Account and Authorization Management System

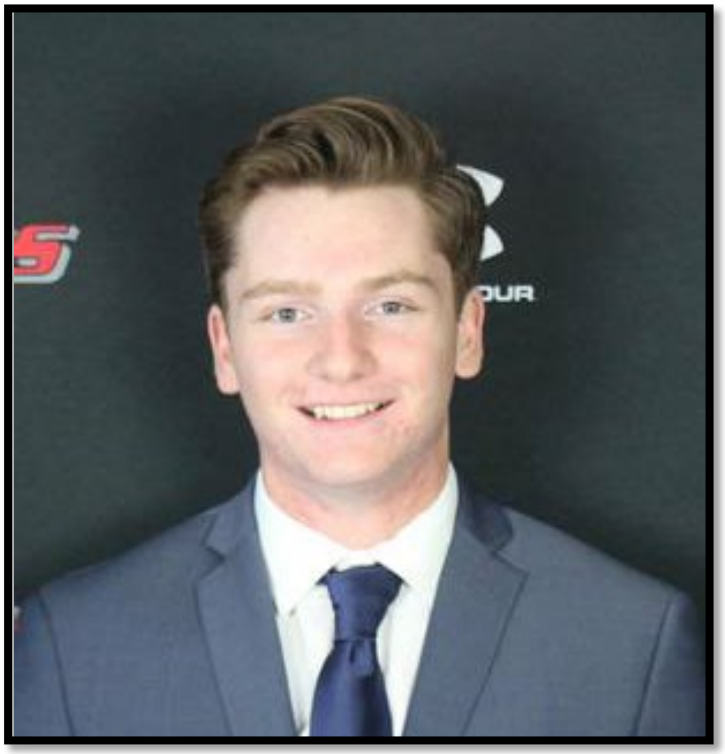
Megan Acosta

Shane Cancilla

August 10th, 2023

Who We Are

Megan Acosta
University of the Pacific
Graduating May 2024
Computer Science
Conc. Networking & Security



Shane Cancilla
Cal State East Bay
Graduating Dec 2023
Computer Science

Goals of Our Project

Use Case: Streamlined user account management on HPC Academy clusters

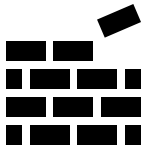
- Centralized authentication
  - Login to several systems/services with single set of credentials
  - Robust authentication methods
- User/Group Management
  - Tools for admin to manage users, groups, and roles
  - Web UI that makes system simpler to visualize
- Host-Based Access Control
  - Control which users/groups can access specific servers in the network
- Certificate Authority
  - Provides secure communication for web services

Initial Investigation

- Research Objectives:
  - Determine Identity Management Systems implementation
  - Determine 2 Factor Authentication implementation
- Considerations:
  - Open source
  - Extent of documentation
  - Ease of user / policy management
  - OS compatibility
  - Perceived complexity / time-friendly

