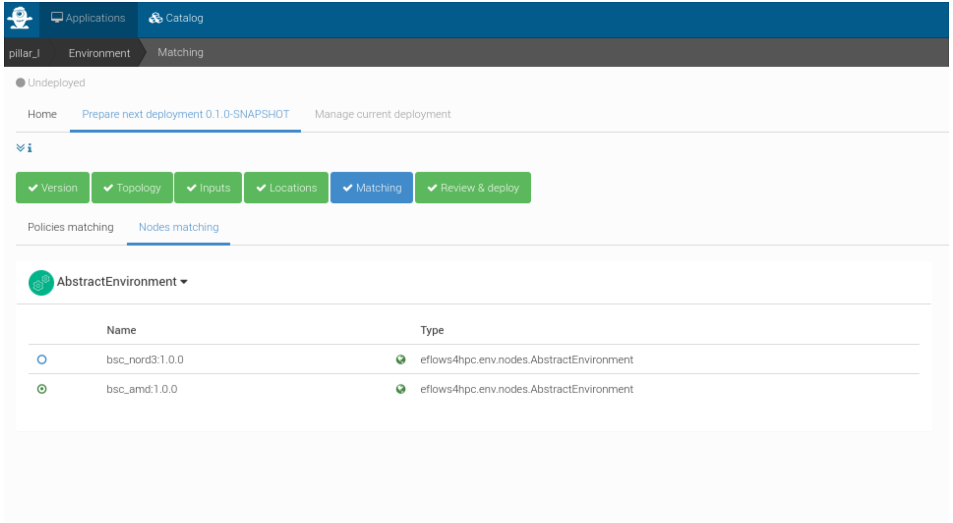
✓ Version | ✓ Topology | ✓ Inputs | ✓ Locations | ✓ Matching | ✓ Review & deploy

Input properties

debug	☑ *
user_id	bsc19611
vault_id	eba73c03-470e-430a-bd0e-67f...
container_image_transfer_directory	/gpfs/projects /bsc44/images
mid	71e863ac-aee6-4680-a57c-de3... *
register_result_in_datacat	□

Preconfigured input properties ▾

No data available.



Applications | Catalog

pillar_I | Environment | Matching

Undeployed

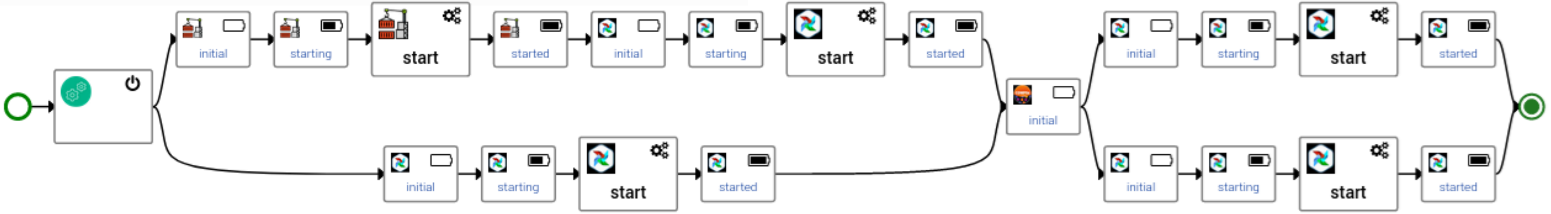
Home | Prepare next deployment 0.1.0-SNAPSHOT | Manage current deployment

