

FirecREST: a common interface for HPC and AI workflows

Eirini Koutsaniti and Elia Palme

CSCS - Swiss National Supercomputing Centre

Lugano

January, 2026

Introducing FirecREST

FirecREST in a nutshell

- **FirecREST is an open-source web API to access HPC resources**



FirecREST in a nutshell

- **FirecREST is an open-source web API to access HPC resources**
- **Standard/common API**
 - Abstracts the underlying HPC technology
 - Scheduler
 - Filesystem
 - Storage
 - Based on RESTful design principles (HTTP verbs)



FirecREST in a nutshell

- **FirecREST is an open-source web API to access HPC resources**
- Standard/common API
- Web interface for classic HPC
 - Integrates web standards (OAuth2, REST, OpenAPI, etc.)
 - Accessible via HTTP/HTTPS



FirecREST in a nutshell

- **FirecREST is an open-source web API to access HPC resources**
- Standard/common API
- Web interface for classic HPC
- Modular and lightweight architecture
 - Extremely lightweight and modern stack
 - Stateless does not require any persistent storage
 - Proxy based architecture with high throughput performance
 - Modules types:
 - Auth modules (Keycloak, GoogleID, Microsoft, Shibboleth, etc.)
 - Scheduler modules (Slurm, openPBS, etc.)
 - Storage modules (S3, Filesystem, Data streamer, etc.)



FirecREST in a nutshell

- **FirecREST is an open-source web API to access HPC resources**
- Standard/common API
- Web interface for classic HPC
- Modular and lightweight architecture
- Authentication and Authorization
 - Integrates with various Identity Providers for authorization
 - Provides granular resource access authorization



How FirecREST enables modern workflows on HPC

Use Cases - HPC and AI workflows integration

Execute scientific and AI workflows on HPC infrastructure accessing compute and data resources.

- FirecREST
 - Provides secure and reliable access between the workflow engine and the HPC resources
 - Uses standard technology HTTP REST API
 - Facilitate workflows execution across different sites

Example - FirecREST enabled Airflow for AI workflows

The image displays two overlapping screenshots of the Apache Airflow web interface. The top screenshot shows the 'List Dag Run' page with a table of DAG runs. A large grey arrow points from a row in this table to the bottom screenshot. The bottom screenshot shows the detailed view of a DAG run for 'firecrest_example' on 2024-03-01 at 01:00:00 CET. It includes a task graph, a task duration bar chart, and a list of task statuses.

State	Dag Id	Logical Date	Run Id
running	firecrest_example	2024-03-01, 19:49:54	manual_2024-01T18:49:54.99

DAG: firecrest_example
Schedule: @daily | Next Run: 2024-03-01, 01:00:00

Task Graph (Left to Right):

- wait-for-file (FileSensor) [success]
- upload-in (FirecRESTUploadOperator) [success]
- upload-pp (FirecRESTUploadOperator) [success]
- job-submit (FirecRESTSubmitOperator) [success]
- download-out (FirecRESTDownloadOperator) [success]
- log-results (BashOperator) [success]
- remove-struct (BashOperator) [success]

Task Duration Bar Chart (Mar 01, 01:00):

- wait-for-file: ~00:00:01
- upload-in: ~00:00:01
- upload-pp: ~00:00:01
- job-submit: ~00:00:01
- download-out: ~00:00:01
- log-results: ~00:00:01
- remove-struct: ~00:00:01



 eth-cscs.github.io/firecrest-v2/use_cases/workflow-orchestrator

Use Cases – Custom User Interfaces

Build desktop/web GUI tailored to your HPC/AI workloads

- With FirecREST
 - Enables user authentication over web standards OIDC – OAuth 2.0
 - Provides a secure and reliable channel from web to HPC clusters
 - Provides a simple method (HTTP REST API) to execute jobs
 - **Can be easily integrated into web UIs as it uses standard web technology**

Example – HPC Console (a.k.a. FirecREST-UI)

The screenshot displays the ML Console interface for the 'csstaff' project, managed by 'Elia Palme (palmee)'. The main view is the 'Job Scheduler' for 'csstaff', showing a list of jobs. A 'Submit new Job' button is visible in the top right of the scheduler area.

