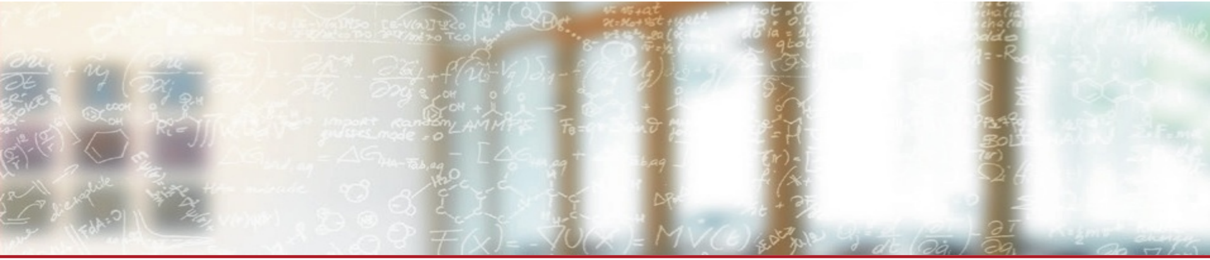
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



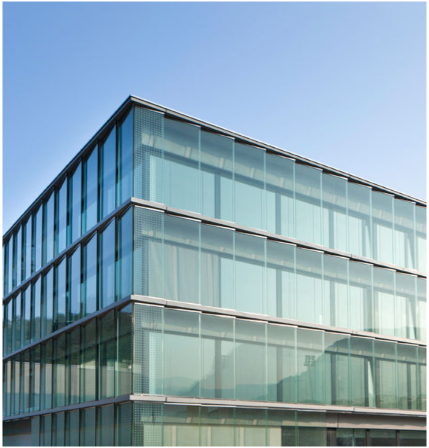
FirecREST: a RESTful API to HPC systems

Access Abstraction to HPC Resources

Eirini Koutsaniti, Computational Scientist, ETH Zurich/CSCS

22 October 2020

FirecREST: a RESTful API to HPC systems



- Introduction to FirecREST
  - What is FirecREST
  - Who is it for
  - How it can be used
- Microservice Architecture
- Advanced FirecREST Workflows
  - Compute Microservice
  - Storage Microservice
- Conclusions

Introduction to FirecREST

What is FirecREST

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 the authorization and authentication infrastructure (AAI) of the HPC center.

What is FirecREST

REST (REpresentational State Transfer) API

- A software design pattern that specifies a uniform and predefined collection of stateless operations.

What is FirecREST

REST (REpresentational State Transfer) API

- A software design pattern that specifies a uniform and predefined collection of stateless operations.
- Builds on existing HTTP methods, such as:
  - **GET** to retrieve resources
  - **PUT/POST** to create/update resources
  - **DELETE** to delete resources

What is FirecREST

REST (REpresentational State Transfer) API

- A software design pattern that specifies a uniform and predefined collection of stateless operations.
- Builds on existing HTTP methods, such as:
  - **GET** to retrieve resources
  - **PUT/POST** to create/update resources
  - **DELETE** to delete resources
- Every request will get a response with a standard status code.
- Provides structured responses, like JSON, that can be easily parsed by any programming language.

Who is it for

FirecREST is meant to be used by platform developers.

- Many scientists would benefit from scientific platforms, instead of working with terminals and SSH connections.
- A RESTful API enables these types of platforms by providing a standard interface to the HPC center.
- It is not meant to be a replacement for SSH.

Accessing HPC resources

Traditionally

- Connect through ssh to the node.
- Provide the password of the user or an SSH key.
- Perform the action in the terminal.
- Parse the output and handle possible errors.

With FirecREST

- Obtain a temporary token that is connected to the account of the user.
- Make the appropriate HTTP request to FirecREST's gateway.
- The reply of the request is in JSON format.

Concrete examples of the API

How to list the contents of a directory:

```
$ curl -X GET "<firecrest_ip>/utilities/ls?targetPath=<targetpath>" \
       -H "Authorization: Bearer <token>" \
       -H "X-Machine-Name: <machine_name>"
```

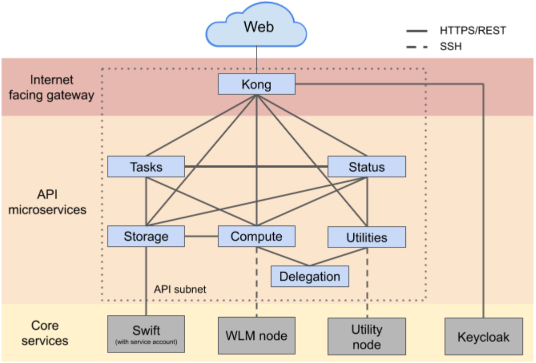
How to submit a job:

```
$ curl -X POST "<firecrest_ip>/compute/jobs" \
       -H "Authorization: Bearer <token>" \
       -H "X-Machine-Name: <machine_name>" \
       -F "file=@/path/to/script.sh"
```