IDMs	2FA
FreeIPA	**Google Auth**
OAuth	OAuth
	DUO Security

Major Components of Our Project



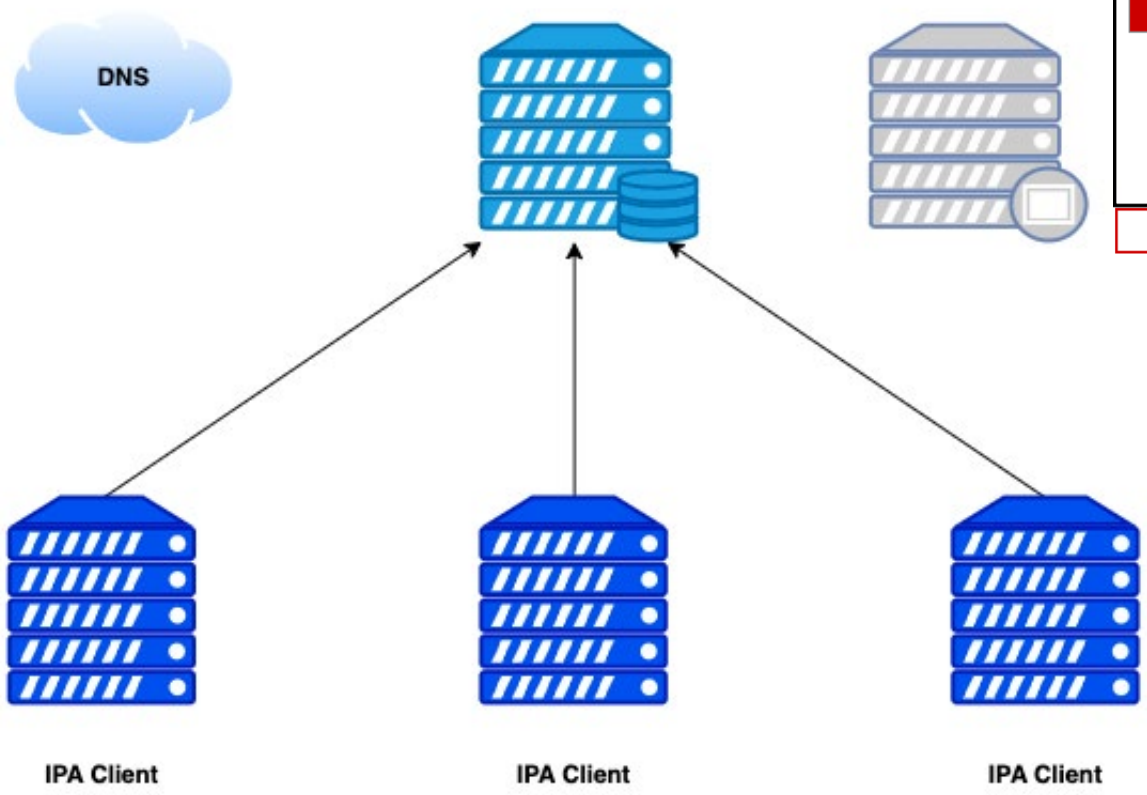
- FreeIPA

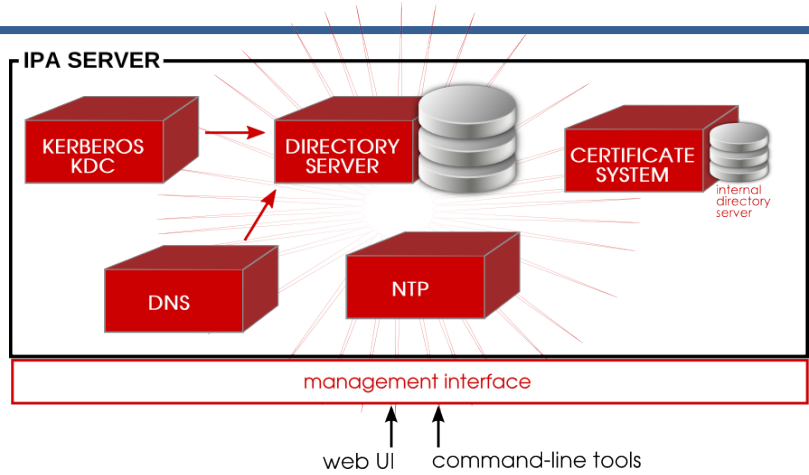
  - Kerberos
    - Authentication
  - LDAP
    - Database for network environment
  - Apache
    - Web server for UI
  - Dogtag PKI
    - Certificate Authority

- Google Auth

  - 2FA provider
- DNS Server
  - Integrated
  - DNSmasq
- RADIUS Server
  - Risk-based authentication

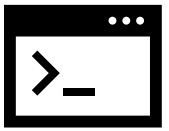
Cluster Layout





Source: https://abbra.fedorapeople.org/.todo/freeipa-docs/

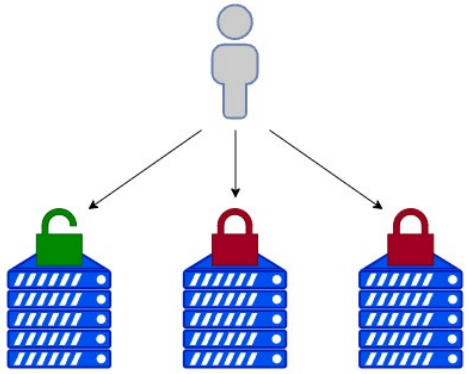
Central Auth & User/Group Mgmt



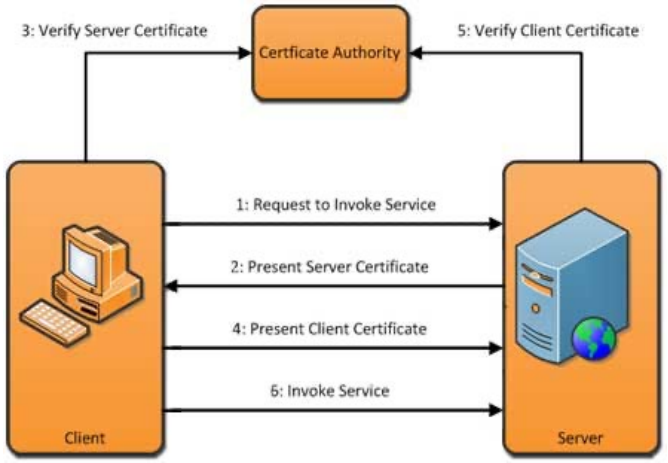
- Defined a FreeIPA domain
  - Installed one FreeIPA server and several FreeIPA clients
- Created several users within FreeIPA
  - Implemented 2FA with Google Authenticator to specific users
  - Issued password resets to those users
- Defined user groups and host groups
  - Assigned various Host Based Authentication Rules

HBAC & CA Goals

- Defined HBAC rules
  - Limited user access to certain client nodes
  - Utilized defined user and host groups



- Certificate Authority
  - Created nginx web server that requested certificate
  - Demonstrates FreeIPA's ability to provide intranet certificates

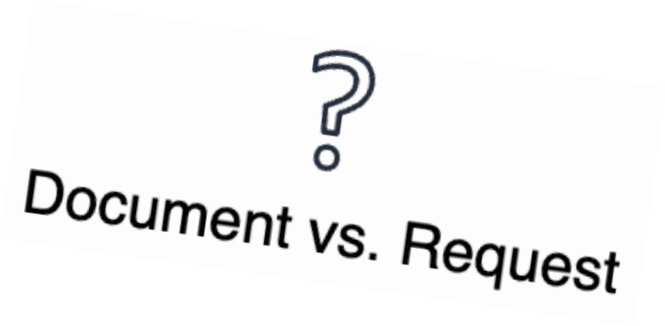


Source: https://help.mulesoft.com/s/article/Configure-HTTPS-Listener-secured-by-TLS-1-2

Issues We Faced

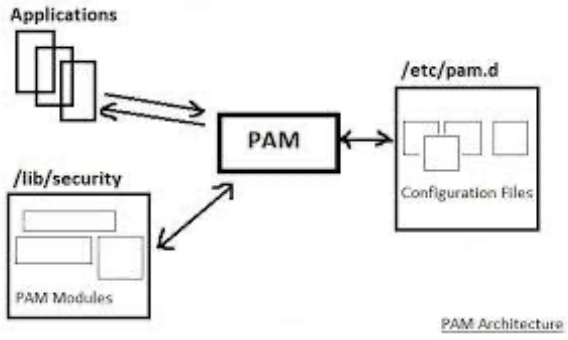
And how we solved them

- Installation of FreeIPA
  - Misunderstanding documentation
  - Sensitivity to hostname (FQDN)
- DNS Implementation
  - Configured without integrating
  - Dnsmasq, or reconfigured
- RADIUS authentication methods
  - Intention for risk-based authentication
  - Issues with LDAP configuration



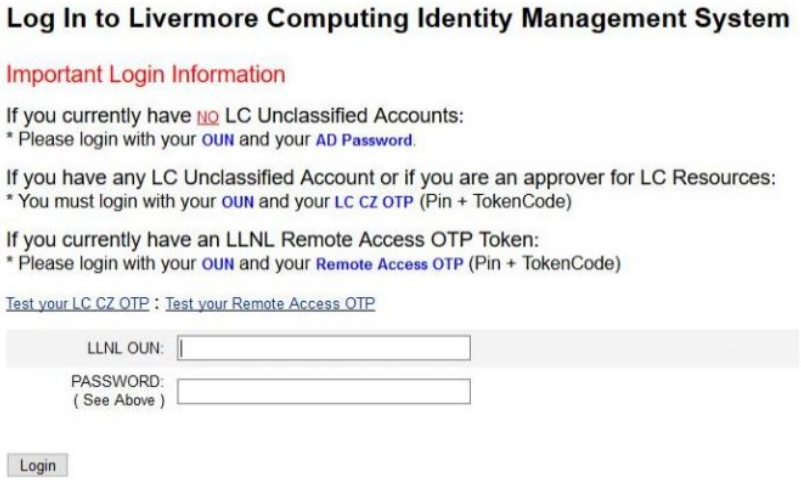
What We Would Do Differently
with more time

