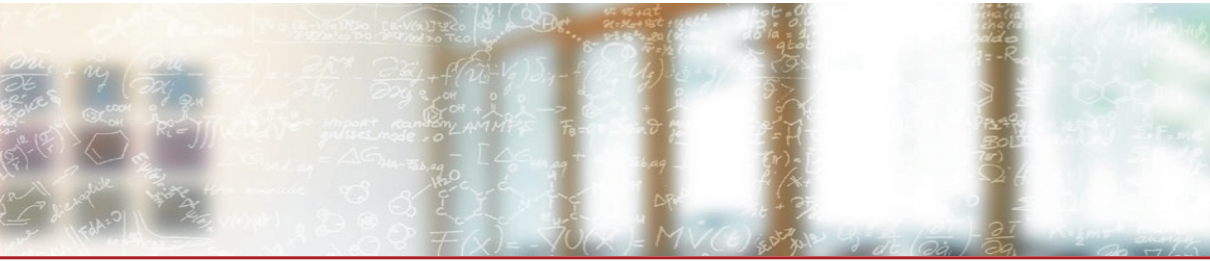
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



FirecREST: enabling programatic access of HPC resources

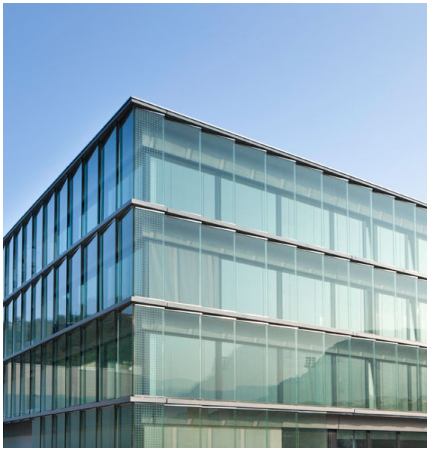
EuroCC Webinar

Eirini Koutsaniti, CSCS/ETH Zurich
Juan Dorsch, CSCS/ETH Zurich

12 November 2021

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU received support from the European Union's Horizon 2020 research and innovation.

FirecREST: a RESTful API to HPC systems



- Firecrest in a Nutshell
- Requirements of the system
- Microservice Architecture
- Important components of the system
- Deployment at CSCS

Firecrest in a Nutshell

Motivation

- Users wanted to develop applications and platforms that take advantage of the HPC resources.
- Need for a standard modern interface to the HPC resources:
  - HPC clusters
  - Job scheduler
  - Filesystem operations
  - Internal and external data transfers
- Need to integrate with the existing infrastructure

Firecrest in a Nutshell

FirecREST is a **RESTful Web API infrastructure**.

- Provides advanced HPC functionality for modern web-enabled portals and applications. It gives access to
  - HPC Workload Management
  - Data Mover
- Enforces integration with Identity Access Management (IAM) of the HPC center.

Requirements of the system

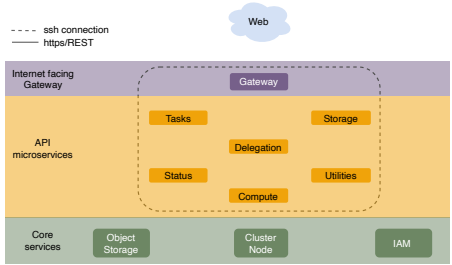
Requirements of the system

- The only Workload Manager that is supported right now is Slurm.
- FirecREST relies on a Slurm queue to handle large data transfers between internal file systems.
- FirecREST relies on an Object Storage service for large external data transfers. Compatible APIs:
  - OpenStack Swift
  - Amazon Web Services S3
- An Identity and Access Management that supports OpenID Connect (OIDC) protocol

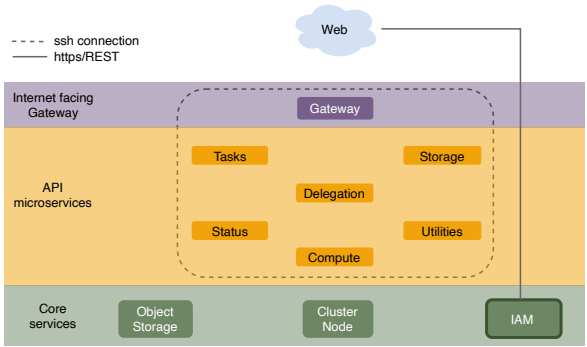
Microservice Architecture

Microservice Architecture

- FirecREST is a collection of loosely coupled services.
- This architecture provides maintainability, security and stability.