✓ Version | ✓ Topology | ✓ Inputs | ✓ Locations | ✓ Matching | ✓ Review & deploy

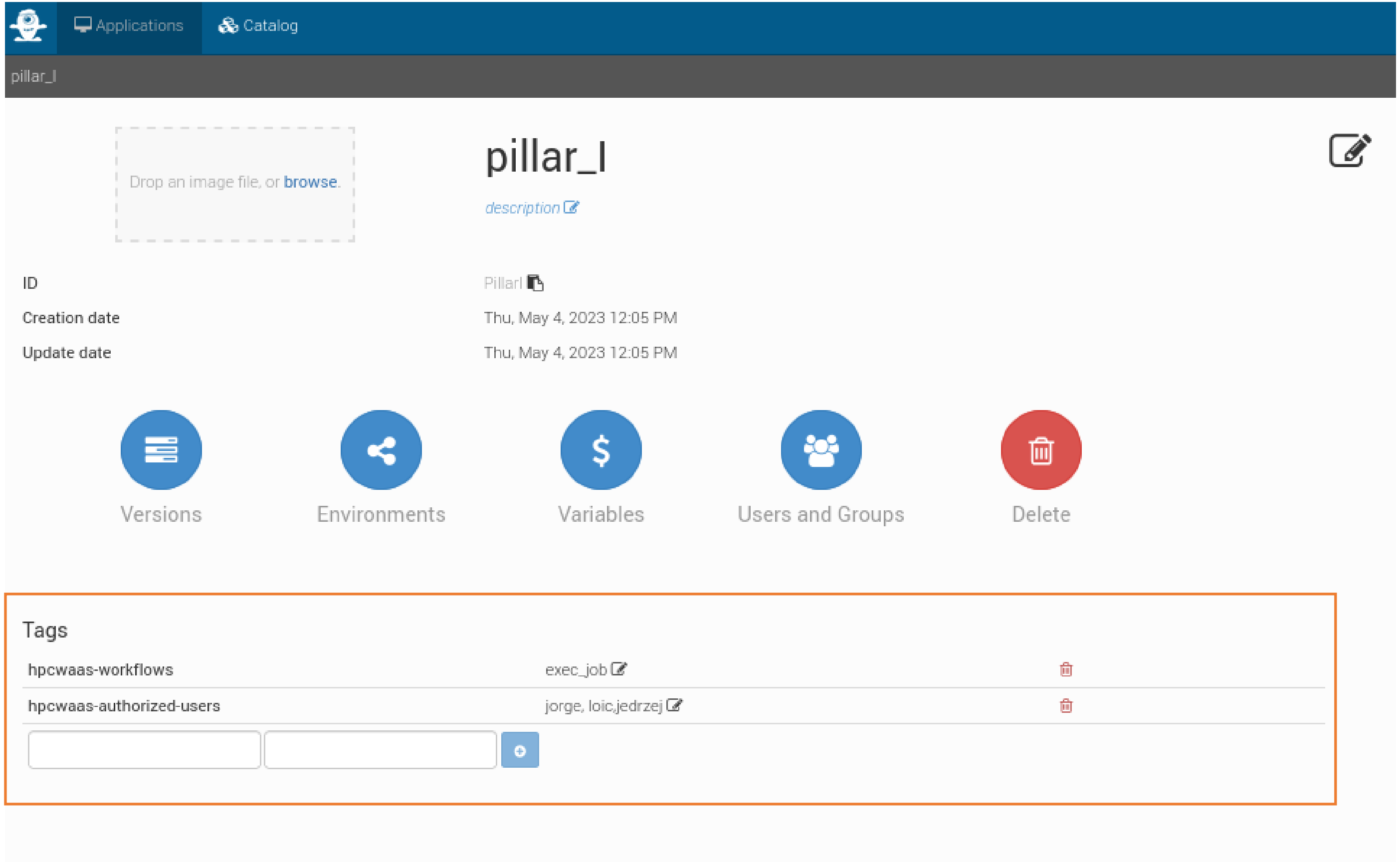
Policies matching | Nodes matching

AbstractEnvironment ▾

	Name	Type
○	bsc_nord3:1.0.0	eflows4hpc.env.nodes.AbstractEnvironment
●	bsc_amd:1.0.0	eflows4hpc.env.nodes.AbstractEnvironment