- Find more work arounds for risk-based authentication
  - RADIUS was a bit of a struggle
  - Interact directly with the source code
  - pam.d configuration



Source: https://borosan.gitbook.io/lpic2-exam-guide/2102-pam-authentication

- Learn more about LC's IDM development
  - Try and replicate some of that work
  - Implement components related to that system



References

- FreeIPA - https://freeipa.org/page/Quick_Start_Guide#getting-started-with-ipa
- FreeIPA Manual - https://abbra.fedorapeople.org/.todo/freeipa-docs/
- FreeIPA Tutorial - https://linux.how2shout.com/how-to-install-freeipa-on-almalinux-or-rocky-8/
- RedHat IPA - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/configuring_identity_management/installing_the_ipa_client_on_linux
- Dogtag PKI + FreeIPA - https://www.admin-magazine.com/Archive/2022/70/Certificate-management-with-FreeIPA-and-Dogtag
- Risk-Based Authentication - https://riskbasedauthentication.org/

Questions?

Thank you to our mentors!

Dave Fox

Martin Baltezore

Rigo Moreno

Naomi Cheeves

And all of Livermore Computing!

CerebrasGPT Web Application

Utilizing RabbitMQ and Kafka

Karis Kim & Taylor Rohovit
HPC Academy Interns

August 10, 2023

Meet the Team!

Karis Kim

- UC Berkeley
- Electrical Engineering & Computer Sciences
- Expected Grad: May 2025

Taylor Rohovit

- Johns Hopkins University
- Computer Science
- Expected Grad: May 2025

High Level Overview of Our Project

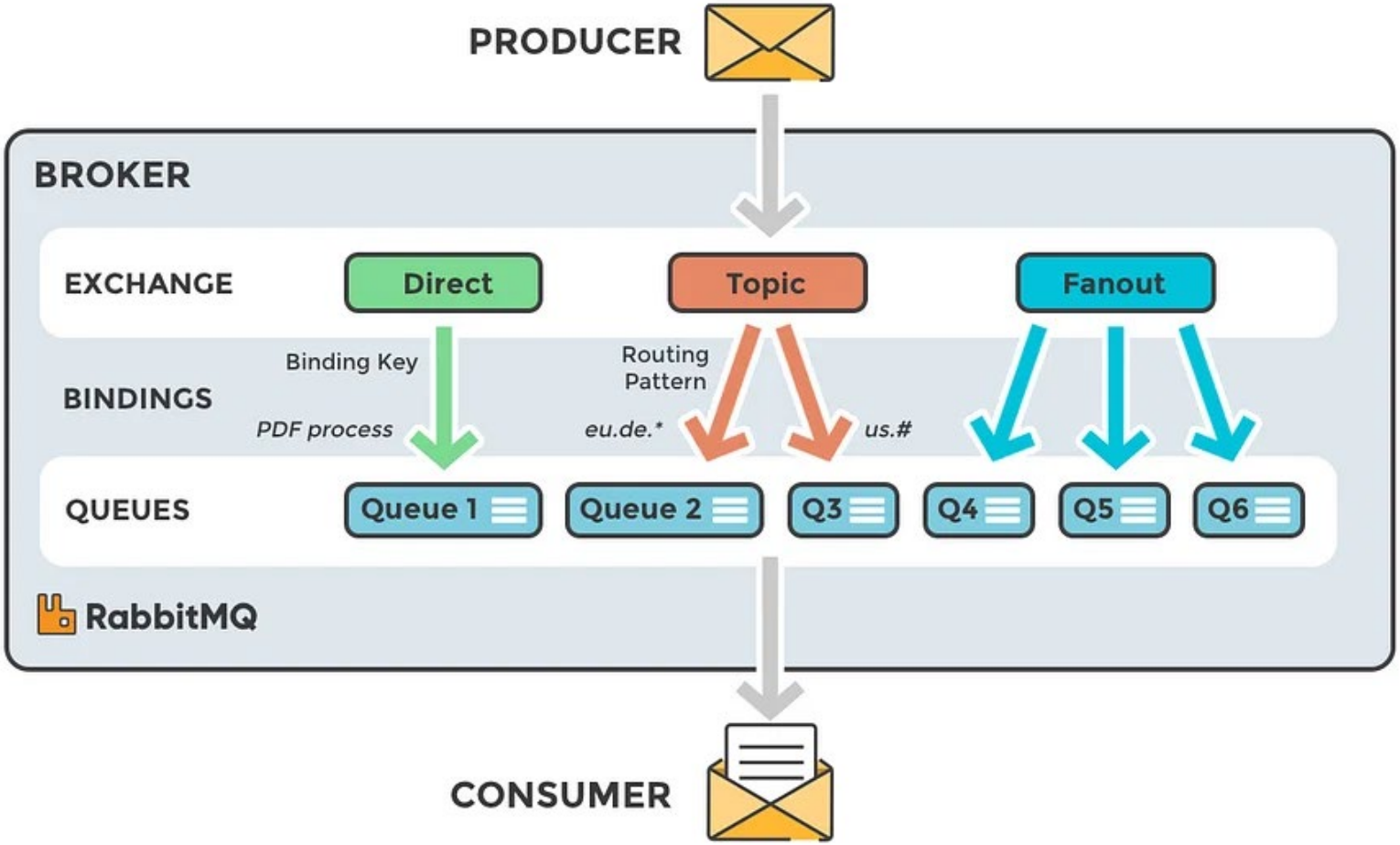
Goals

- Enable messaging between multiple nodes on HPC clusters
- Create a Flask web application using RabbitMQ that allows users to interact with CerebrasGPT
- Perform analytics using Kafka
  - User satisfaction
  - Response time
  - Sentiment