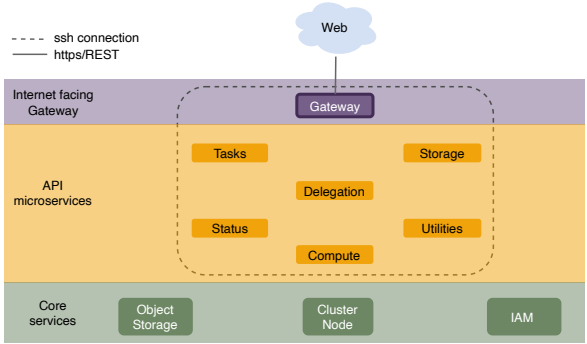
Microservice Architecture



IAM Layer

- Each FirecREST request has to include an OIDC token in the header.
- The first thing a client would have to do is to aquire a valid token from the OIDC server.

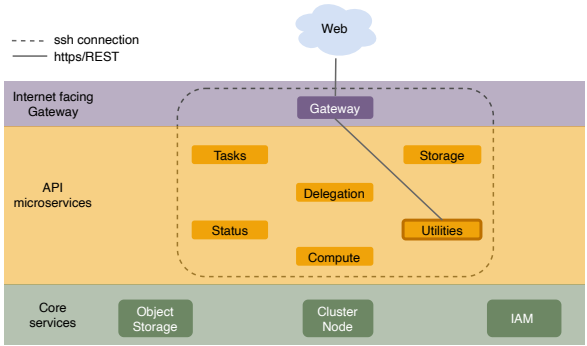
Microservice Architecture



Gateway

- It should be the only service that is open to the internet.
- It is the responsible microservice that will implement and enforce:
  - authentication
  - authorization
  - traffic control
  - analytics and logging of requests

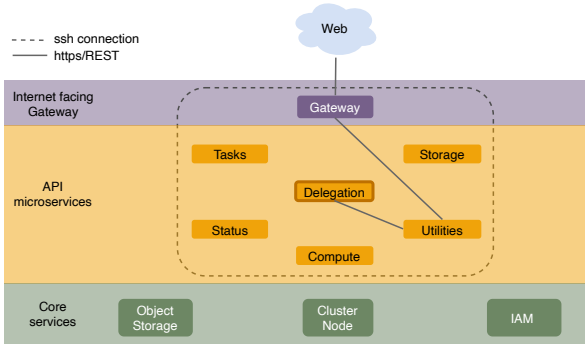
Microservice Architecture



Utilities microservice

- Provides filesystem utilities.
- Checks the validity of the parameters passed with the request.
- All calls are blocking operations.

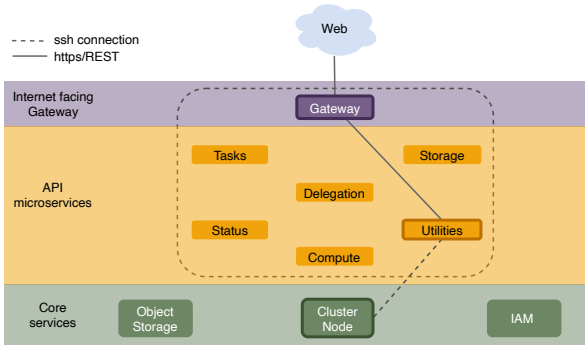
Microservice Architecture



Delegation microservice

- Creates a short-lived SSH certificate to be used for user authentication.
- These certificates are created for the given combination of username, shell command and arguments.

Microservice Architecture



- The Utilities microservice uses the SSH certificate to log in to a **Cluster node**.
- Parses the output of the command.
- Returns a json object to the client.

Microservice Architecture

Other microservices of FirecREST:

- **Compute:** Non-blocking calls to workload manager for submitting/querying/canceling jobs.
- **Storage:** Non-blocking calls to high-performance storage services.
- **Tasks:** Keeps track of the tasks that are created during asynchronous calls.
- **Status:** Provides information on services and infrastructure.

Advanced FirecREST Workflows

Compute Microservice

Every time FirecREST interacts with the scheduler, it is creating a task resource.

- To submit/query/cancel a job the client makes the appropriate request to the Compute microservice.
- It gets a response immediately with the newly created task.
- The task can be used to track the status of the request in an asynchronous way.

Advanced FirecREST Workflows

Storage Microservice - External transfers

- A staging area is used: Object Storage.
- The client will upload/download the file to/from this area.
- The requests from the client to FirecREST aim to get the url to this staging area.
- This allows FirecREST to be responsive and lightweight, since it delegates the large transfers to a service that is more suitable for this.