Status	Job	User
COMPLETED	nccl-vetting.sh Job id: 1403915	palmee
COMPLETED	nccl-vetting.sh Job id: 1403911	palmee
FAILED	nccl-vetting.sh Job id: 1403900	palmee
TIMEOUT	nccl-vetting.sh Job id: 1403858	palmee
COMPLETED	nccl-vetting.sh Job id: 1403792	palmee

The detailed view shows the 'Job StdOut' for a job. The output includes system messages and test results:

```
[notice] To update, run: python3.12 -m pip install --upgrade pip
[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python3.12 -m pip install --upgrade pip
[vetnode] diagnose
Running sanity checks: ML Vetting on node:nid007642
[vetnode] diagnose
Running sanity checks: ML Vetting on node:nid007616

** Results: **

[32mNode: nid007616 result:test_name='NCCL-Low-Level'
test_type='vetnode.evaluations.nccl_lib_eval.NcclLibEval' passed=True
elapsedtime=6.400349378585815 metadata={'bandwidth': ' 17.81 GB/s'} [0m
[32mNode: nid007616 result:test_name='NCCL-PyTorch'
test_type='vetnode.evaluations.nccl_pytorch_eval.NcclPytorchEval' passed=True
elapsedtime=11.325170278549194 metadata={'bandwidth': ' 21.42 GB/s', 'rank': 0, 'local_rank': 0,
'world_size': 2, 'mesurment_matrix': []} [0m
Vetted: nid007616

** Results: **

[32mNode: nid007642 result:test_name='NCCL-Low-Level'
test_type='vetnode.evaluations.nccl_lib_eval.NcclLibEval' passed=True
elapsedtime=6.9221930503845215 metadata={'bandwidth': ' 17.68 GB/s'} [0m
[32mNode: nid007642 result:test_name='NCCL-PyTorch'
test_type='vetnode.evaluations.nccl_pytorch_eval.NcclPytorchEval' passed=True
elapsedtime=11.848727464675903 metadata={'bandwidth': ' 21.42 GB/s', 'rank': 1, 'local_rank': 0,
'world_size': 2, 'mesurment_matrix': []} [0m
Vetted: nid007642
```

The 'Job details' panel on the right shows:

- Job ID: #1403915
- Name: nccl-vetting.sh
- Status: COMPLETED
- Submitted by: palmee
- Account: csstaff
- Execution times: Start time 20-01-2026 18:20:22, Start time 20-01-2026 18:20:59, Execution time 00:00:37
- Files: StdOut /users/palme/S/SD-68370/slurm-1403915.out
- StdErr: N/A
- StdIn: /dev/null

© 2026 CSCS - All rights reserved. Version 2.4.2

Use Cases – CI/CD pipelines for HPC

CI/CD pipelines are used to automate deployment of scientific software

- With FirecREST
 - Provides secure OAuth 2.0 authentication (no risk to expose user credentials)
 - Enables a fast and reliable connection into HPC infrastructure
 - **Portable across pipeline engines (GitLab CI, GitHub Actions, etc.) and HPC infrastructures**

Example – FirecREST enabled Alps Image Deployment

Project

- gitlab-runner-firecrest-default
- Manage
- Plan
- Code
- Build
- Pipelines**
- Jobs
- Pipeline schedules
- Artifacts
- Deploy
- Operate
- Monitor
- Analyze

renamed CRAY_CUDA_MPS to CSCS_CUDA_MPS

Passed CSCS Ciext created pipeline for commit e115dfa5 5 days ago, finished 5 days ago

For main

21 jobs 28 minutes 1 second, queued for 2 seconds

Pipeline Jobs 21 Tests 0

Group jobs by Stage Job dependencies

build	test	deploy
<ul style="list-style-type: none">build_base_serverbuild_glrbuild_glr-f7t-implbuild_helper_utils_aarch64build_helper_utils_x86_64build_server	<ul style="list-style-type: none">test-f7t-controllertest-f7t-cred-overridetest-long-runningtest-reframe-runnertest-uenv-builder-gh200test-uenv-builder-zen2test-uenv-runner-gh200 jfrogtest-uenv-runner-gh200 uenv-buildertest-uenv-runner-zen2 jfrogtest-uenv-runner-zen2 uenv-buildertest_baremetaltest_container-builder_aarch64test_container-builder_x86_64test_container-runner	<ul style="list-style-type: none">deploy to jfrog



 eth-cscs.github.io/firecrest-v2/use_cases/CI-pipeline

Use Cases - Quality of Service via Continuous Regression Testing

Execute periodic testing against the HPC infrastructure to validate performance and catch QoS issues.

- With FirecREST
 - Allows remote access to HPC infrastructure over HTTP
 - Regression tests can be executed from laptops, pipelines, cloud, etc.
 - Facilitates tests portability across different sites and infrastructures
 - **Simple development using Python (pyFirecREST)**