Major Technologies Used

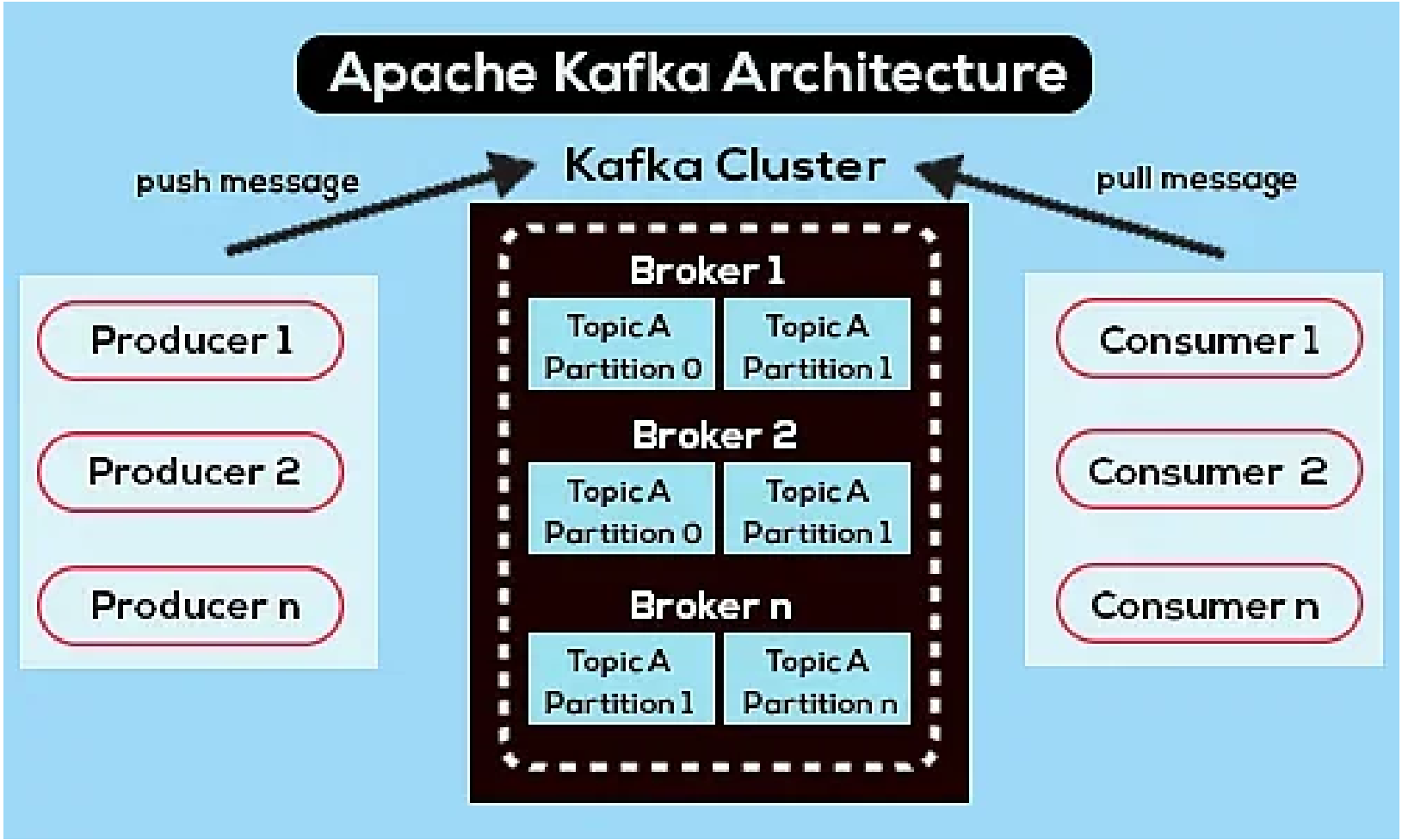
- **Cerebras-GPT:** a family of 7 open compute-optimal language learning models
  - We used the 13B parameter model
- **RabbitMQ:** message broker that uses an exchange to route messages to queues to be picked up by consumers
  - Push Model → Smart broker, dumb consumer
- **Kafka:** message broker that stores messages in topics; optimized for real time event streaming
  - Pull Model → Dumb broker, smart consumer