Advanced FirecREST Workflows

Storage Microservice - Internal transfers

- For small files' transfers you can simply use the Utilities Microservice.
- The maximum file size for data transfers through Utilities is configurable and you can get it from the Status Microservice.
- FirecREST has configurable time limit for all commands, so for larger files you will have to use the dedicated WLM queue for internal data transfers.
- FirecREST will create the job script and submit it based on the request's arguments to Storage Microservice.

Important components of the system

Kong API Gateway

- Open-source microservice API Gateway
- Optimized for distributed architecture.
- The Kong server is built on NGINX.

Kong

Configuration

- Each microservice is redirected to a different path.
- Kong's configuration needs to hold the public key used to verify the JWT tokens.
- Plugins:
  - jwt
  - request-termination
  - rate-limiting
  - zipkin (for integration with Jaeger)

Delegation microservice (Certificator)

- FirecREST has to use the JWT access token and create a short-lived SSH certificate before executing the command of the user.
- This microservice acts as a certificate authority that can issue SSH certificates for the users of the system.
- The certificate is valid for a short time, for the username that is on the token and only for this command.

Delegation microservice (Certificator)

Configuration of the Delegation microservice

- System administrators of the system have to create a pair of Certificate Authority (CA) keys.
- Delegation microservice will need the private key in order to create SSH certificates for the machine.
- The cluster node will need the public key to validate the certificates.
- The certificates that are created are created for a ”dummy” key, that is used by the microservices and in this way only the microservices can use these keys.

Delegation microservice (Certificator)

How is the command executed on the machine node

In the system nodes we don't immediately run the command but set **ForceCommand** in *sshd_config* and run the command through a filter and logging wrapper script.
This script:

- Validates the certificate type.
- Validates that the certificate signature.
- Checks that this command is one the allowed ones.
- Finally executes the command if everything was okay.
- Logs every step.

IAM Layer

Integration with the IAM layer:

- FirecREST verifies the signature of the token using Keycloak's public key.
- FirecREST can check for a custom scope in the token, to make sure this token belongs to a FirecREST client.

IAM Layer

Supported OIDC workflows:

- Authorization Code Flow
- Authorization Code Flow with Proof Key for Code Exchange (PKCE)
- Client-credentials Flow

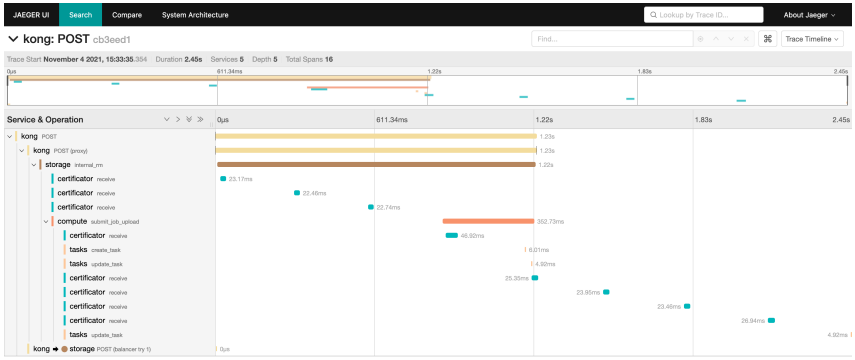
Open Policy Agent (OPA)

- It is an agent that can be used to authorize fine-grained access to specific resources in a RESTful API environment.
- The idea behind the integration of OPA with FirecREST is to adapt access to microservices, endpoints, and systems in a more secure and flexible way.