Publish workflow and authorize users



pillar_I

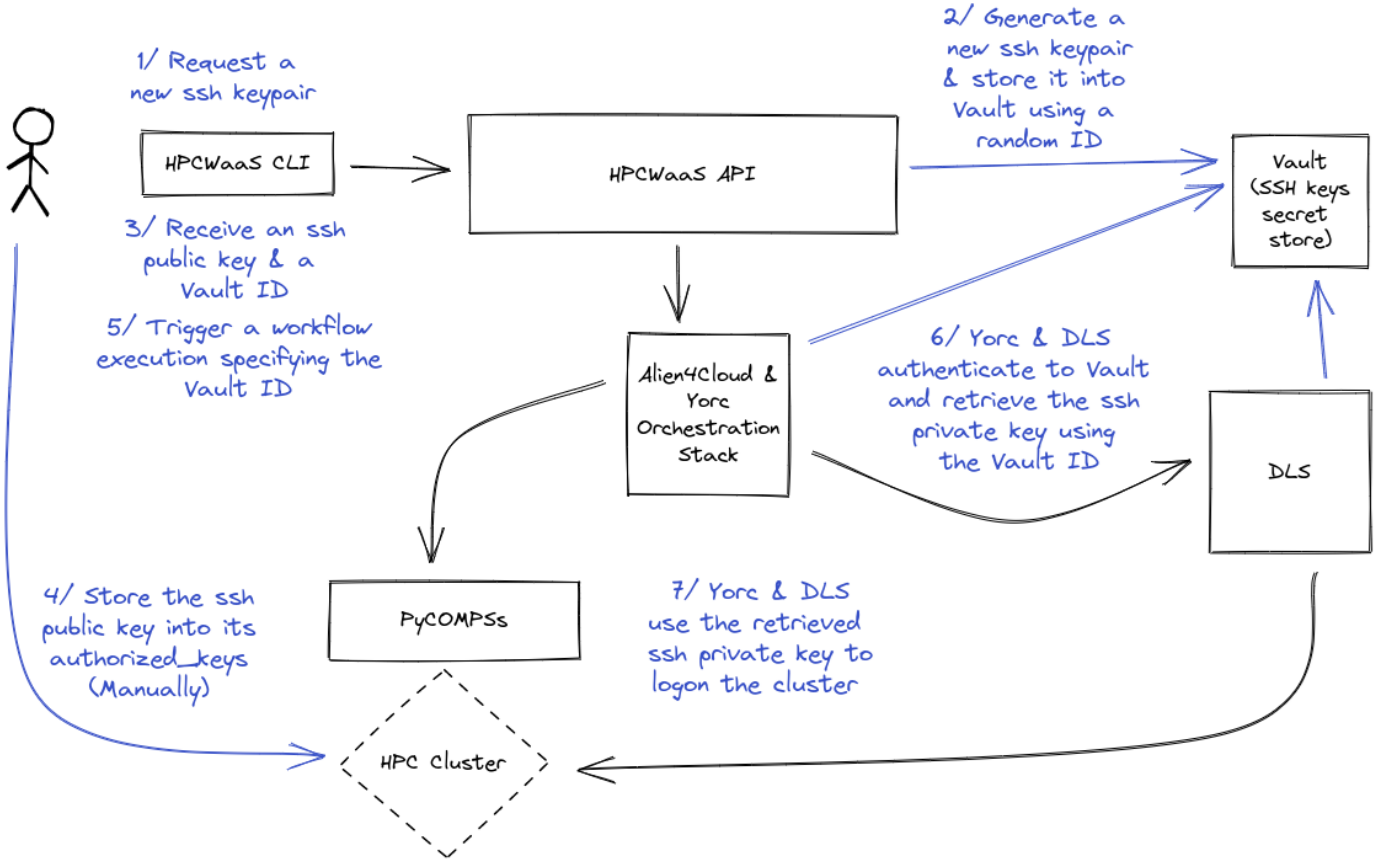
description

ID	PillarI
Creation date	Thu, May 4, 2023 12:05 PM
Update date	Thu, May 4, 2023 12:05 PM

Tags

hpcwaas-workflows	exec_job
hpcwaas-authorized-users	jorge, loic,jedrzej

Workflow Execution End user


1/ Request a new ssh keypair
HPCWaaS CLI
HPCWaaS API
2/ Generate a new ssh keypair & store it into Vault using a random ID
Vault (SSH keys secret store)
3/ Receive an ssh public key & a Vault ID
5/ Trigger a workflow execution specifying the Vault ID
Alien4Cloud & Yorc Orchestration Stack
6/ Yorc & DLS authenticate to Vault and retrieve the ssh private key using the Vault ID
DLS
4/ Store the ssh public key into its authorized_keys (Manually)
PyCOMPSs
7/ Yorc & DLS use the retrieved ssh private key to logon the cluster
HPC Cluster

Thank you

eFlows4HPC

Enabling dynamic and Intelligent workflows in the future EuroHPC ecosystem

www.eFlows4HPC.eu

@eFlows4HPC   in 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.