Major Technologies Used: RabbitMQ



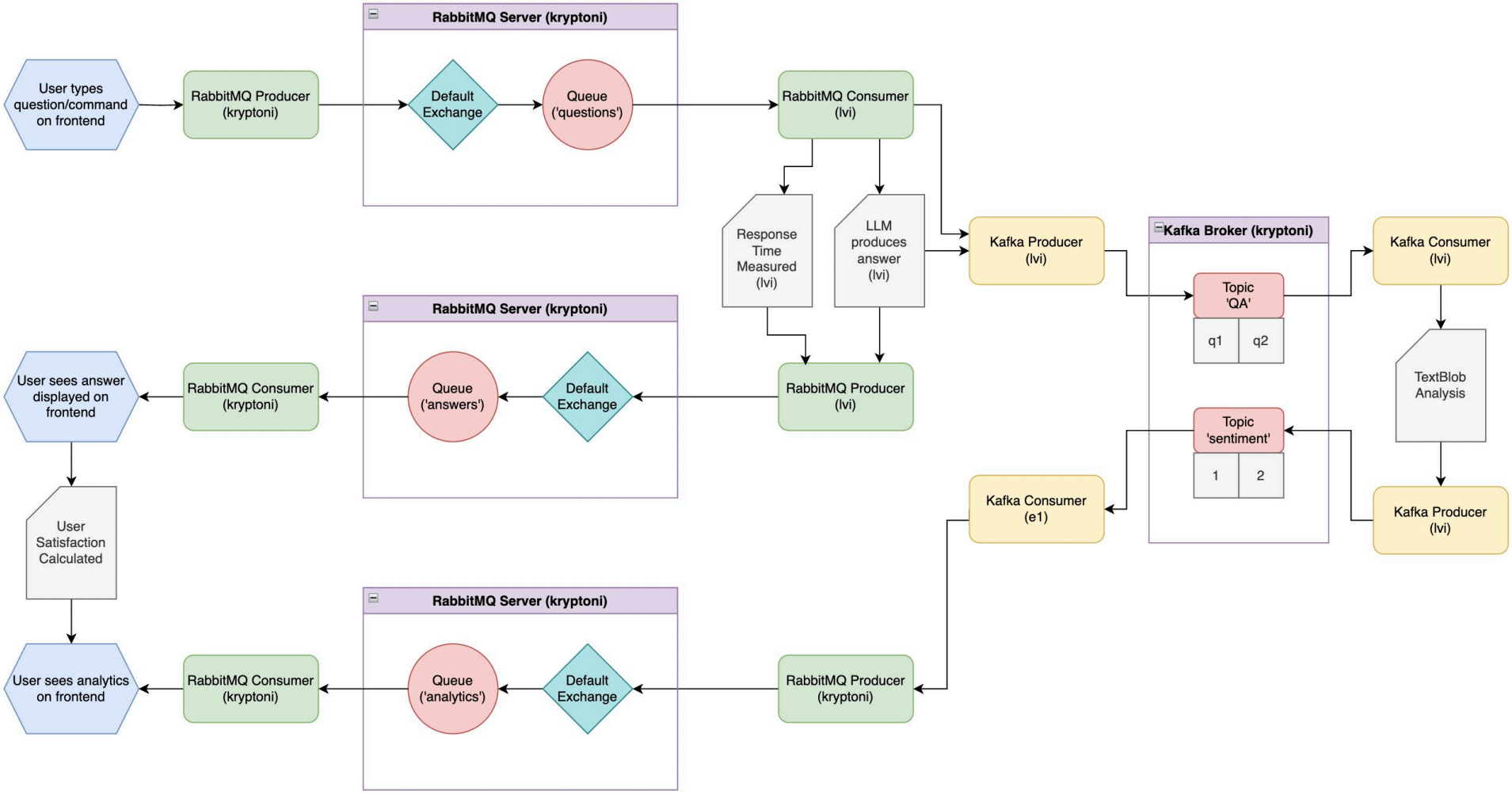
source: https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Major Technologies Used: Kafka



source: https://www.projectpro.io/article/apache-kafka-architecture-/442

Architecture



Demo!

Challenges

- Learning new tools - Flask, RabbitMQ, Kafka, HTML, CSS, Chart.js, VNC
- Connecting RabbitMQ and Kafka in the same pipeline
  - infinite loop issues with Kafka
- Installing dependencies for CerebrasGPT to use model later on
  - Python version, pip3 install, xturing
- RabbitMQ syncing issues

Future Ideas

- Use analytics to train model to produce better responses
- Real time constant updates on the analytics page
- Experiment with different models
- Create user database to store specific history for each user

Works Cited

RabbitMQ - https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Install RabbitMQ - https://www.rabbitmq.com/install-rpm.html

Flask - https://flask.palletsprojects.com/en/2.3.x/

Cerebras-GPT - https://www.listendata.com/2023/03/open-source-chatgpt-models-step-by-step.html