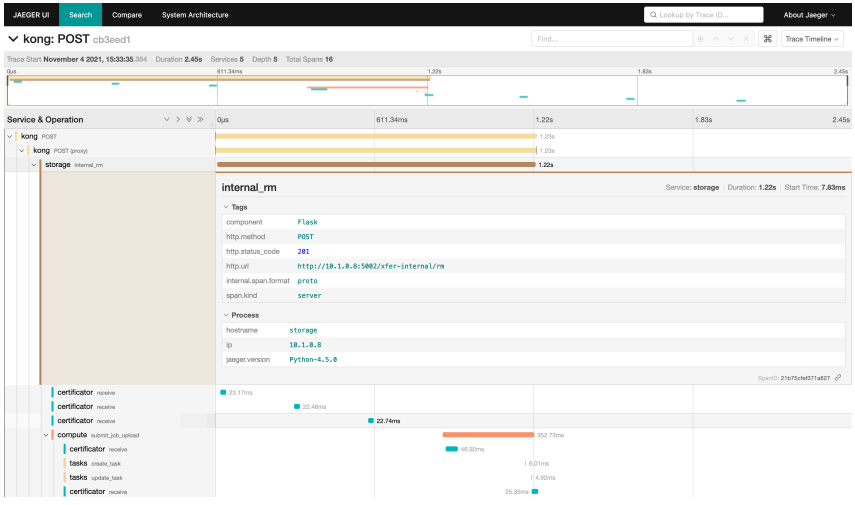
Jaeger

- One request can trigger actions in multiple microservices.
- Jaeger is an open-source tracing system for distributed microservices.
- The Gateway needs to add an identifier to each new request.
- Microservices have to propagate this identifier in every request.
- We log this identifier with every command in the cluster.

Jaeger


JAEGER UI
Search
Compare
System Architecture
Lookup by Trace ID...
About Jaeger
kong: POST cb3eed1
Find...
Trace Timeline
Trace Start November 4 2021, 15:33:35.334 Duration 2.45s Services 5 Depth 5 Total Spans 16
0µs
611.34ms
1.22s
1.83s
2.45s
Service & Operation
kong POST
kong POST (proxy)
storage internal_rm
certificator receive
certificator receive
certificator receive
compute submit_job_upload
certificator receive
tasks create_task
tasks update_task
certificator receive
certificator receive
certificator receive
certificator receive
tasks update_task
kong → storage POST (balancer try 1)
23.17ms
22.46ms
23.74ms
352.73ms
46.92ms
6.01ms
4.90ms
25.35ms
23.95ms
23.46ms
26.94ms
4.92ms

Jaeger


JAEGER UI
Search
Compare
System Architecture
Lookup by Trace ID...
About Jaeger
kong: POST cb3eed1
Find...
Trace Timeline
Trace Start November 4 2021, 15:33:35.394 Duration 2.45s Services 5 Depth 5 Total Spans 16
0µs
611.34ms
1.22s
1.83s
2.45s
Service & Operation
kong POST
kong POST [proxy]
storage internal_rm
internal_rm
Service: storage | Duration: 1.22s | Start Time: 7.83ms
Tags
component Flask
http.method POST
http.status_code 201
http.url http://10.1.0.8:5002/xfer-internal/rm
internal.span.format proto
span.kind server
Process
hostname storage
ip 10.1.0.8
jaeger.version Python-4.5.0
SpanID: 21b75cfef371a527
certificator receive
23.17ms
certificator receive
22.46ms
certificator receive
22.74ms
compute submit_job_upload
352.73ms
certificator receive
46.90ms
tasks create_task
6.01ms
tasks update_task
4.92ms
certificator receive
25.35ms

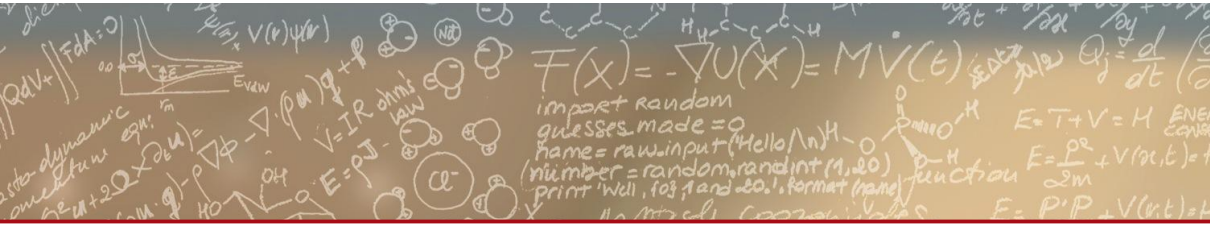
Deployment at CSCS

Deployment at CSCS

- Only Kong is open to the users.
- /users filesystem is not mounted.
- Kong can have rate limiters.
- HTTPS between microservices.
- Network rules between micro services.
- OPA is used for to deny access to normal users in the TDS environment.

Where to find more information

- The complete API: https://firecrest-api.cscs.ch/
- Source on Github: https://github.com/eth-cscs/firecrest/
  It includes a template client in Python and a demo environment in Docker.
- Documentation page and examples: https://firecrest.readthedocs.io
- Python library for the API: https://github.com/ekouts/pyfirecrest
- EuroCC page: https://products.cscs.ch/firecrest/



Thank you for your attention!

EURO CASTIEL

EuroHPC
Joint Undertaking