Example – FirecREST enabled ReFrame Regression Testing

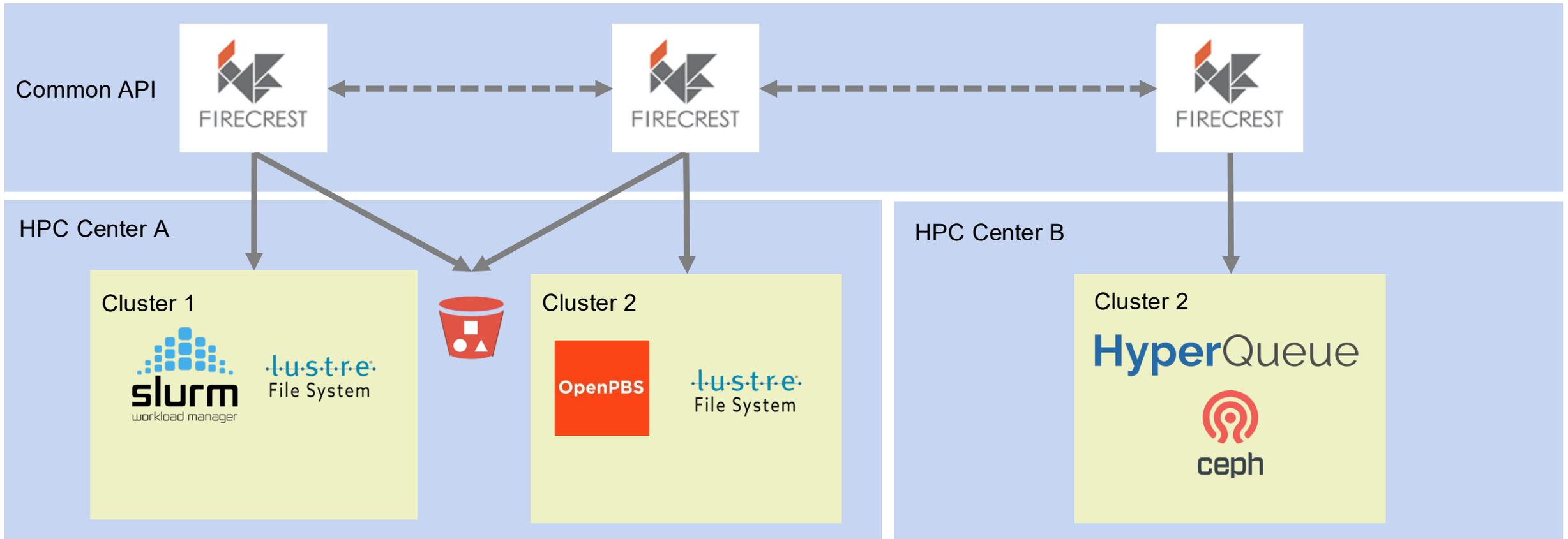
```
518 [ FAIL ] (134/137) MemoryOverconsumptionMpiCheck /6a7583af @clariden:nvgpu+PrgEnv-gnu
519 P: cn_avail_memory_from_sysconf: 482 GB (r:0, l:None, u:None)
520 P: cn_max_allocated_memory: 472 GB (r:497, l:-0.05, u:None)
521 ==> test failed during 'performance': test staged in '/builds/ekoutsaniti/clariden-testing-pc
age/2024-03-05_04-06-05/clariden/nvgpu/PrgEnv-gnu/MemoryOverconsumptionMpiCheck'
522 [ FAIL ] (135/137) MemoryOverconsumptionMpiCheck /6a7583af @clariden:nvgpu+PrgEnv-nvidia
523 P: cn_avail_memory_from_sysconf: 482 GB (r:0, l:None, u:None)
524 P: cn_max_allocated_memory: 471 GB (r:497, l:-0.05, u:None)
525 ==> test failed during 'performance': test staged in '/builds/ekoutsaniti/clariden-testing-pc
age/2024-03-05_04-06-05/clariden/nvgpu/PrgEnv-nvidia/MemoryOverconsumptionMpiCheck'
526 [ OK ] (136/137) MemoryOverconsumptionMpiCheck /6a7583af @clariden:amdgpu+PrgEnv-cray
527 P: cn_avail_memory_from_sysconf: 457 GB (r:0, l:None, u:None)
528 P: cn_max_allocated_memory: 484 GB (r:497, l:-0.05, u:None)
529 [ OK ] (137/137) MemoryOverconsumptionMpiCheck /6a7583af @clariden:amdgpu+PrgEnv-gnu
530 P: cn_avail_memory_from_sysconf: 465 GB (r:0, l:None, u:None)
531 P: cn_max_allocated_memory: 484 GB (r:497, l:-0.05, u:None)
532 [-----] all spawned checks have finished
533 [=====] Retrying 1 failed check(s) (retry 1/2)
534 [-----] start processing checks
535 [ RUN ] MemoryOverconsumptionMpiCheck /6a7583af @clariden:nvgpu+PrgEnv-gnu
536 [ RUN ] MemoryOverconsumptionMpiCheck /6a7583af @clariden:nvgpu+PrgEnv-nvidia
537 [ OK ] (1/2) MemoryOverconsumptionMpiCheck /6a7583af @clariden:nvgpu+PrgEnv-gnu
538 P: cn_avail_memory_from_sysconf: 480 GB (r:0, l:None, u:None)
539 P: cn_max_allocated_memory: 473 GB (r:497, l:-0.05, u:None)
```



FirecREST enables Federation

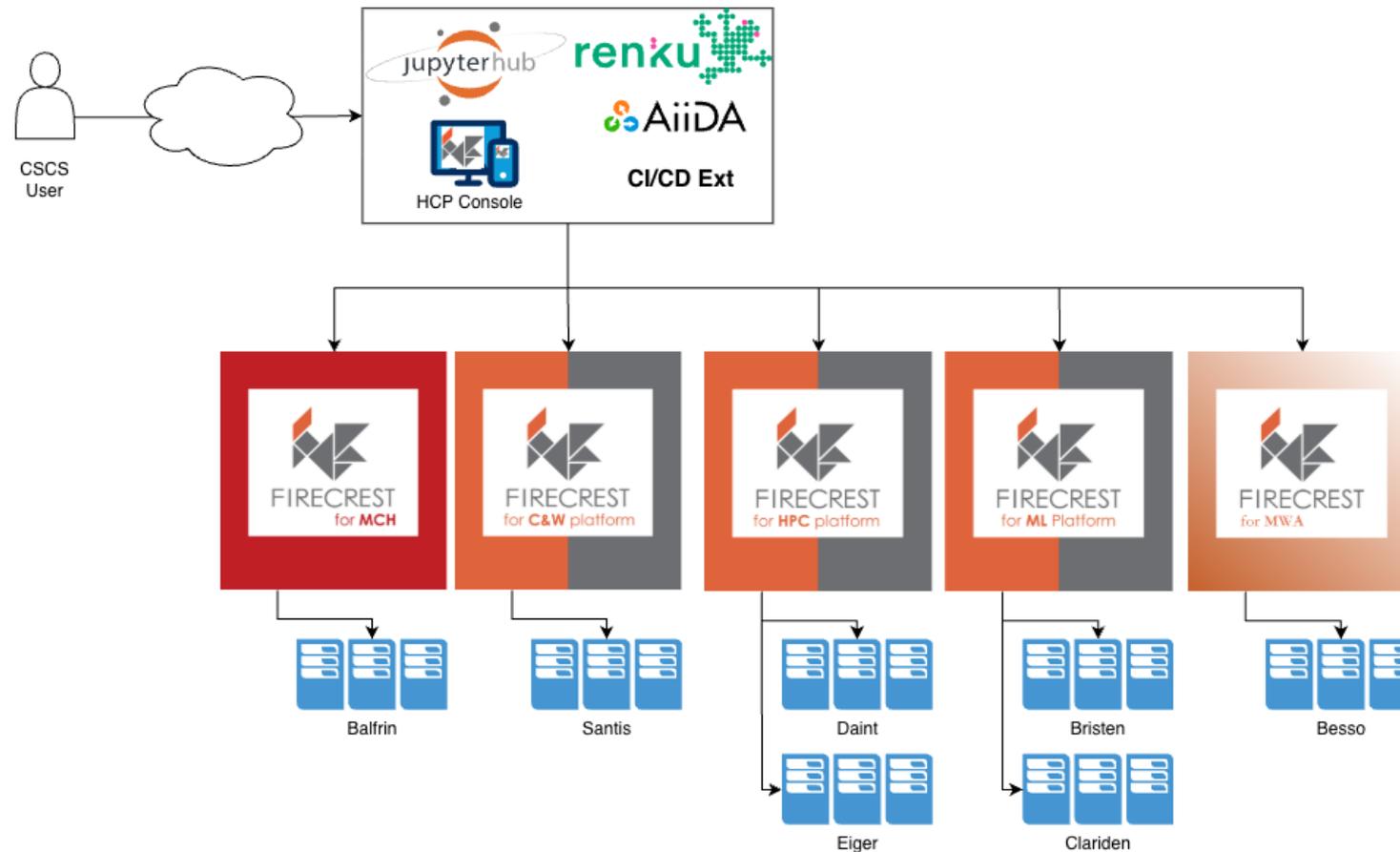
Federation with FirecREST

FirecREST provides a standardized interface across different sites and set of HPC technologies.



Example – FirecREST provides a common interface for Alps

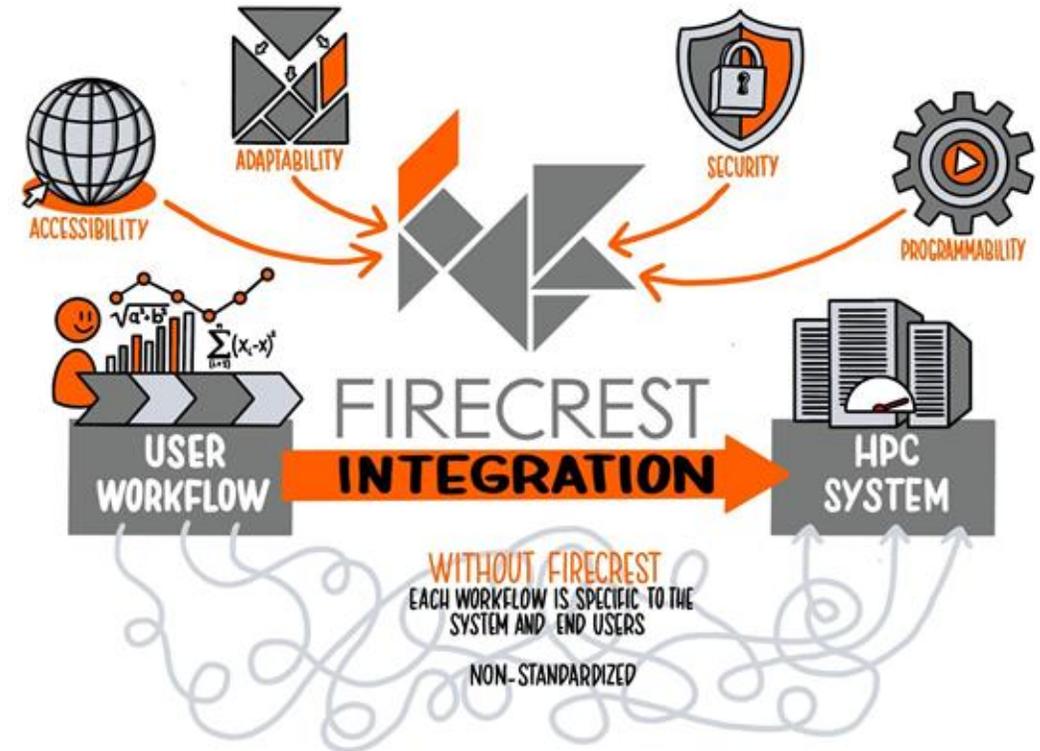
- Alps serves diverse AI, Climate, and HPC workflows from research and industry
- FirecREST is a key interface of the 4 main platforms and their exposed services



Conclusions

Conclusions

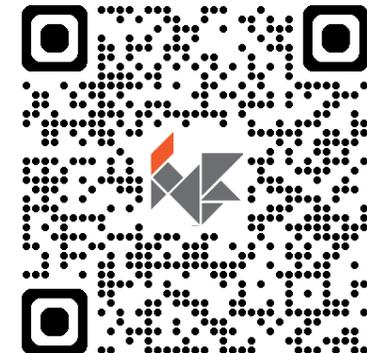
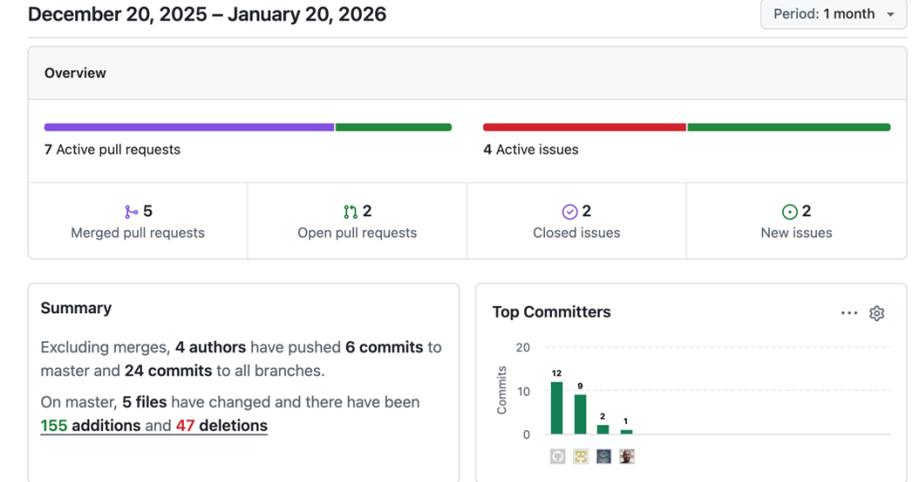
- FirecREST abstracts HPC technologies providing a standardized interface
- FirecREST acts as a proxy enabling web access to HPC infrastructures
- FirecREST enables HPC infrastructure federations across sites for AI and HPC workflows
- FirecREST's modern design, built on widely adopted standards and a modular architecture, enables easy extensibility



Links and references



- More on Firecrest
 - API Reference: eth-cscs.github.io/firecrest-v2/openapi
 - Firecrest: github.com/eth-cscs/firecrest-v2
 - pyFirecrest: github.com/eth-cscs/pyfirecrest
 - Firecrest Web UI: github.com/eth-cscs/firecrest-ui
 - Join our community on Slack: firecrest-community.slack.com
 - Contact us: firecrest@cscs.ch

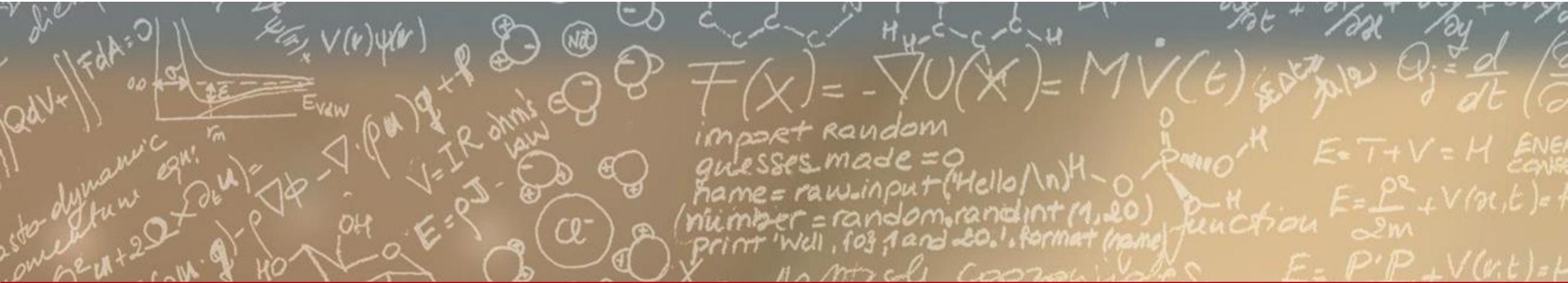




CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



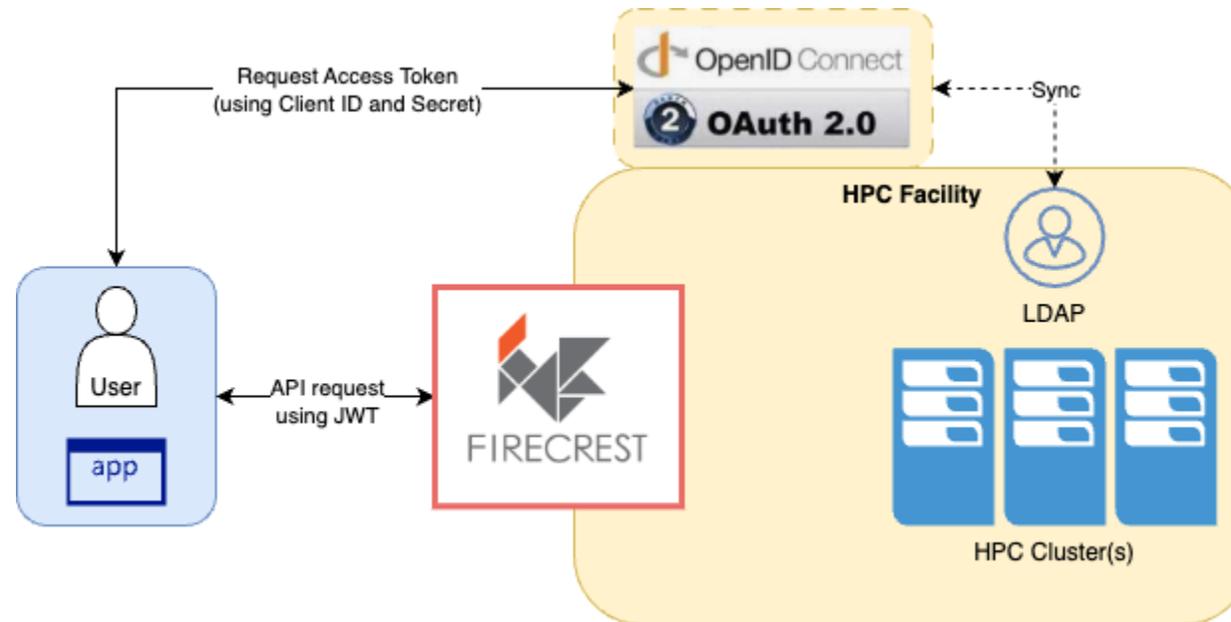
Thank you for your attention.

Q&A Tech Deep Dive

FirecREST Deep Dive

- Authentication

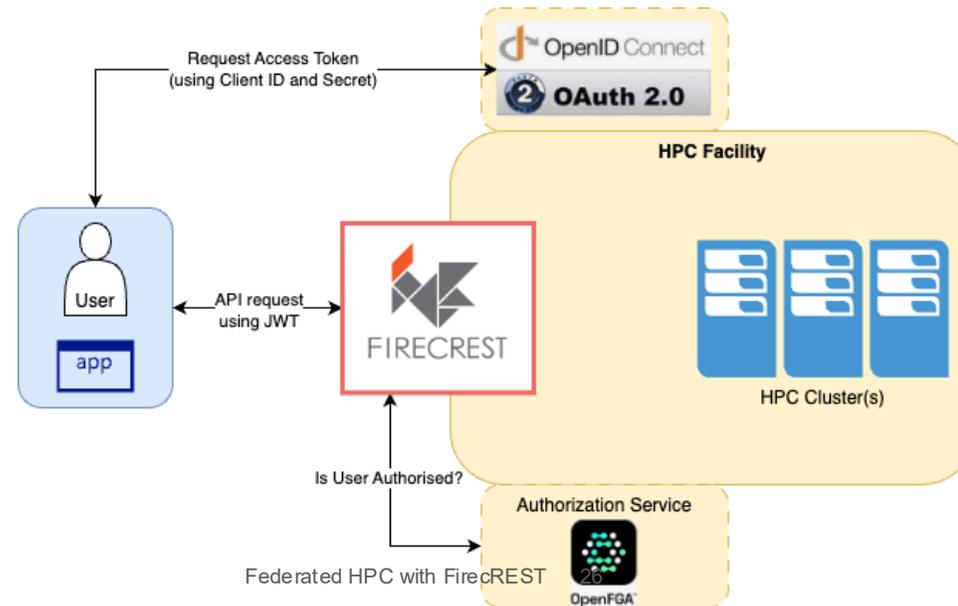
- AuthN relies on an OpenID Connect server (OIDC) - OAuth2 protocol
- FirecREST trusts in access token from trusted sources
- JSON Web Tokens (JWT) standard is used as access tokens



FirecREST Deep Dive

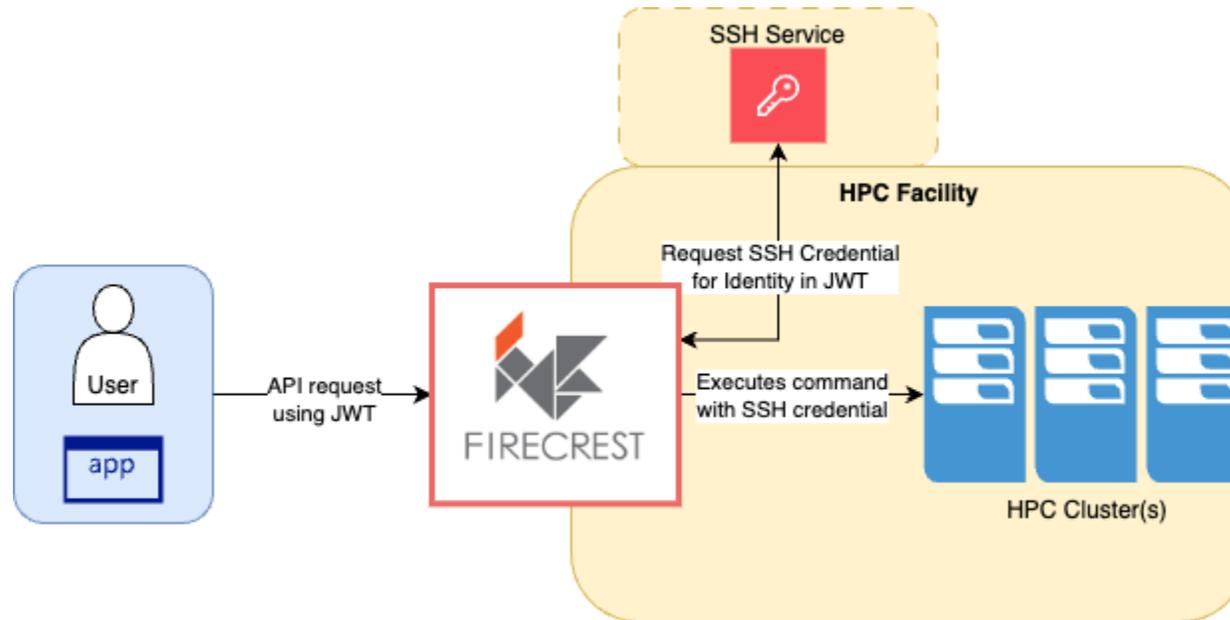
- Authorization

- FirecREST-v2 provides interface for Authorization
- Currently provides plugins for OpenFGA, an authorization service based in ReBAC (Relationship Based Access Control)
- JWT scopes can be used to limit access
- The idea is to limit the use of endpoints depending on the system or resources the user has access to



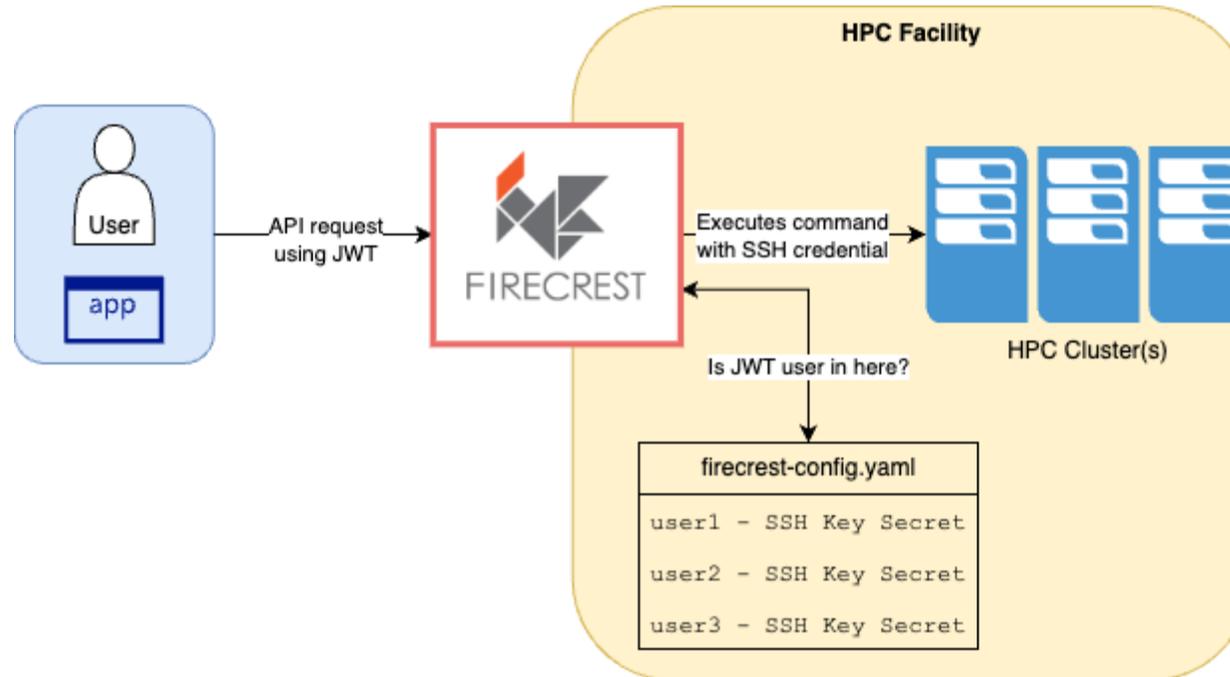
FirecREST Deep Dive

- Command execution
 - FirecREST translate JWT into **user credentials** for HPC systems
 - The SSH Service Adapter provides an abstraction to use the bundled CSCS SSH Service or any type of JWT-to-SSH service



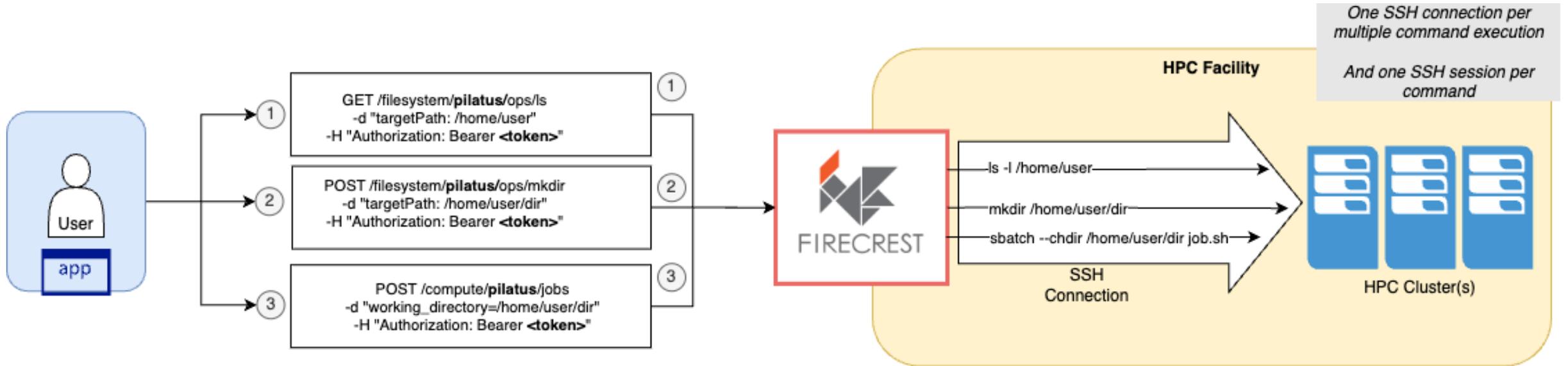
FirecREST Deep Dive

- Command execution
 - FirecREST translate JWT into **user credentials** for HPC systems
 - Using a user-SSH key list (not ideal, but a workaround)



FirecREST Deep Dive

- SSH Connection Pool
 - Needs to adjust the MaxSession setting in SSH Config



FirecREST Deep Dive

- External Data Transfers
 - FirecREST uses S3 Service to decouple data transfer channel from API
 - Data is staged for download using the Workload Scheduler