Kafka - https://linuxconfig.org/how-to-install-kafka-on-redhat-8

Kafka vs. RabbitMQ - https://www.projectpro.io/article/kafka-vs-rabbitmq/451

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Flux RestAPI

HPC Academy

Khoi Nguyen, Xander Armatis

August 10, 2023

Flux RestAPI Project Members

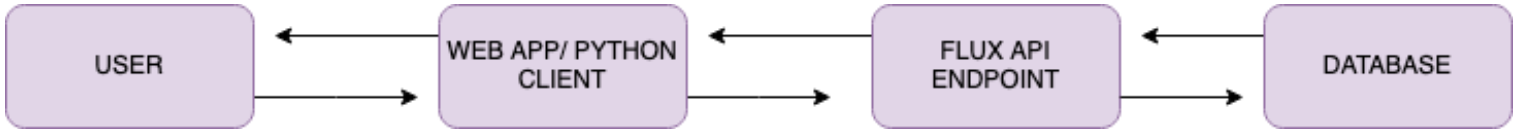
Khoi Nguyen
UC Berkeley
4th year EECS

Xander Armatis
UC Santa Cruz
3rd Year CS

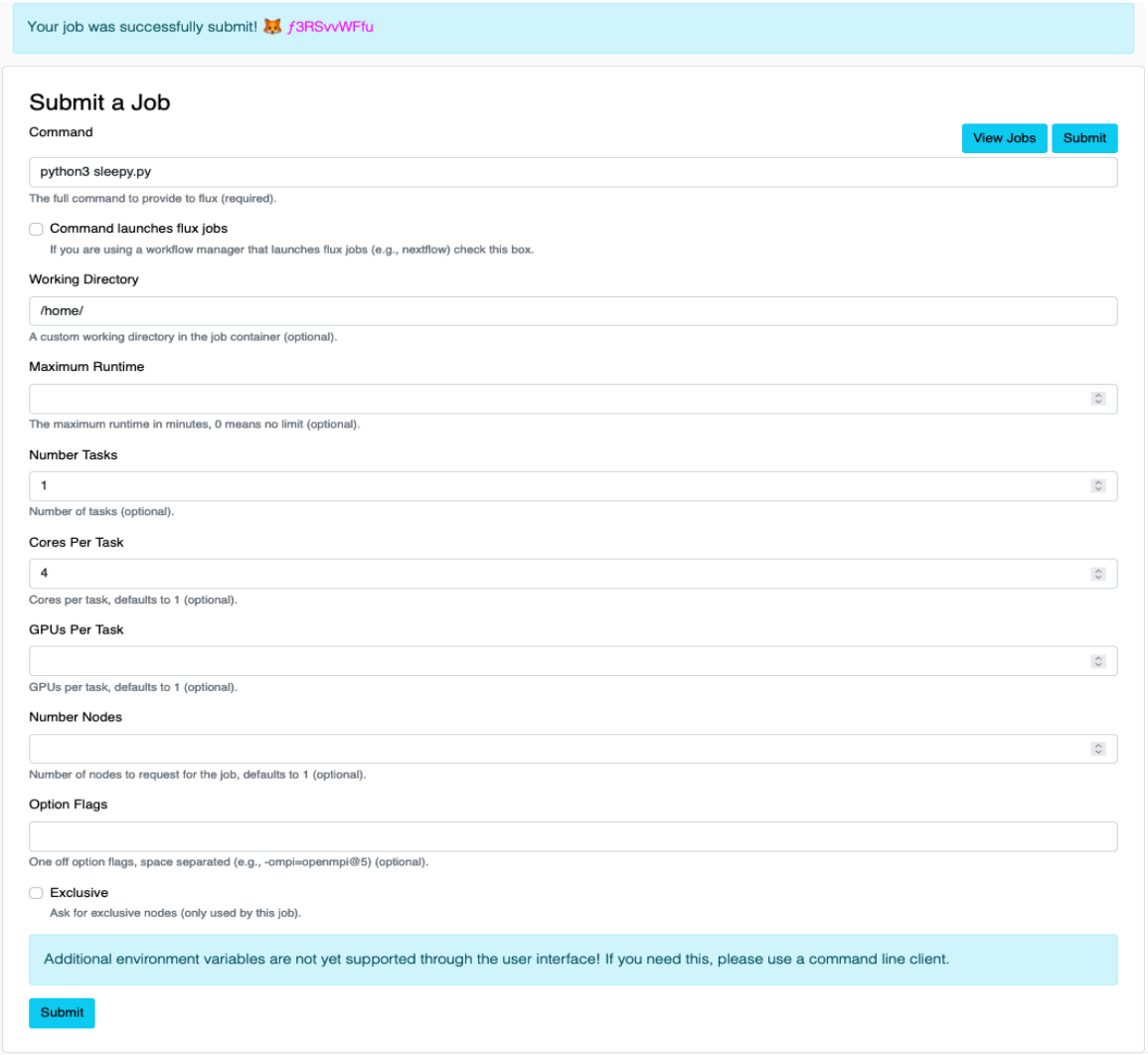
Overview of Flux RestAPI

With Python and web clients

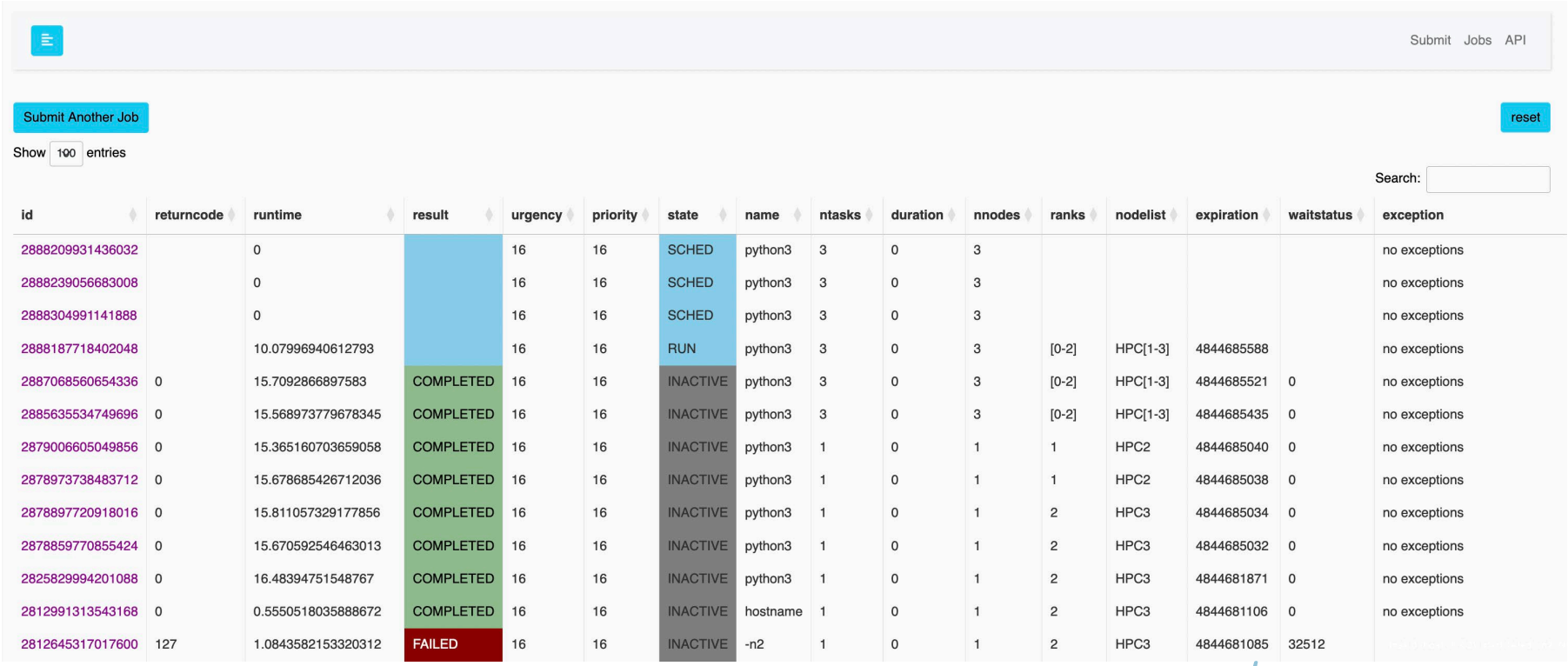
- Can be run with little to no command line.
- Provides web frontend for Flux.
- Provides an interface for Flux in Python.
- Improve accessibility for users to interact with Flux.
- Multiple ways to run the API endpoint:
  - Proxy and Port Forwarding
  - VNC Viewer
  - Visual Studio Code (VSCode)