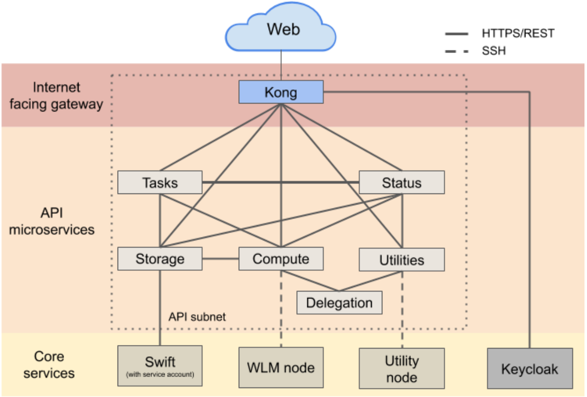
Microservice Architecture

Microservice Architecture

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



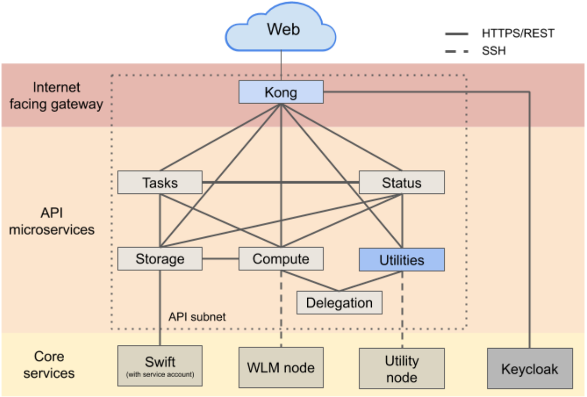
Microservice Architecture



Kong API Gateway

- Open-Source microservice API Gateway
- Implements and enforces:
  - authentication
  - authorization
  - traffic control
  - analytics
  - logging

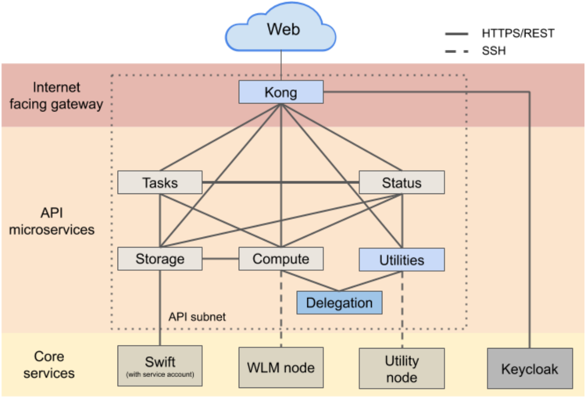
Microservice Architecture



Utilities microservice

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

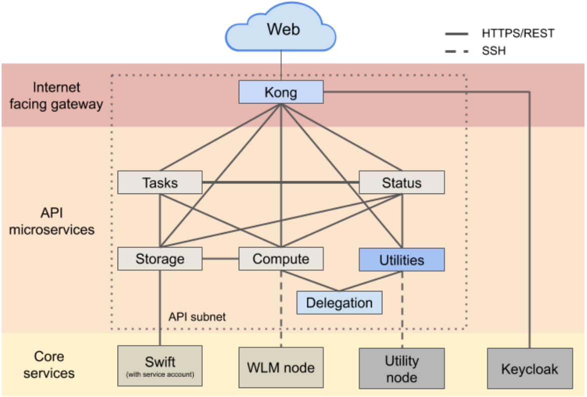
Microservice Architecture



Delegation microservice

- Takes a valid JWT access token as input.
- Creates a short-lived SSH certificate to be used for user authentication.

Microservice Architecture



- The Utilities microservice uses the SSH certificate to log in to a **Utility 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 jobs.
- **Storage:** Non-blocking calls to high-performance storage services.

Both **Compute** and **Storage** microservices respond with a reference to a temporary **task** resource tracking the request state.

Microservice Architecture

Other microservices of FirecREST:

- **Tasks:** Keeps track of the tasks that are created during asynchronous calls.
- **Status:** Provides information on services and infrastructure

Advanced FirecREST Workflows

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 get 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

- For external transfers a staging area is used.
- 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.

Conclusions

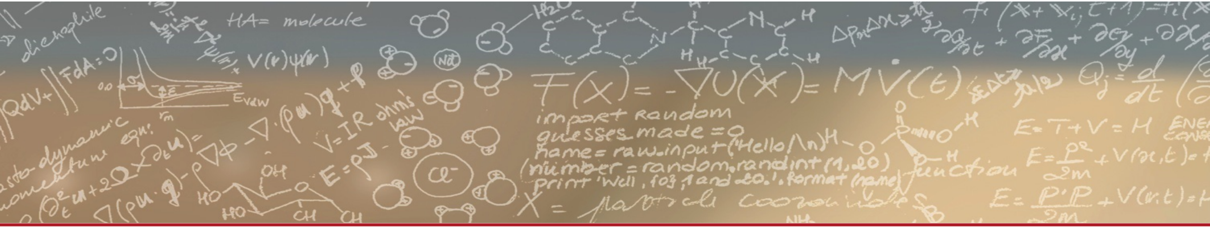
Conclusions

Why use FirecREST?

- For automated workflows a rest API is more convenient than other custom solutions.
- FirecREST enables managing of the workload manager.
- It enables data transfers.
- It is a common, stable, maintainable API.
- It enforces that all API requests are authenticated.

Where to find more information

- The complete API: http://firecrest-api-tds.cscs.ch:8000/
- Source on Github: https://github.com/eth-cscs/firecrest/
  It includes a template client in Python.
- Documentation page and examples: https://firecrest.readthedocs.io



Thank you for your attention