Flux RestAPI Webserver Visuals – Job Submission



Flux RestAPI Webserver Visuals – Job List



Submit Jobs API

Submit Another Job

reset

Show 100 entries

Search:

id	returncode	runtime	result	urgency	priority	state	name	ntasks	duration	nnodes	ranks	nodelist	expiration	waitstatus	exception
2888209931436032		0		16	16	SCHED	python3	3	0	3					no exceptions
2888239056683008		0		16	16	SCHED	python3	3	0	3					no exceptions
2888304991141888		0		16	16	SCHED	python3	3	0	3					no exceptions
2888187718402048		10.07996940612793		16	16	RUN	python3	3	0	3	[0-2]	HPC[1-3]	4844685588		no exceptions
2887068560654336	0	15.7092866897583	COMPLETED	16	16	INACTIVE	python3	3	0	3	[0-2]	HPC[1-3]	4844685521	0	no exceptions
2885635534749696	0	15.568973779678345	COMPLETED	16	16	INACTIVE	python3	3	0	3	[0-2]	HPC[1-3]	4844685435	0	no exceptions
2879006605049856	0	15.365160703659058	COMPLETED	16	16	INACTIVE	python3	1	0	1	1	HPC2	4844685040	0	no exceptions
2878973738483712	0	15.678685426712036	COMPLETED	16	16	INACTIVE	python3	1	0	1	1	HPC2	4844685038	0	no exceptions
2878897720918016	0	15.811057329177856	COMPLETED	16	16	INACTIVE	python3	1	0	1	2	HPC3	4844685034	0	no exceptions
2878859770855424	0	15.670592546463013	COMPLETED	16	16	INACTIVE	python3	1	0	1	2	HPC3	4844685032	0	no exceptions
2825829994201088	0	16.48394751548767	COMPLETED	16	16	INACTIVE	python3	1	0	1	2	HPC3	4844681871	0	no exceptions
2812991313543168	0	0.5550518035888672	COMPLETED	16	16	INACTIVE	hostname	1	0	1	2	HPC3	4844681106	0	no exceptions
2812645317017600	127	1.0843582153320312	FAILED	16	16	INACTIVE	-n2	1	0	1	2	HPC3	4844681085	32512	

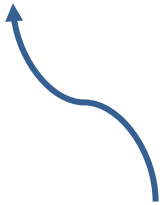
Flux RestAPI Webserver Visuals – Job Details

Job 2825829994201088

Attribute	Value
ID	2825829994201088
Name	python3
State	INACTIVE
Result	NO RESULT
Urgency	16
Priority	16
Number Tasks	1
Cores	1
Nodes	1 (HPC3)
Ranks	2
Runtime	16.48394751548767
Return Code	0

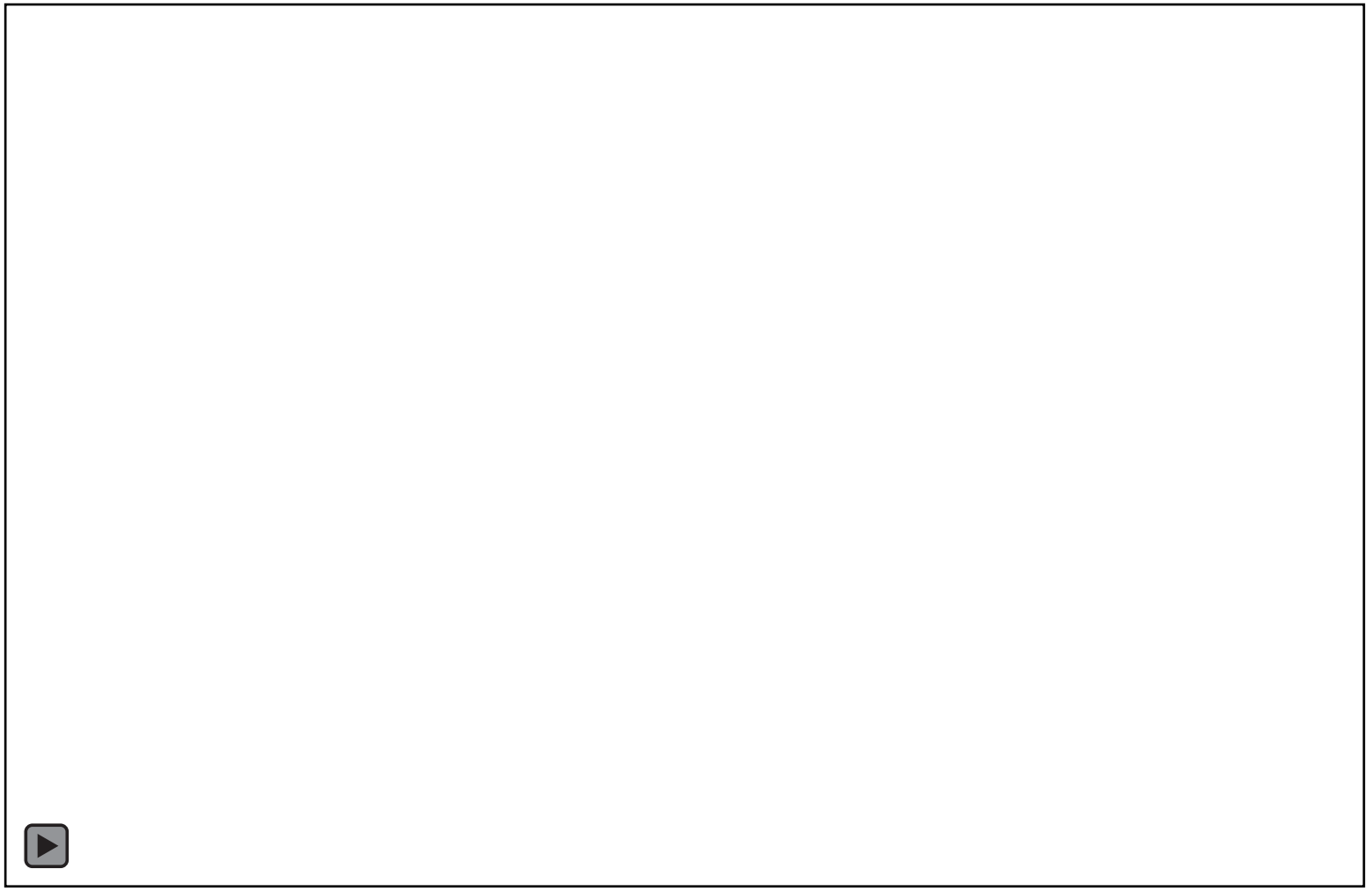
```
hello, the machine you are on is HPC3

This will now sleep for 15 seconds
```



Standard Output

Flux Web Client Demo



Flux RestAPI – Python methods

- get_client(): Set up Python client object to communicate with Flux
- client.submit(command, **kwargs): submit a job with a command. **kwargs is for extra parameters.
- client.cancel(jobid): cancel a job given jobid argument.
- client.list_nodes(): list available Flux nodes in cluster
- client.jobs(jobid=None, detail=False, listing=False): list all jobs if jobid is None, else list a job of jobid.
- client.stop(): stop Flux service
- etc…

Flux RestAPI Visuals – Python Client

```python
res = cli.submit(command='cat /etc/hostname', num_tasks=3, num_nodes=3)
id = None
if res:
    print(json.dumps(res, indent=4))
    id = res['id']

res = cli.jobs(id)
if res:
    print(json.dumps(res, indent=4))
```


```
{
    "Message": "Job submit.",
    "id": 404308977778688
}
{
    "id": 404308977778688,
    "userid": 1000,
    "urgency": 16,
    "priority": 16,
    "t_submit": 1691529761.2747245,
    "t_depend": 1691529761.2873735,
    "state": "SCHED",
    "name": "cat",
    "ntasks": 3,
    "ncores": 3,
    "duration": 0.0,
    "nnodes": 3,
    "result": "",
    "returncode": "",
    "runtime": 0.0,
    "waitstatus": "",
    "nodelist": "",
    "exception": {
        "occurred": "",
        "severity": "",
        "type": "",
        "note": ""
    }
}
```

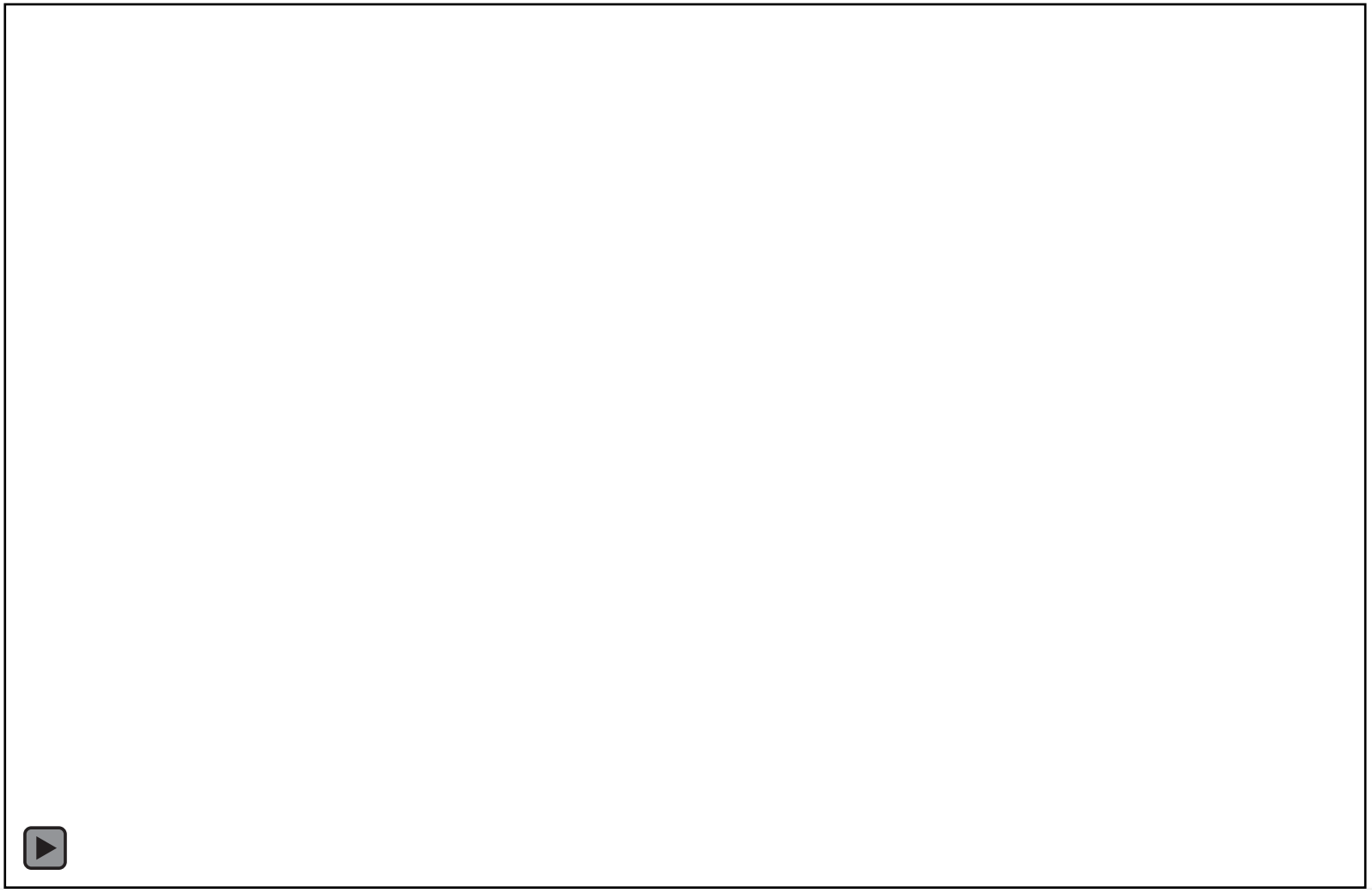

```python
cli.list_nodes()
```


```
{'nodes': ['HPC2', 'HPC3', 'HPC1']}
```


```python
cli.output(404308977778688)
```


```
{'Output': ['HPC1\n', 'HPC3\n', 'HPC2\n']}
```


Flux Python Client Demo



Challenges

- Took a while to navigate the documentation.
- User authentication difficulties
  - Credentials don't expire and there's not a straightforward way to clear the FastAPI's HTTP Auth credentials yet.
  - Even after a successful login, jQuery module is not correctly interpreting credentials.
  - Submitting jobs with authentication runs with "sudo" command, potentially opening up for security issues.
- RestAPI is still under development:
  - E.g. Logout API request is still in the work

Future Work and Goals with Flux

- Improve user authentication process and documentation for it.
- User submitted jobs to run as themselves (multi-user) on the machine instead of Flux instance user (single-user).
- Implement more Python API methods to match the number of commands of Flux.

Cited Resources

- Flux RestAPI Documentation – http://flux-framework.org/flux-restful-api/index.html
- Flux RestAPI Repository -- https://github.com/flux-framework/flux-restful-api

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC