Hybrid Reusable Computational Analytics Workflow Management with Cloudmesh

Gregor von Laszewski

References

- https://cybertraining-dsc.github.io
- http://laszewski.github.io

Acknowledgments

- NSF
- University of Virginia
- BII

Acknowledgements

- **NIST**: NIST 60NANB21D151T
- **NSF**: CyberTraining for Students and Technologies from Generation Z with the award numbers 1829704 and 2200409.
- J.P. Fleischer, University of Virginia
- Robert Knuuti
- Geoffrey C. Fox, University of Virginia
- Wo Chang, NIST
- NIST Analytics Services Working Group
- Other students that contributed
- University of Virginia Biocomplexity Institute

Abstract

Over the last several years, the computation landscape for conducting data analytics has completely changed. While in the past, a lot of the activities have been undertaken in isolation by companies, and research institutions, today's infrastructure constitutes a wealth of analytics services offered by a variety of providers that offer opportunities for reuse, interactions while leveraging service collaboration, and service cooperation. In this talk, we project our vision to develop an analytics services framework to integrate reusable hybrid multi-services across different cloud infrastructure and service providers as well as on-premise infrastructure and services. It includes (a) analytics services that explicitly target the intersection of hybrid multi-provider analytics services, (b) the integration of sensor networks into the service data, (c) the integration of advanced modelling services as part of analytical tasks such as earthquake prediction, and (d) the coordination of these services. We showcase how our vision can integrate leveraging service composition with cooperation and competition analytics services. We demonstrate the activities while utilizing benchmark analytics services as used by the MLCommons science benchmark working group.

Resources

- Gregor von Laszewski, J. P. Fleischer, Geoffrey C. Fox (2022). Hybrid Reusable Computational Analytics Workflow Management with Cloudmesh. *arXiv*.
- Gregor von Laszewski, Anthony Orlowski, Richard H. Otten, Reilly Markowitz, Sunny Gandh, Adam Chai, Geoffrey C. Fox, Wo L. Chang (2021). Using GAS for Speedy Generation of Hybrid Multi-Cloud Auto Generated AI Services. *IEEE COMPSAC 2021: Intelligent and Resilient Computing for a Collaborative World 45th Anniversary Conference*.
- There are many more references to this at: https://laszewski.github.io/publication/

- https://github.com/cloudmesh/cloudmesh-cc
- https://github.com/cloudmesh/cloudmesh-sbatch
- https://github.com/cloudmesh/cloudmesh-nlp
- https://infomall.org
- https://cybertraining-dsc.github.io

Approximate Content

- Abstract
- About me
- About you
- Communication
- Motivation: Large scale cyberinfrastructure
- Rivanna and Slurm
- cloudmesh-sbatch
- cloudmesh-cc

About me

Organizations

- PhD in Computer Science
- Research experience from:
    - German National Institute of Computer Science (GMD) now Frauenhofer (Germany)
    - NASA
    - Argonne National Laboratory
        - Sabbatical: RIT
    - Indiana University
    - University of Virginia

Topics

- AI (Genetic Algorithms)
- Cyberinfrastructure
- Metacomputing
- Grid Computing
- Cloud Computing
- Education
- Applied AI
- Benchmarking

About you: Who will succeed?

- Be self motivated …
- Be interested in learning new things …
- Have the desire to experiment …
- Like to communicate what they learned to others …
- Like to discover on their own new things …
- Do not shy away when it becomes difficult …
- Do not say, I do not know because I am a beginner (and expect solutions from others) …
  - Instead they say: Yes I do not know but I will find out and will have a solution for you in an hour, tomorrow, next week, …

- Do not expect to just consume information but produce information actively …
- Have the desire to apply what is learned after the experience is over …
- Be sensitive to deadlines,
- (Possibly do this also as hobby …)

- **Work with others and communicate !!! – Science is done in large teams !!!!!!**
  - **Do the things you have been asked**
  - **Do more than you have been asked to do**
    - **When a task is not done by others, jump in and offer help**
    - **Learn together**

About others: Who will not succeed? Examples

- Always say this was not my task …
- Have 4.0 GPA
  - and nothing else
  - and feel superior to others with lesser GPA
    (GPA is not necessarily measurement of if you succeed in research)
  - and be competitive and not collaborate in order to
- Do it for the money
- Have multiple jobs and you can not balance which is more important /only do this as is convenient to get money
- Only do things that you enjoy/ are fun
- Do not push your boundaries
- Lie / Pretend to know something to give good impression

.

Communication

- Zoom
  - Gregor: https://virginia.zoom.us/my/meetgregor
- Web Pages
  - https://cybertraining-dsc.github.io
  - http://laszewski.github.io
  - http://infomall.org/tutorials
- Discord: Only Bii_dsc_community
  - Only applies to students working with Geoffrey (you got an email from me)
  - Discord invitation https://discord.gg/5q8VFEA7
  - Discord is used by Gregor for Rivanna communication of dsc team and general teaching lessons
  - When using a name on discord, for this please use a name that reflects firtsname and l astname so we can easier associate you with you

Communication: Presenting your own work …

- Check with your advisor
- Gregors recommendations
  - Presentations: Google slides (sometimes powerpoint)
    - no dark backgrounds for main text
    - use format from UVA
    - drafting in google docs ok
  - Papers: LaTeX in github or overleaf.com
    - focus on content
    - superb reference management
    - can easily be adapted to format of journal, conference, poster
  - Documentation
    - markdown files in github
  - Zoom meetings
    - do not use dark background in terminals you share
    - when using shells use white background dark font

Communication: Creating your documentation

- Consult with your advisor
- Documents
  - Collaboratively
    - overleaf.com
    - google docs
    - github + editor
  - Editors:
    - emacs (aquamacs on mac)
    - pycharm (possibly vscode)
    - graphics: powerpoint in 4:3
  - Spellchecker
    - Grammarly (copy/paste)
  - Content
    - focus on specifics and do not spend hours on writing about why AI is the best thing since sliced bread.
    - do not rely on document generators
      - do not rely on chatgpt/bard as it still created wrong content and “invents” things that are not true
- Presentations
  - google slides, powerpoint

Motivation: Why advanced cyberinfrastructure?

- **Modern scientific and industrial research requires sophisticated research infrastructure**
  - Analysis is done often with computers
  - A desktop or laptop is just one small tool
  - Other research infrastructure is accessible via the internet

  => We call it **cyberinfrastructure**

- **Cyber Infrastructure needs to provide**
  - Storage to store large amounts of data (Petabytes and beyond)
  - Fast networks to transfer data quickly between infrastructure resources
  - Specialized compute resources
    - Large computational resources as found in super computers and clouds
    - Large numbers of accelerators for Machine and deep learning

- **Cyberinfrastructure must be general to address many different application areas**
  - physics, health care, transportation, finance, natural language processing, sports, lifestyle, …

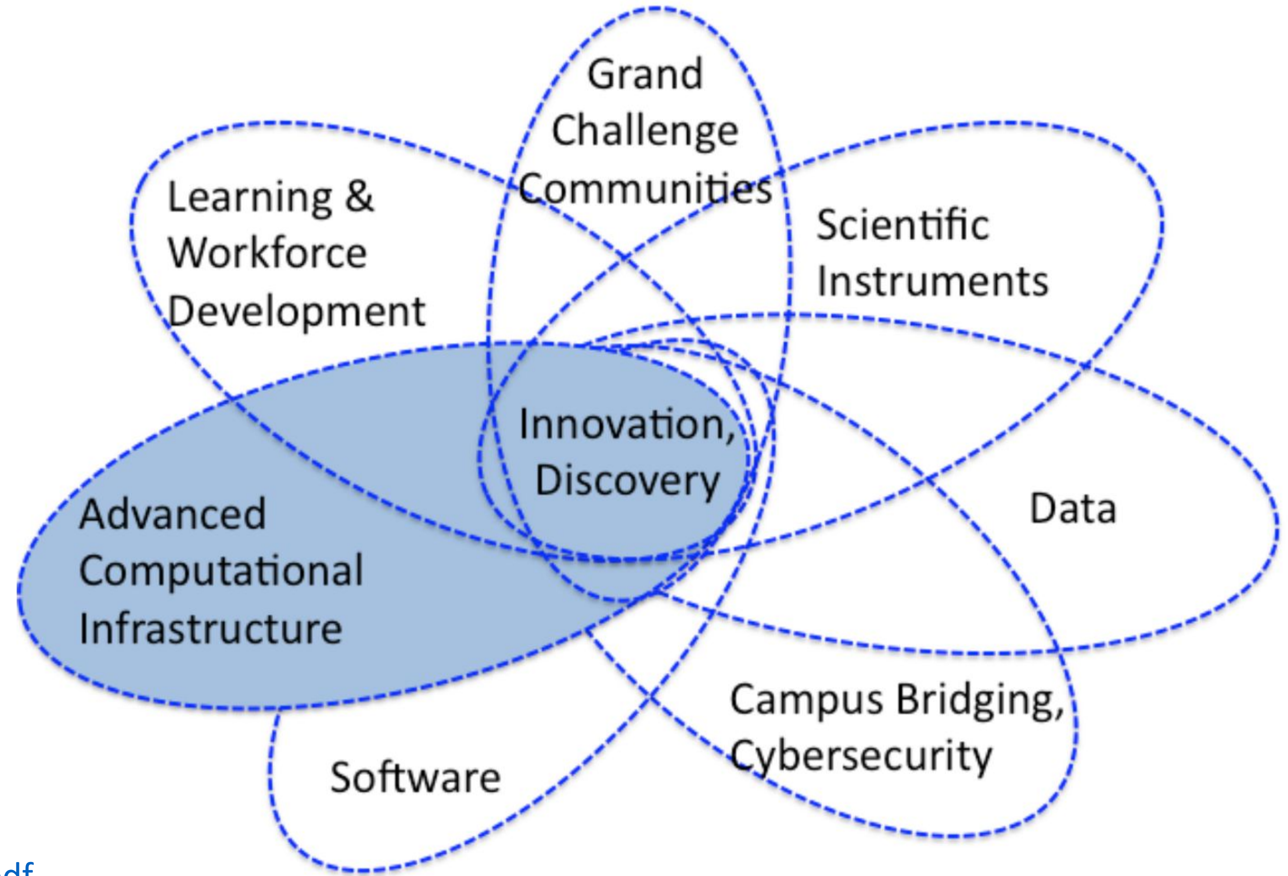
- **This is a daunting task that excludes efficient utilization and participation by many**

  => Consequence: You need to learn how to use advanced cyberinfrastructure

Motivation:
The Big Picture (NSF) (older but still accurate)

- Innovation and discovery is driven by a combination of efforts addressing issues while providing cyberinfrastructure to support solving them. This includes areas as depicted in the Figure.

Cyberinfrastructure Framework for the 21st Century

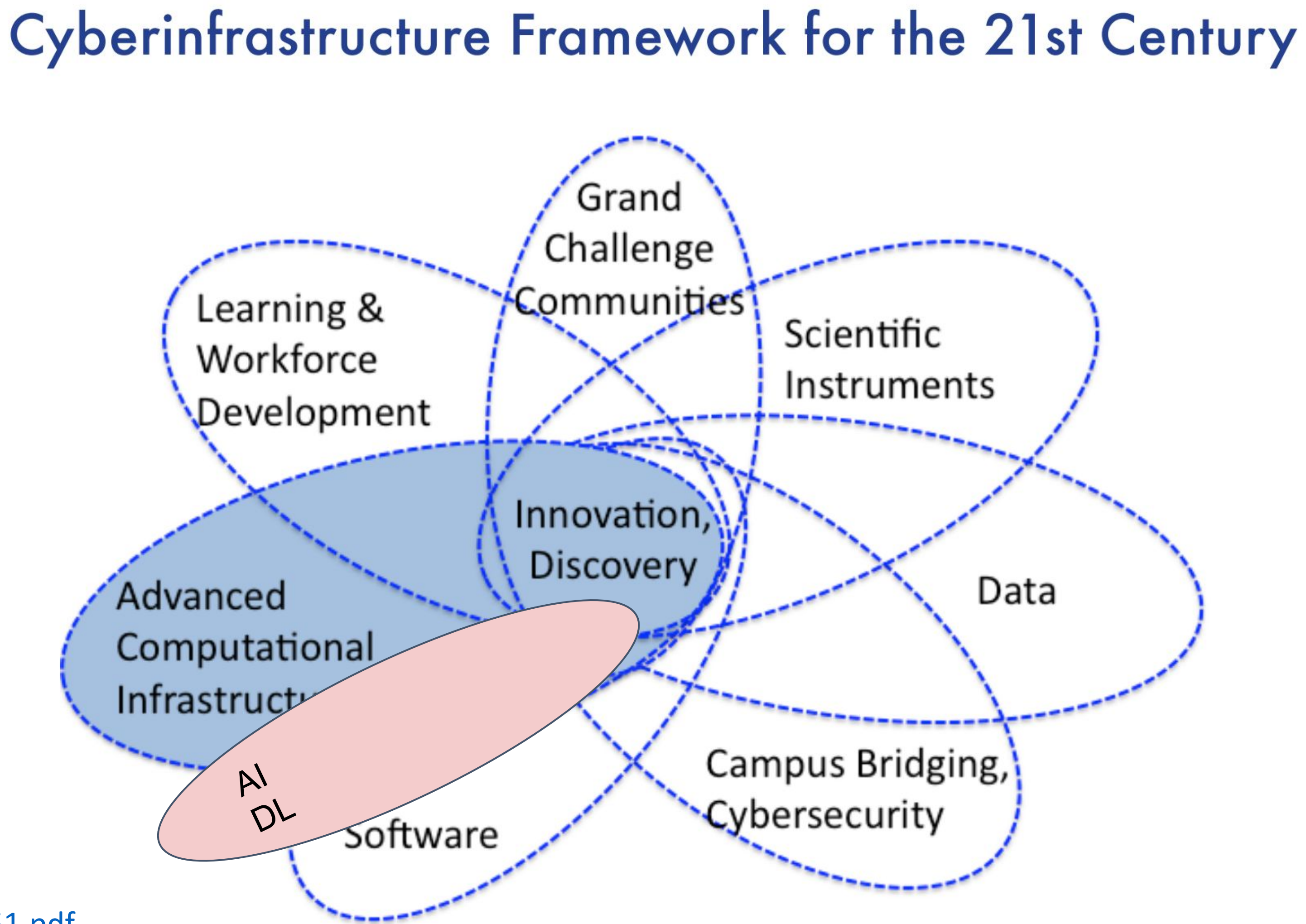

Grand Challenge Communities
Learning & Workforce Development
Scientific Instruments
Innovation, Discovery
Advanced Computational Infrastructure
Data
Software
Campus Bridging, Cybersecurity

- https://www.nsf.gov/pubs/2012/nsf12051/nsf12051.pdf

Motivation:
The Big Picture (NSF) (add AI)

- Innovation and discovery is driven by a combination of efforts addressing issues while providing cyberinfrastructure to support solving them. This includes areas as depicted in the Figure.

- https://www.nsf.gov/pubs/2012/nsf12051/nsf12051.pdf


Cyberinfrastructure Framework for the 21st Century
Grand Challenge Communities
Learning & Workforce Development
Scientific Instruments
Innovation, Discovery
Data
Advanced Computational Infrastructure
AI
DL
Software
Campus Bridging, Cybersecurity

Motivation: What will you learn?

- **We can not cover everything in such a short period … but … this is cool …**

- **Introduction to advanced Cyberinfrastructure**
  - Documenting and reporting tools for Science
  - How to use your computer to access advanced cyberinfrastructure
  - Supercomputers and high performance computers for science
  - GPUs for science
  - Python for Science
    - Jupyter notebooks for scientific research
    - Python ecosystem for scientific research
  - ML and DL for science
  - Applications for science

Motivation: Information about a lot of …!

- Develop and contribute modules to

  https://cybertraining-dsc.github.io/docs/modules/

- Improve Python modules

  https://laszewski.github.io/publication/las-2020-book-python/

- Improve Linux documentation for Cloud and science computing

  https://laszewski.github.io/publication/las-2020-book-linux/

- Create a Web site similar to

  https://laszewski.github.io/

- Learn and improve selected sections from

  https://laszewski.github.io/publication/las-2020-book-cloudeng/

- create books / proceedings from distributed information in github md files
  - examples: see my book page on github, but also MPI example

books will be updated, so links may change
go to https://laszewski.github.io/ for online books

help improve via github

Tutorials by Gregor et. al.

• Introduction and Advanced Batch Jobs	
• **Rivanna**	Basic Rivanna tutorial
• **Rivanna Pod**	Using the new Rivanna Pod (needs likely to be updated)
• **Rivanna and Singularity**	We recommend to use Singularity which requires special permissions
• File Transfer	
• **Rclone on Rivanna**	Using Rclone to upload and download from cloud services
• **Globus**	File transfer with Globus
• General Cybertraining	
• **Cybertraining**	Cybertraining Links
• **Raspberry Pi Cluster**	Build your one cluster with Raspberry Pi’s
• **MPI with Python**	The best MPI resource for python that I know about ;-)

SSH

- You need to know what ssh is and how to use it
- We have tutorials on this and I can give practical lecture on this

- Use bash
  - On windows install gitbash
-

  Commands:
  - ssh-keygen, ssh-add, ssh-agent
  - ssh
  - scp
  - /.ssh/config
  - vpn

cloudmesh installation

- convenient installation via pip/conda but do not install in $USER
  - python -m venv /scratch/$USER/ENV3
  - source /scratch/$USER/ENV3
  - pip install cloudmesh-common
  - pip install cloudmesh-cc
  - pip install cloudmesh-sbatch
  - cms help (important for initialization)
- Now you can use it
- This is just a subset … related to this talk
  - there are many more cloudmesh components. You can contribute

UVA | SCHOOL *of* DATA SCIENCE

Slurm Introduction

UVA Rivanna

Robert Knuuti
Gregor von Laszewski
laszewski@gmail.com

slurm
workload manager

https://docs.google.com/presentation/d/1mqP0QgYxbESCCDnlptTFp1HJvpAN_RW2tIX5-dDBSQ0/edit?usp=sharing

What is Slurm?

- Cluster and Job management ecosystem
- Provides a series of interactive and batch tools based on a reservation request
- The system ensures that users have confidence that they will have reserved access to system hardware, and to enforce limits on users to prevent abuse.

- Fun Fact: Slurm is what powers all self-service capabilities on Rivanna, if you've requested a Desktop, JupyterLab, or RStudio from the OnDemand portal, you've used a slurm script that the Rivanna team has written to setup all these tools.

How to request an Allocation

There's two ways

Interactive (ijob)

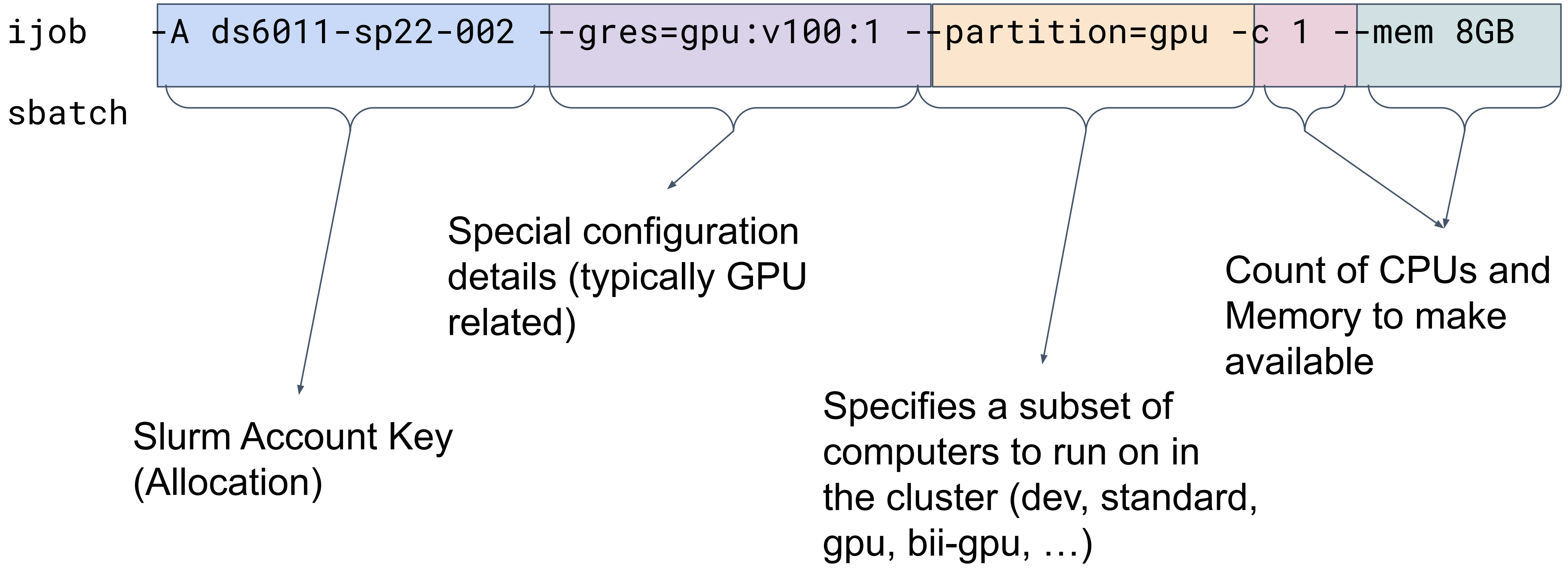
- You place you request and are placed in a queue.
- Your prompt will pause until the hardware needed for the time of your job can be made
- You are then presented with a prompt within the allocation for the duration of your request
- Once you exit, the allocation is revoked and you are returned to your normal prompt.

Batch (sbatch)

- You write a shell script that has annotations to specify your reservation needs and the commands to be run once the resources are available.
- You issue the command, and the job is registered in the background and you can continue work.
- Your job will be placed in the global queue in the background, allowing you to log off or run asynchronous jobs

Both commands use common arguments keeping the configurations similar

Parts of a Slurm Reservation Request



```
ijob     -A ds6011-sp22-002 --gres=gpu:v100:1 --partition=gpu -c 1 --mem 8GB
sbatch
```

Slurm Account Key (Allocation)

Special configuration details (typically GPU related)

Specifies a subset of computers to run on in the cluster (dev, standard, gpu, bii-gpu, …)

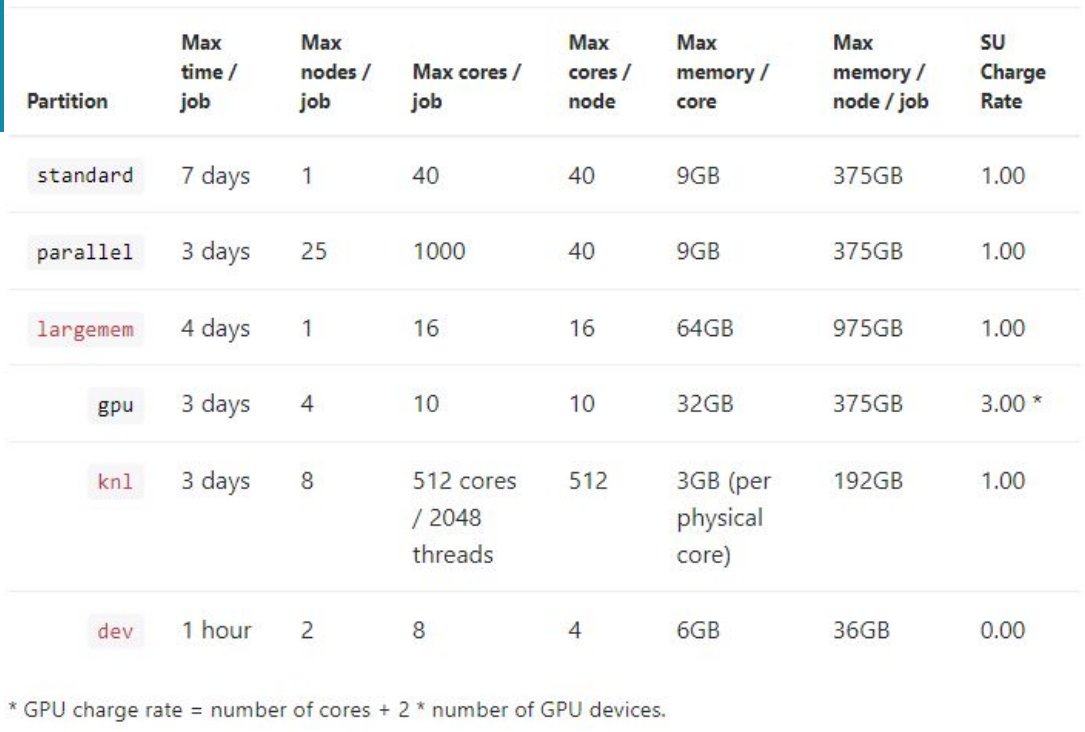
Count of CPUs and Memory to make available

This is just a subset of all the options.
See https://slurm.schedmd.com/sbatch.html for all possible arguments

How to spawn a job

1. Login to Rivanna either via SSH, or by using the cluster access at https://rivanna-portal.hpc.virginia.edu/
2. Position your desired files on rivanna (either check out your code using git, or upload it to your home directory)
3. Run the ijob / sbatch command as you need.

Expanding on Slurm Arguments



Partition	Max time / job	Max nodes / job	Max cores / job	Max cores / node	Max memory / core	Max memory / node / job	SU Charge Rate
`standard`	7 days	1	40	40	9GB	375GB	1.00
`parallel`	3 days	25	1000	40	9GB	375GB	1.00
`largemem`	4 days	1	16	16	64GB	975GB	1.00
`gpu`	3 days	4	10	10	32GB	375GB	3.00 *
`knl`	3 days	8	512 cores / 2048 threads	512	3GB (per physical core)	192GB	1.00
`dev`	1 hour	2	8	4	6GB	36GB	0.00

* GPU charge rate = number of cores + 2 * number of GPU devices.

Account → Your UVA ID must be in the allocation listed to be used

You can check your allocations by running the **`allocations`** command on rivanna

Gres → On rivanna follows the pattern `` `gpu:<card_type>:<number_of_gpus>` ``.

So for a job needing 2 v100 GPUs, you'd use the string `` `--gres=gpu:v100:2` ``

Time → You need to specify the total amount of time upfront for your allocation.

Rivanna doesn't allow jobs greater than 3 days in length

There are rules on the maximum concurrently, and your job may go to a queue.

UVA | SCHOOL *of* DATA SCIENCE

Considerations for writing sbatch jobs

- Slurm itself isn't a complex of a system to use, but writing batch jobs that maximize its abilities requires automation to be implemented.
- Your job is triggered as a shell script (bash), and ends when the script either finishes or encounters an error; it may not be immediately obvious if things worked or not.
- Your job does need to run without any type of user input or feedback (headless)
  - This is where most of the work will be.
  - Tools like lmod, papermill and apptainer/singularity become important.

- To convert a shell script to an slurm-aware script, you simply add the CLI arguments mentioned before with the prefix of `` `#SBATCH <argument-here>` `` in the file.

Reminder: you can supply any of the sbatch CLI arguments in these files.
See https://slurm.schedmd.com/sbatch.html for all possible arguments

Example sbatch script

```
#!/usr/bin/env bash

#SBATCH --job-name=mydemojob
#SBATCH --output=%u-%j.out
#SBATCH --error=%u-%j.err
#SBATCH --partition=standard
#SBATCH --cpus-per-task=1
#SBATCH --mem=4GB
#SBATCH --time=3:00
#SBATCH --account=ds6011-sp22-002

module load anaconda # we do have custom versions of python in /project/ds6011-sp22-002 for native python.

conda create -y -n demo python=3.10

conda activate demo
# your automation code here...
```

(The same configuration using ijob)

```
ijob --job-name=mydemojob --partition=standard --cpus-per-task=1 --mem=4GB --time=3:00 --account=ds6011-sp22-002

# Or in abbreviated values…

ijob -J mydemojob -p standard -c 1 -t 3:00 -A ds6011-sp22-002 --mem=4GB
```

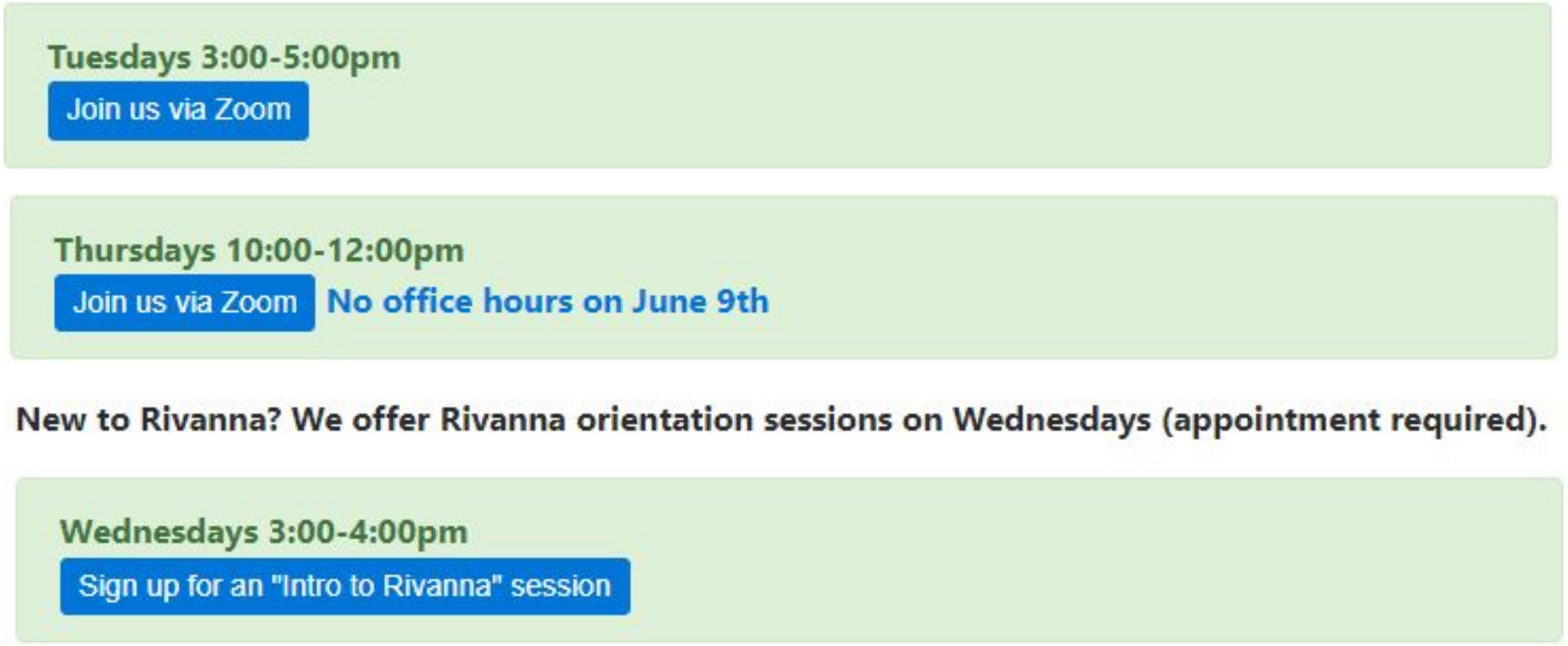
Example sbatch script with GPU

```
#!/usr/bin/env bash

#SBATCH --job-name=mydemojob
#SBATCH --output=%u-%j.out
#SBATCH --error=%u-%j.err
#SBATCH --partition=gpu
#SBATCH -c 1
#SBATCH --gres="gpu:v100:1"
#SBATCH --mem=4GB
#SBATCH --time=3:00
#SBATCH --account=ds6011-sp22-002

module load cuda cudnn
module load anaconda

conda create -y -n demo python=3.10   # needs to be improved and installed in /scratch see infomall.org tutorials

conda activate demo
# your automation code here...
```

This line is critical for CUDA and Deep Learning frameworks. If you do not load these modules, it's very likely you'll be using a CPU workloads unless you install your own version of GPU libraries.

Rivanna has regular office hours meetings

See: https://www.rc.virginia.edu/support/


Tuesdays 3:00-5:00pm

Join us via Zoom

Thursdays 10:00-12:00pm

Join us via Zoom **No office hours on June 9th**

New to Rivanna? We offer Rivanna orientation sessions on Wednesdays (appointment required).

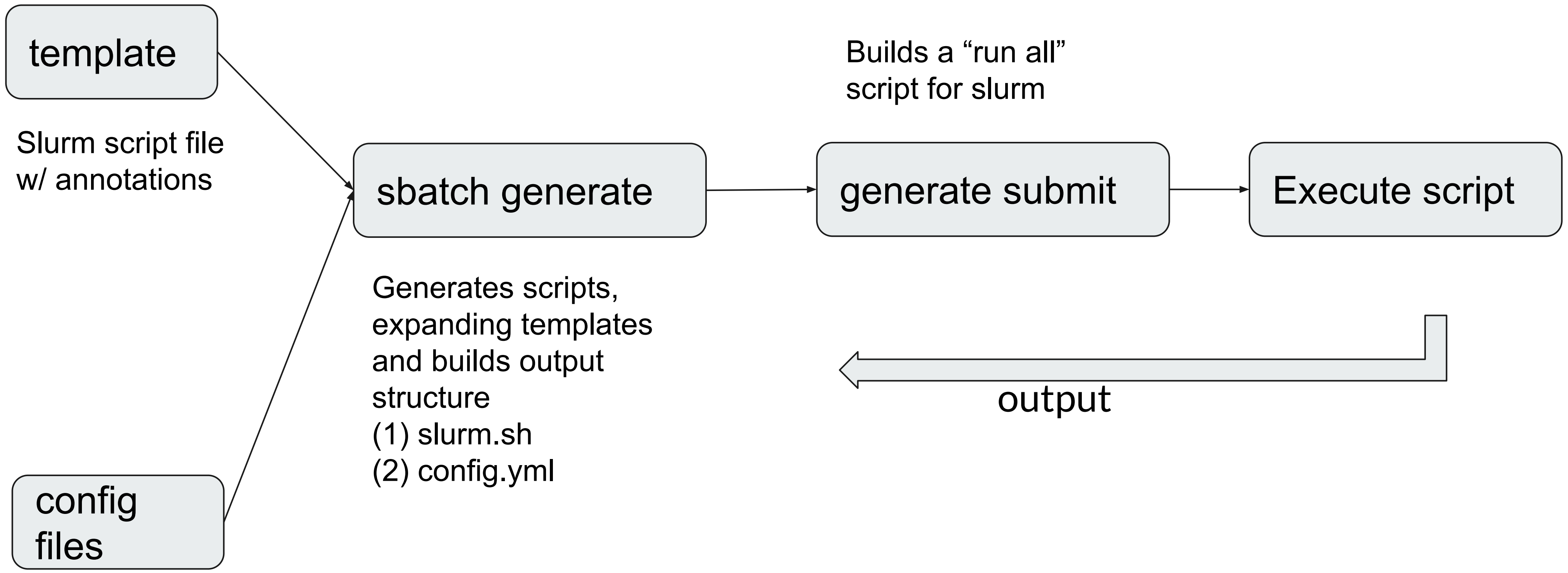
Wednesdays 3:00-4:00pm

Sign up for an "Intro to Rivanna" session

Experiment Management using Cloudmesh Sbatch

- **Slurm has a basic configuration system and runs batch jobs**
  - limits names of jobs and other parameters
  - You can use sbatch and ijob without this tool, but it becomes difficult to scale out experiments consistently.
  - Usage of queue and other resources may be too restrictive (runtime)
  - Although you can use job arrays, they may exceed runtime restrictions
- **cloudmesh-sbatch: allows multiple permutation based experiments**
  - Templated config files in YAML
  - More scalable as each job is its one job
  - Creates energy traces if added to slurm scripts with single command
  - Creates consistent output directories based on parameters
- **Example use:**
  - cloudmesh-sbatch was built as a templating and parameterization solution to address this gap, so generating hundreds of different permutations of experiments can be done using a single code baseline

How it Works



Slurm script file w/ annotations

Builds a “run all” script for slurm

Generates scripts, expanding templates and builds output structure
(1) slurm.sh
(2) config.yml

YAML file including:
(1) attributes (direct subtitutions)
(2) experiments (substitutions that are permutations)
(3) Run configurations

Examples can be found in the sbatch repo:
https://github.com/cloudmesh/cloudmesh-sbatch

Example Generation

Experiment	epoch	x	y	Experiment	epoch	x	y	Experiment	epoch	x	y	Experiment	epoch	x	y
1	1	1	10	**4**	1	4	10	**7**	1	1	11	**10**	1	4	11
2	2	1	10	**5**	2	4	10	**8**	2	1	11	**11**	2	4	11
3	3	1	10	**6**	3	4	10	**9**	3	1	11	**12**	3	4	11

slurm.in.sh

config.yaml

```
name: myexperment
cpus: 6
gpu: a100
mem: 32GB
time: 6:00:00
rivanna:
  account: ds6011-sp22-002
dir: example
experiments:
  epoch: “1,2,3”
  x: “1,4”
  y: “10,11”
```

```
#!/usr/bin/env bash
#SBATCH --job-name=epoch_{epoch}-{gpu}
#SBATCH --output=epoch_{epoch}-{gpu}.out
#SBATCH --error=epoch_{epoch}-{gpu}.err
#SBATCH --partition=gpu
#SBATCH --cpus-per-task={cpus}
#SBATCH --mem={mem}
#SBATCH --time={time}
#SBATCH --gres=gpu:{gpu}:1
#SBATCH --account={rivanna.account}

python --epoch={epoch}
```

```
#!/usr/bin/env bash
#SBATCH --job-name=epoch_2-a100
#SBATCH --output=epoch_2-a100.out
#SBATCH --error=epoch_2-a100.err
#SBATCH --partition=gpu
#SBATCH --cpus-per-task=6
#SBATCH --mem=32GB
#SBATCH --time=6:00:00
#SBATCH --gres=gpu:a100:1
#SBATCH --account=ds6011-sp22-002

python --epoch=2
```

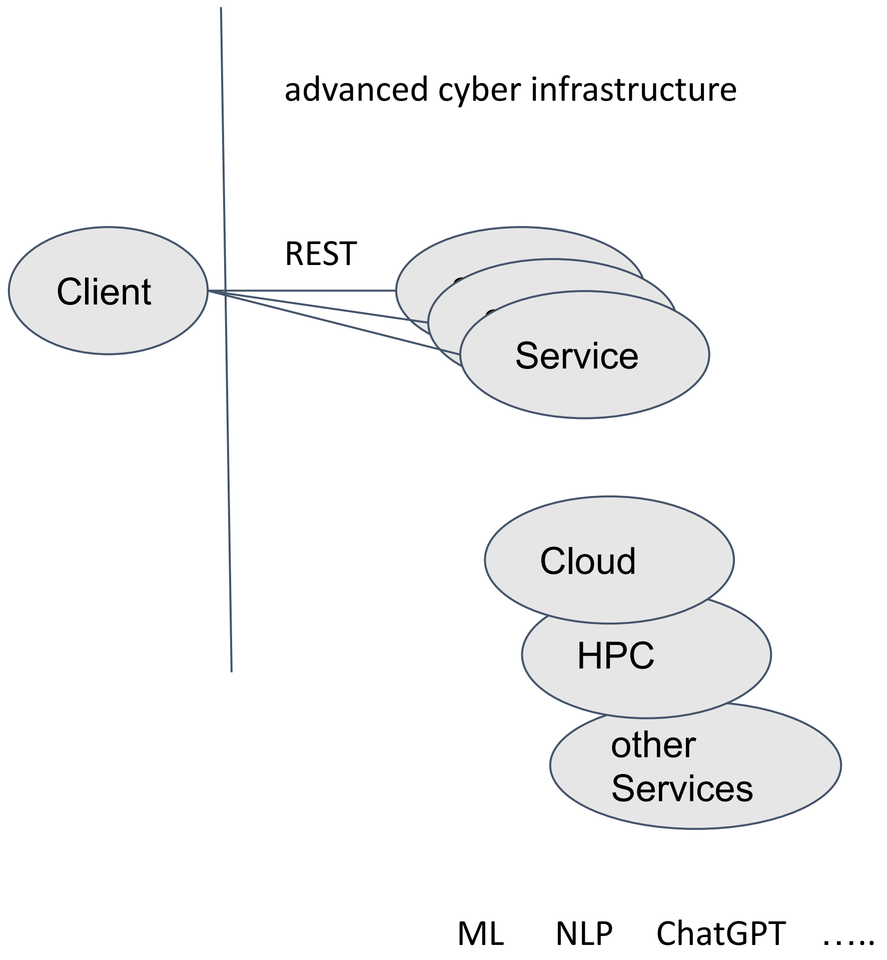
```
cms sbatch generate slurm.in.sh --setup=config.yaml
```

UVA | SCHOOL of DATA SCIENCE

Hybrid Reusable Computational Analytics Workflow Management with Cloudmesh

REST Services for AI

- Resources
  - Cloud
  - Hyperscale cloud compute centers
  - Leadership class computing facilities
- Software Concepts
  - REST (REpresentational State Transfer)
  - Microservices
  - Analytics Services
    - Traditional Methods
    - Machine Learning
    - Deep Learning
  - Data analytics as a Service



Specifications are good, but implementation is difficult.

Using GAS for Speedy Generation of Hybrid Multi-Cloud Auto Generated AI Services.

Gregor von Laszewski, et. al. (2021). *IEEE COMPSAC 2021: Intelligent and Resilient Computing for a Collaborative World 45th Anniversary Conference*.

- Scientists and beginners have difficulties comprehending the extensive software stacks to set up a service oriented architecture deployment
- Automatically generate deployable REST services from Python function descriptions
- Effort reduced from month to hours (or less …)
- Leveraging Cloudmesh
- Tested with Researchers and Students

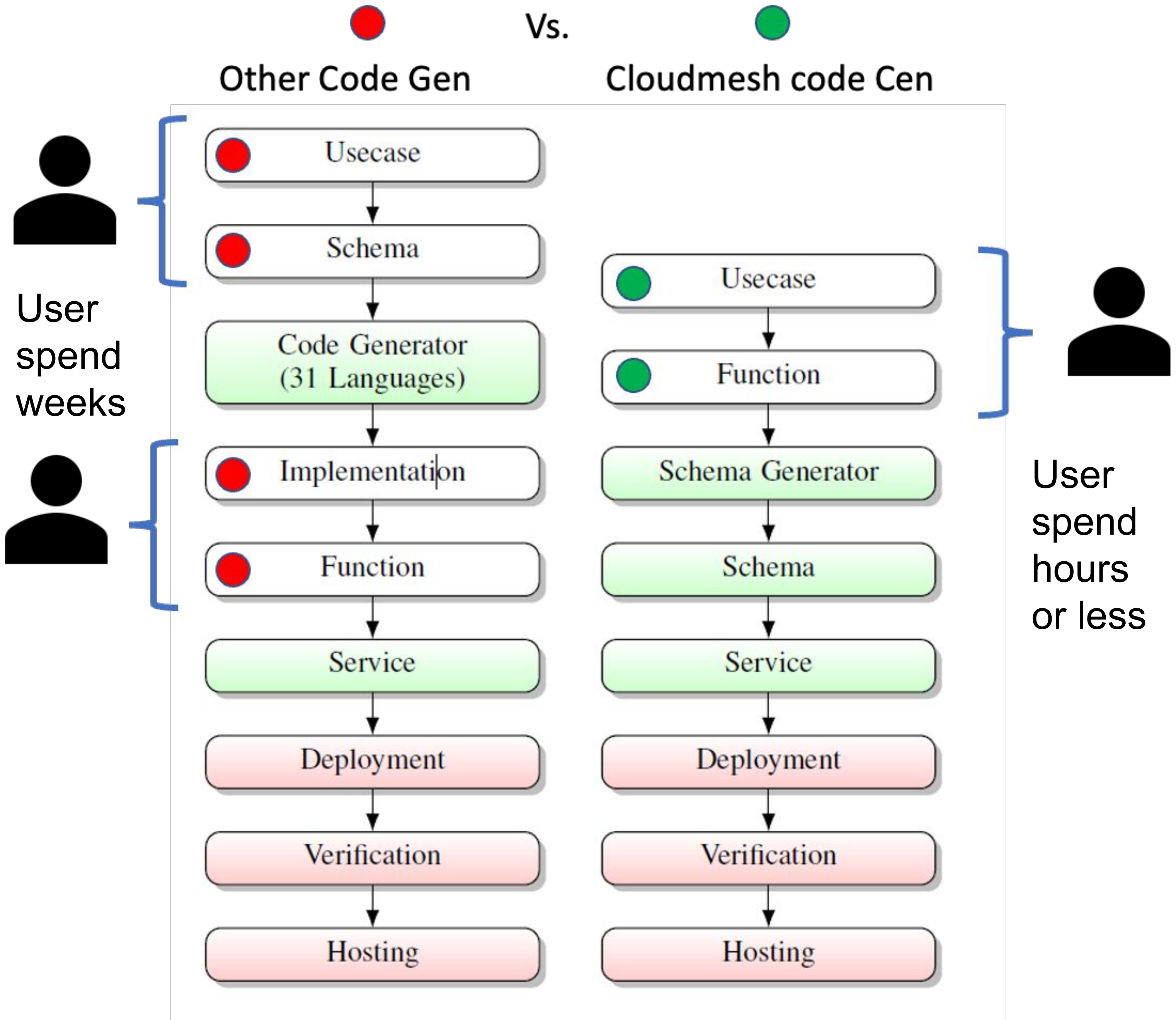


Fig. 3. Schema-based component flow to specify an analytics service.

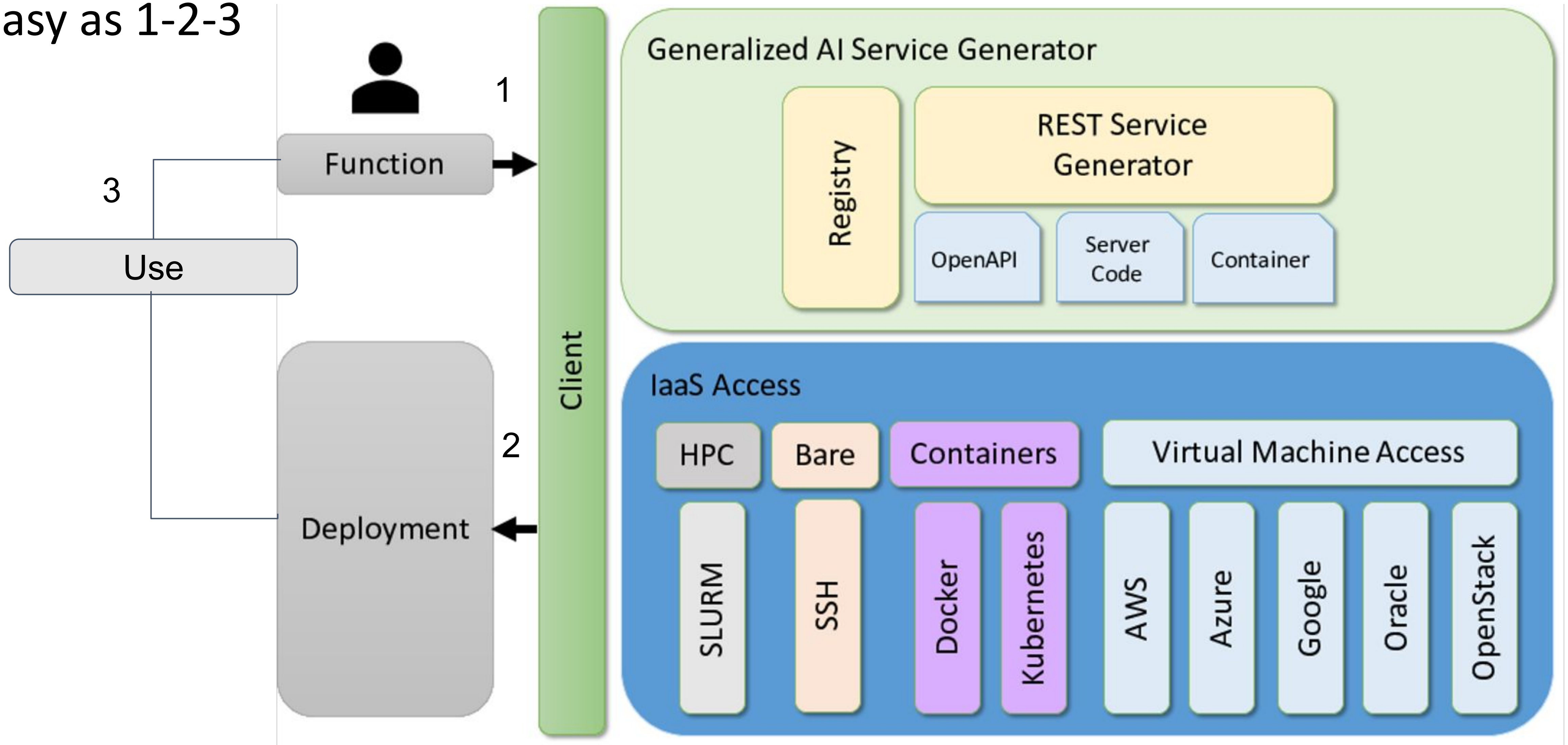
Fig. 4. Function-based component flow to specify an analytics service.

Architecture: GAS Generator

- Creates the REST service from a simple function or class definition
- Typing information integrated in the program specification
- Generates artifacts needed for deployment:
  - specification derived from the python program in OpenAPI format
  - server code that is derived from the OpenAPI specification
  - an optional container specification file (e.g., Dockerfile)
- Provides GAS Service registry including the specification description of the service and deployment information
- Includes security mechanism: currently basic authentication, but capable of incorporating other (OAuth, ApiKey, etc).

Cloudmesh GAS Architecture

As easy as 1-2-3



Why was this possible: Cloudmesh Toolkit under the hood

- Convenient Interfaces: API - Command line - Command Shell - Service via GAS
    - simple client shell that can also be used as a command line executor
- Extensible (abstractions, sophisticated easy to use module integration, GAS Generator, command shell, …)
- Abstract layer for on-demand deployment of IaaS resources across various cloud platforms
- Access to HPC and Bare metal
- Interfaces to VMs in a homogenous fashion including AWS, Azure, Google, Oracle, and OpenStack

Super simple interface:

- $ cms set cloud=AWS
- $ cms vm boot --name=machine_a
- $ cms vm ssh machine_a --command=earthquake_prediction

Leverages preferences for VMs to be staged on different clouds. No need to remember configurations

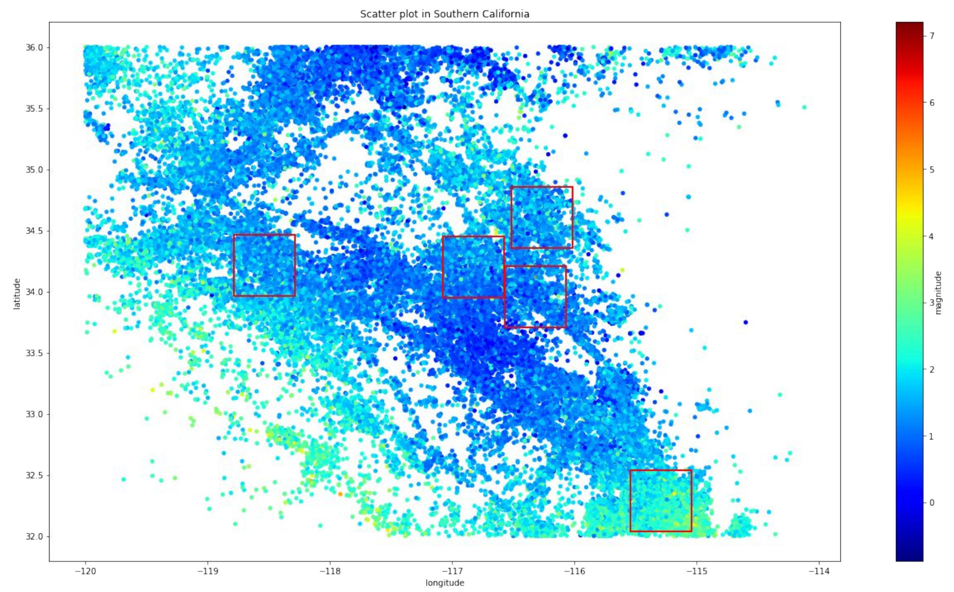
Use Cases

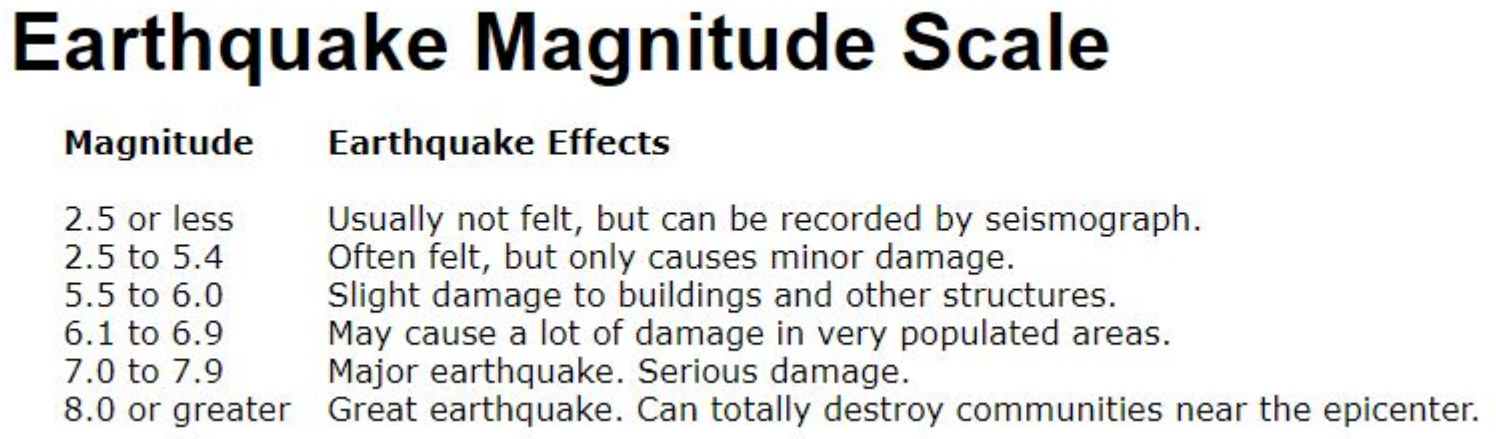
Use case Earthquake Prediction

Training
Prediction

?　　?

- Data
  - From http://www.geo.mtu.edu/UPSeis/magnitude.html
  - Earthquake shocks e.g. 2,623,328 points worldwide and 435,077 in Southern California from Jan 1990 to June 2019.
  - Each event has magnitude m, position (x,y), depth and time
  - The events have a probability of occurrence that obeys the Gutenberg–Richter law N(m) proportional to 10^{-bm} where b≈1. So there are many more small events
  - Also Energy release is proportional to 10^{cm} where c≈1.5 and so there is huge difference between a magnitude 7 and 3





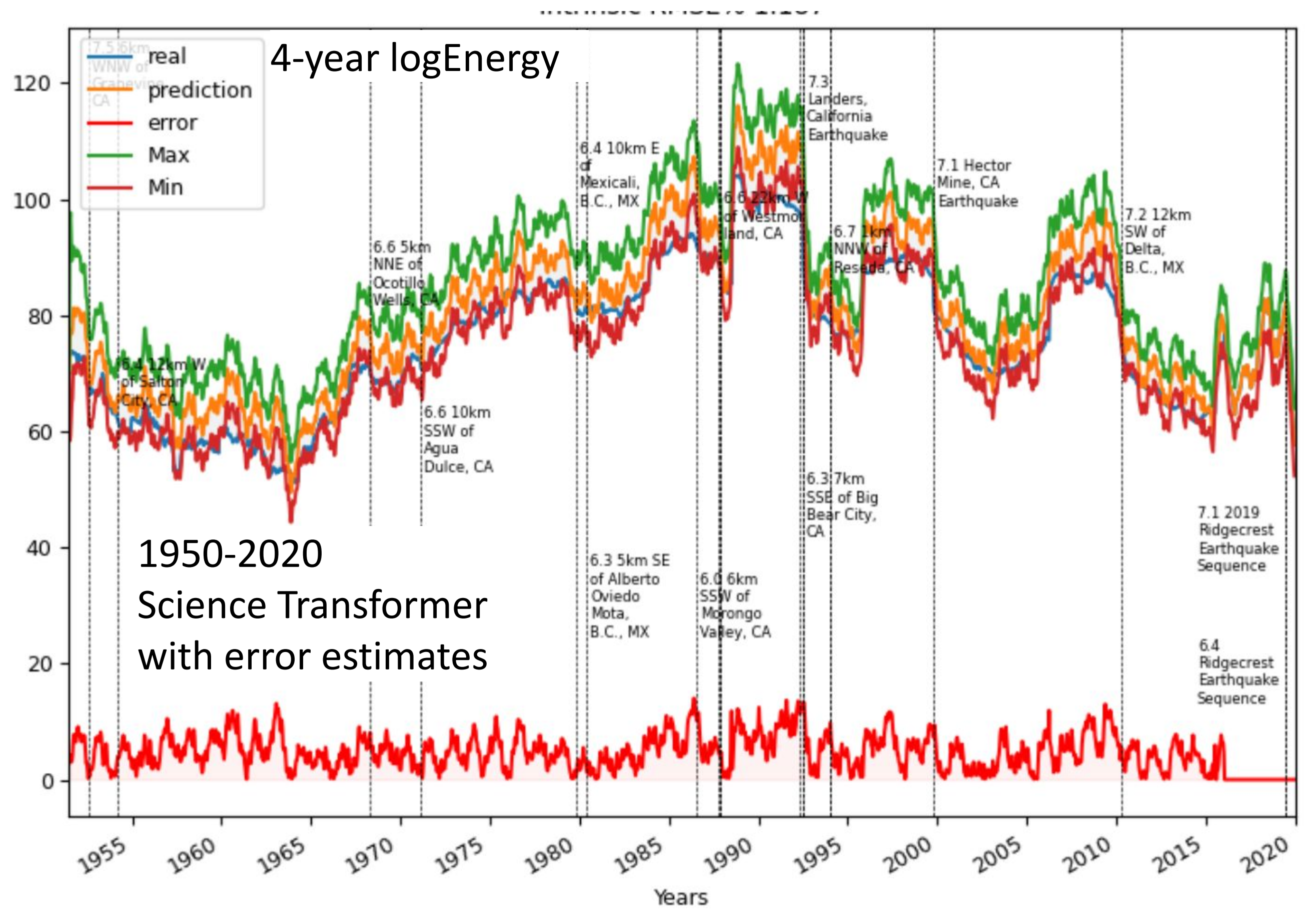
Earthquake Magnitude Scale

Magnitude	Earthquake Effects	Estimated Number Each Year
2.5 or less	Usually not felt, but can be recorded by seismograph.	900,000
2.5 to 5.4	Often felt, but only causes minor damage.	30,000
5.5 to 6.0	Slight damage to buildings and other structures.	500
6.1 to 6.9	May cause a lot of damage in very populated areas.	100
7.0 to 7.9	Major earthquake. Serious damage.	20
8.0 or greater	Great earthquake. Can totally destroy communities near the epicenter.	One every 5 to 10 years

Slide: Adapted from Fox and modified

Summary of Time Series Research

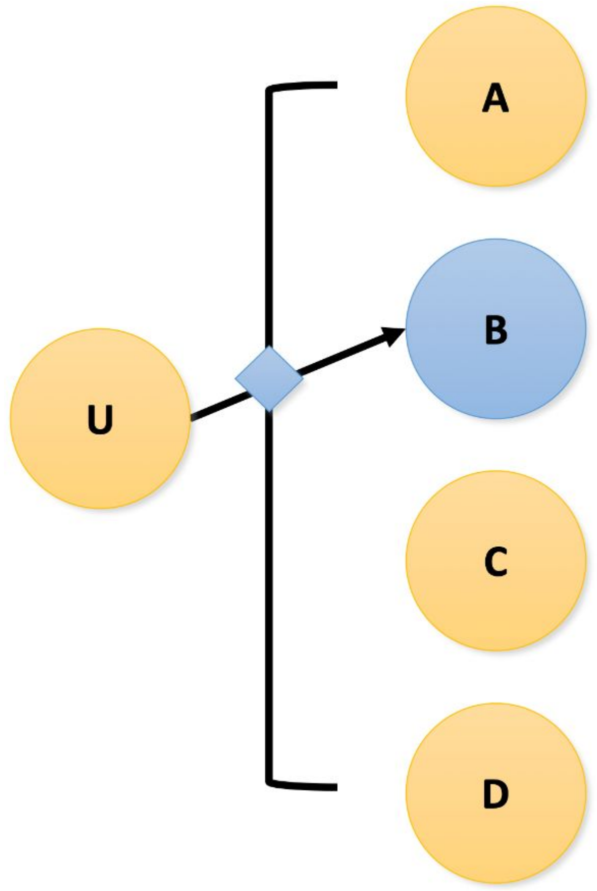
- Earthquake data from 1950-now from USGS binned daily to 4 year time intervals over 2400 regions (11km square) in Southern California; faults tagged -- typical results shown in figure
-



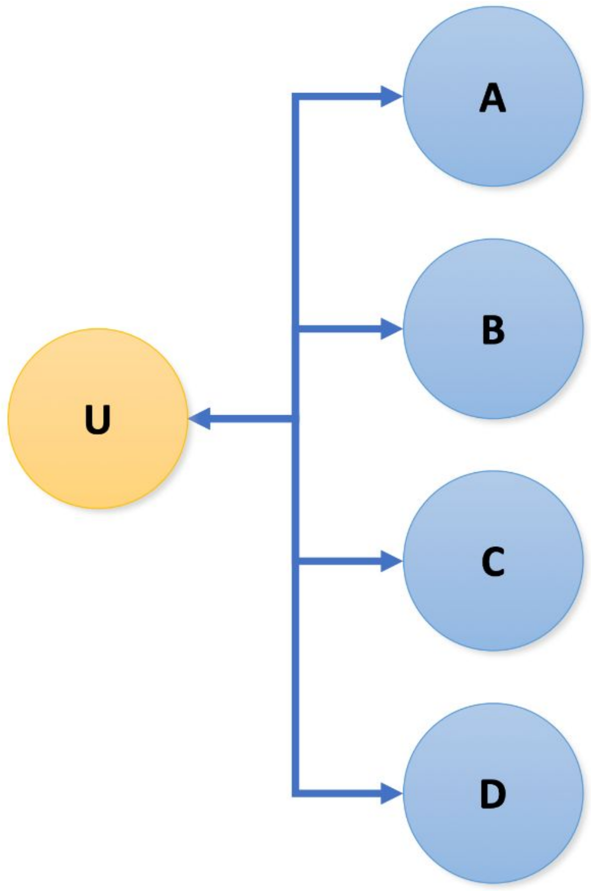
- 3 models a) LSTM b) Science Transformer from Indiana Univ. c) Google Temporal Fusion Transformer recently released by Nvidia
- Earthquake case best developed with "Criterion for success" and use of model c) to be completed and "Choice of Variable" (Energy$^{n=0.25 \text{ to } 0.5}$ to log Energy) incompletely studied.

Slide: Adapted from Fox and modified

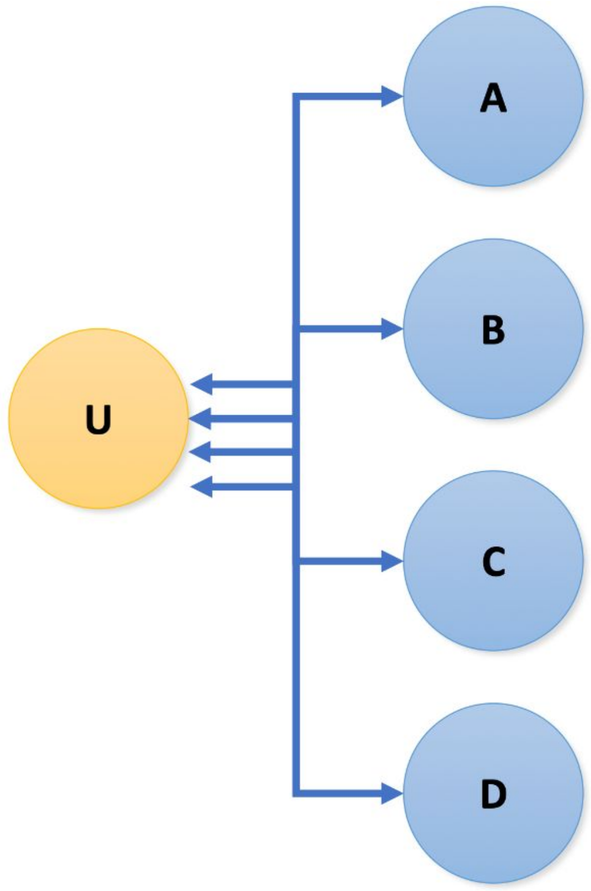
We find the following use patterns



Selection

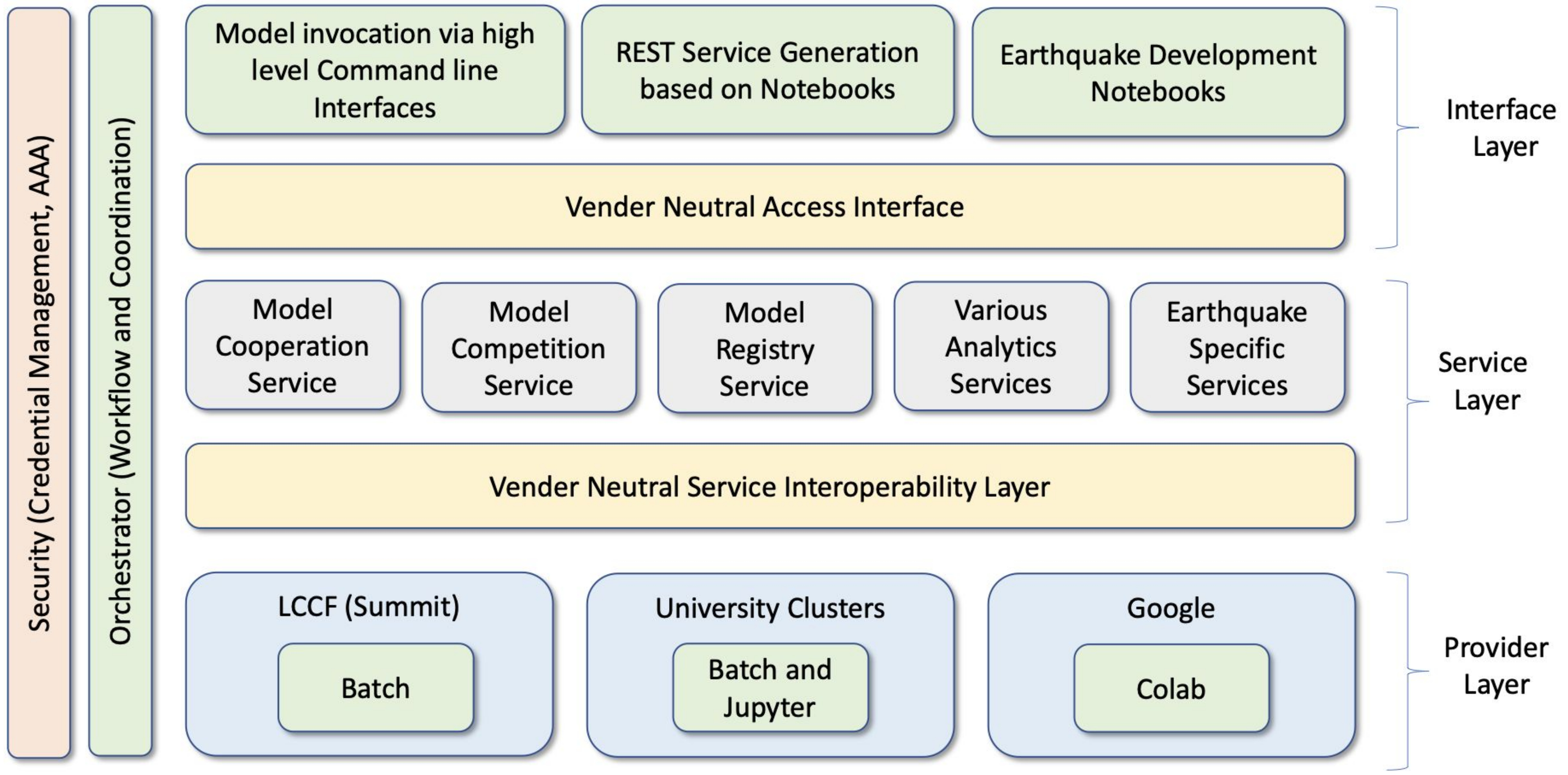


Competition

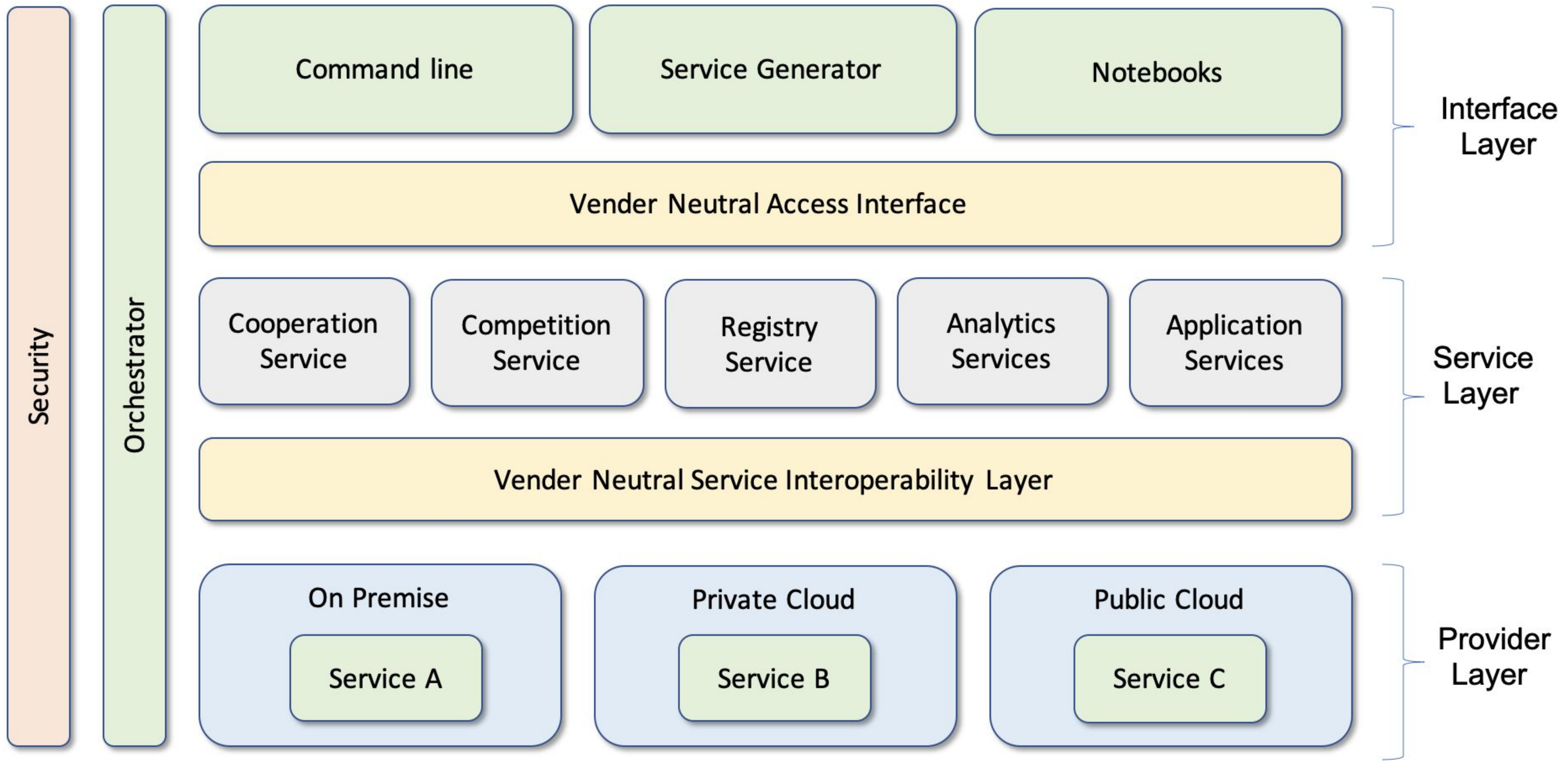


Cooperation

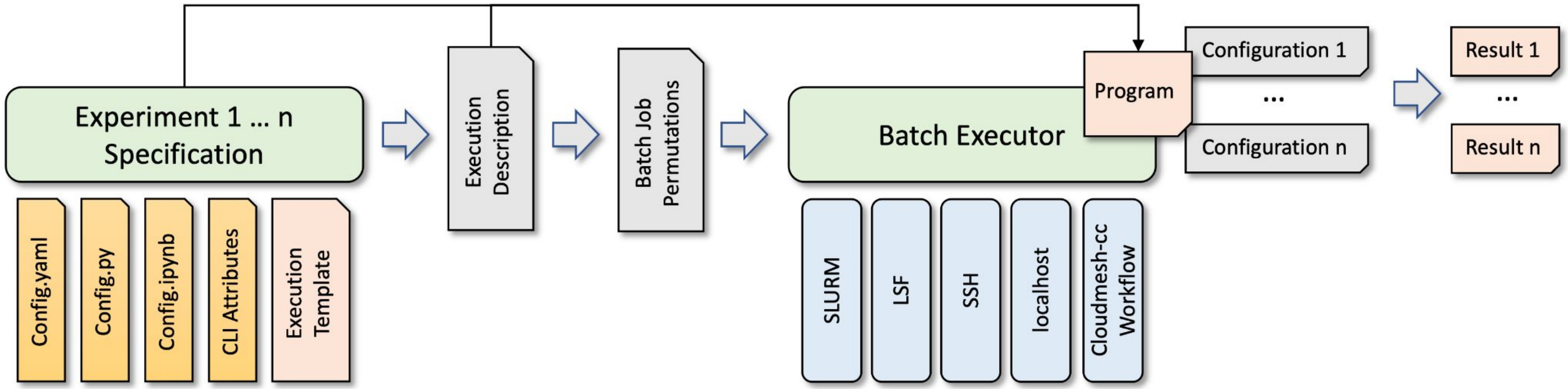
NIST Earthquake Abstract Architecture


Security (Credential Management, AAA)
Orchestrator (Workflow and Coordination)
Model invocation via high level Command line Interfaces
REST Service Generation based on Notebooks
Earthquake Development Notebooks
Vender Neutral Access Interface
Interface Layer
Model Cooperation Service
Model Competition Service
Model Registry Service
Various Analytics Services
Earthquake Specific Services
Vender Neutral Service Interoperability Layer
Service Layer
LCCF (Summit)
Batch
University Clusters
Batch and Jupyter
Google
Colab
Provider Layer

Generalized Hybrid Heterogeneous Analytics Architecture

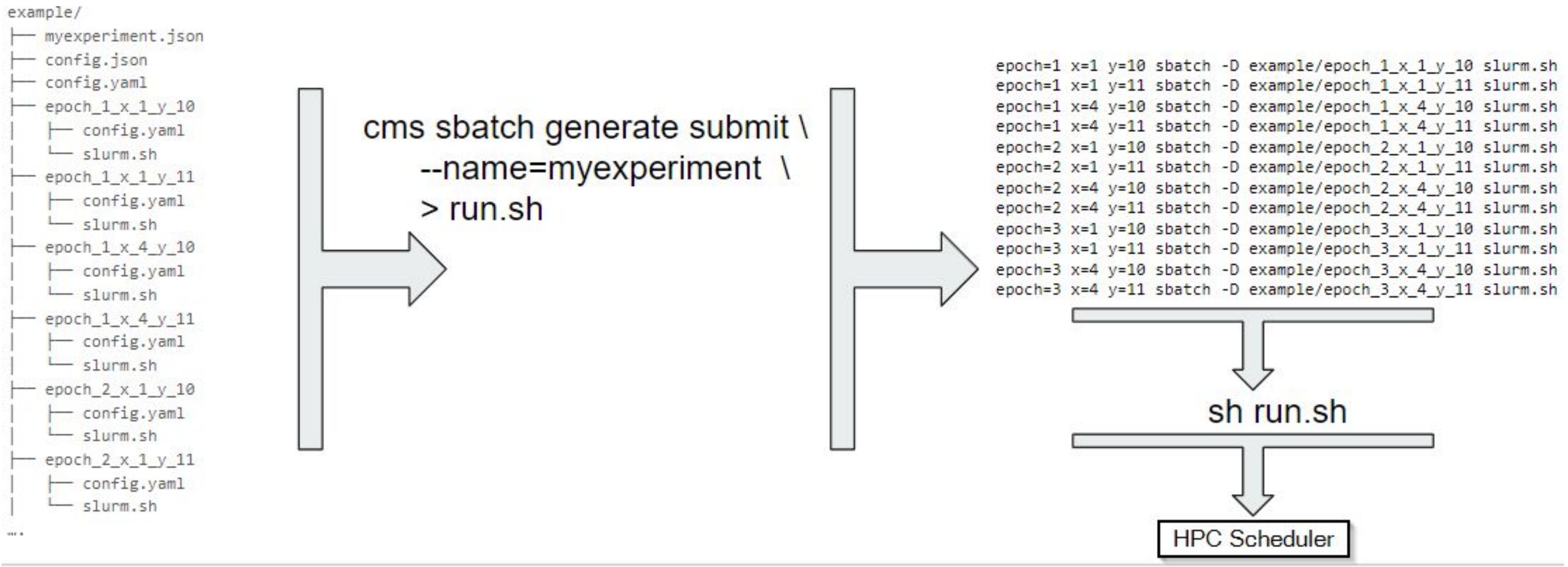


Heterogeneous On Premise Resource Scheduling



Parameterized Job Scheduling

Reproducible Experiments to coordinate many jobs via cloudmesh-sbatch



```
example/
├── myexperiment.json
├── config.json
├── config.yaml
├── epoch_1_x_1_y_10
│   ├── config.yaml
│   └── slurm.sh
├── epoch_1_x_1_y_11
│   ├── config.yaml
│   └── slurm.sh
├── epoch_1_x_4_y_10
│   ├── config.yaml
│   └── slurm.sh
├── epoch_1_x_4_y_11
│   ├── config.yaml
│   └── slurm.sh
├── epoch_2_x_1_y_10
│   ├── config.yaml
│   └── slurm.sh
├── epoch_2_x_1_y_11
│   ├── config.yaml
│   └── slurm.sh
....
```

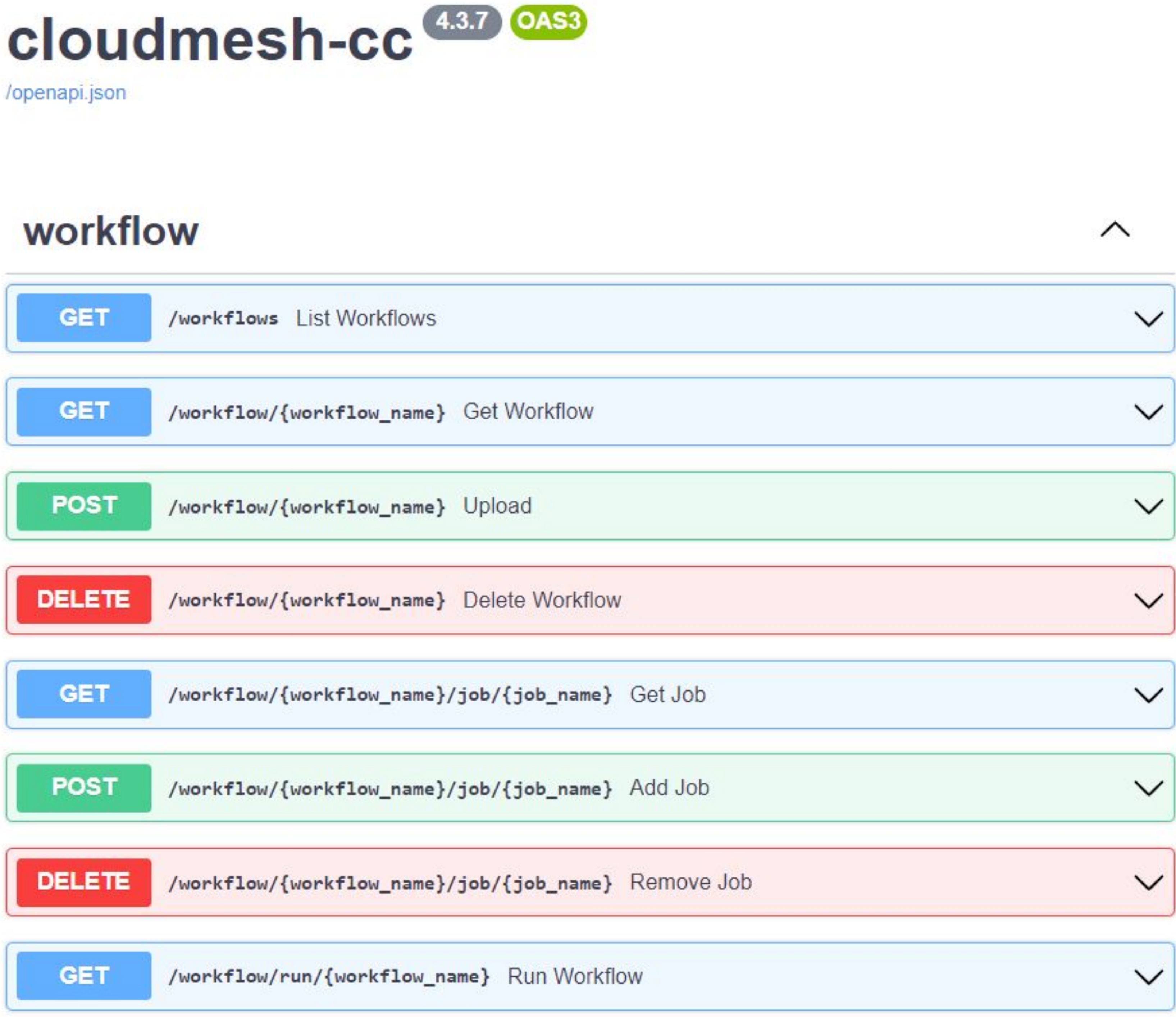
```
cms sbatch generate submit \
    --name=myexperiment  \
    > run.sh
```

```
epoch=1 x=1 y=10 sbatch -D example/epoch_1_x_1_y_10 slurm.sh
epoch=1 x=1 y=11 sbatch -D example/epoch_1_x_1_y_11 slurm.sh
epoch=1 x=4 y=10 sbatch -D example/epoch_1_x_4_y_10 slurm.sh
epoch=1 x=4 y=11 sbatch -D example/epoch_1_x_4_y_11 slurm.sh
epoch=2 x=1 y=10 sbatch -D example/epoch_2_x_1_y_10 slurm.sh
epoch=2 x=1 y=11 sbatch -D example/epoch_2_x_1_y_11 slurm.sh
epoch=2 x=4 y=10 sbatch -D example/epoch_2_x_4_y_10 slurm.sh
epoch=2 x=4 y=11 sbatch -D example/epoch_2_x_4_y_11 slurm.sh
epoch=3 x=1 y=10 sbatch -D example/epoch_3_x_1_y_10 slurm.sh
epoch=3 x=1 y=11 sbatch -D example/epoch_3_x_1_y_11 slurm.sh
epoch=3 x=4 y=10 sbatch -D example/epoch_3_x_4_y_10 slurm.sh
epoch=3 x=4 y=11 sbatch -D example/epoch_3_x_4_y_11 slurm.sh
```

```
sh run.sh
```

Cloudmesh-cc

- workflow REST service to access a compute cluster (cc)
- monitoring progress
- managing man jobs



cloudmesh-cc 4.3.7 OAS3

/openapi.json

workflow

GET /workflows List Workflows

GET /workflow/{workflow_name} Get Workflow

POST /workflow/{workflow_name} Upload

DELETE /workflow/{workflow_name} Delete Workflow

GET /workflow/{workflow_name}/job/{job_name} Get Job

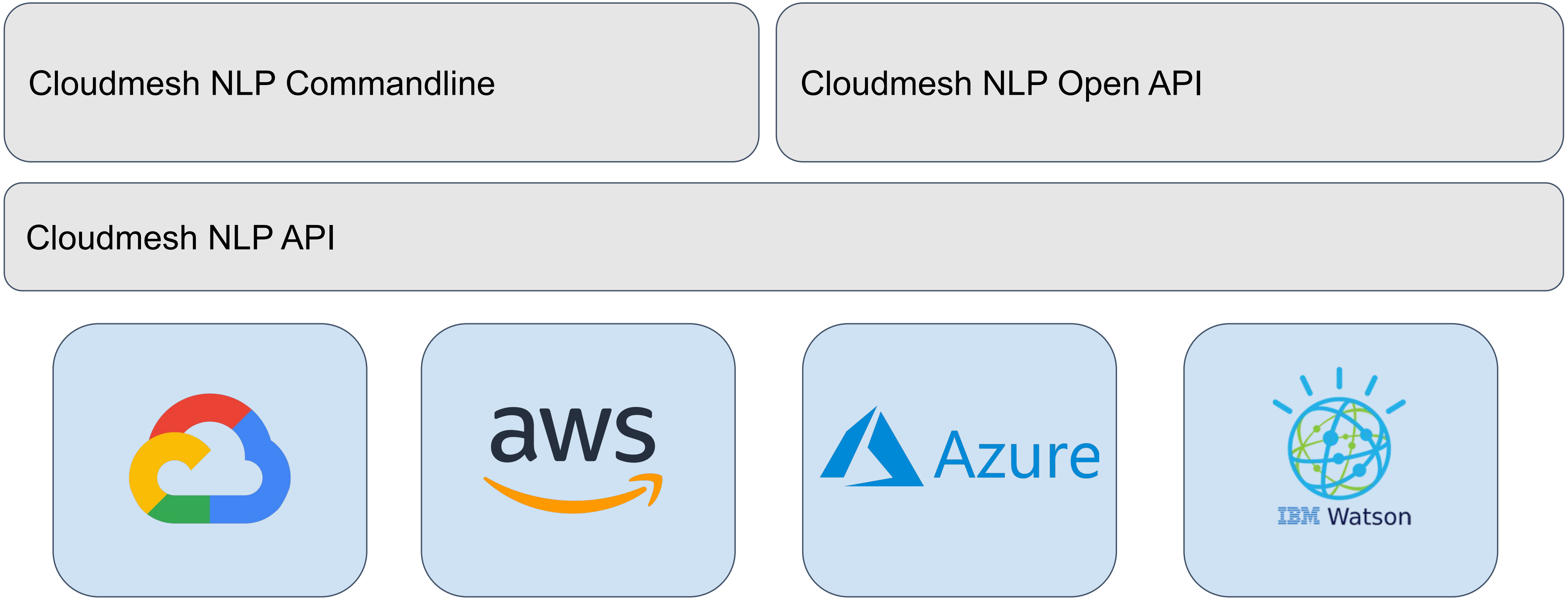
POST /workflow/{workflow_name}/job/{job_name} Add Job

DELETE /workflow/{workflow_name}/job/{job_name} Remove Job

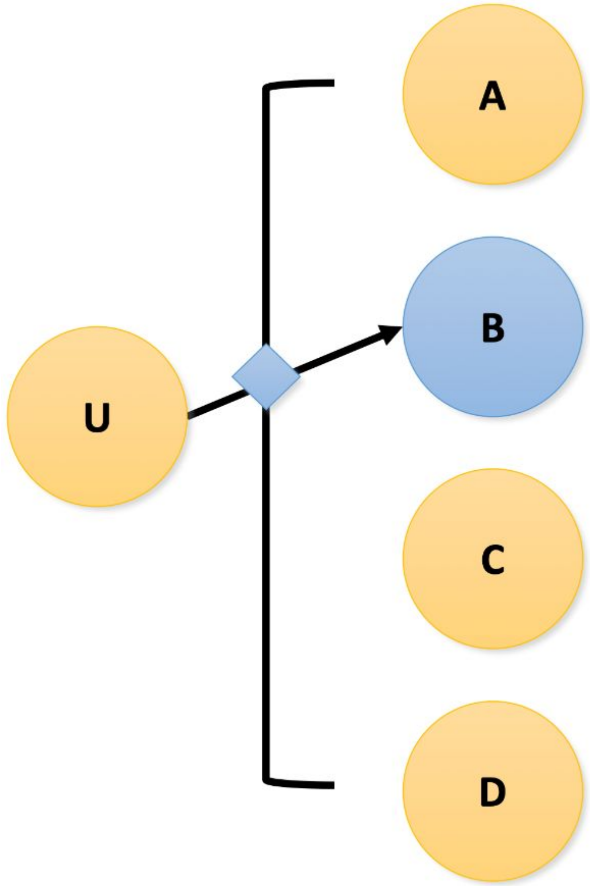
GET /workflow/run/{workflow_name} Run Workflow

Cloudmesh NLP

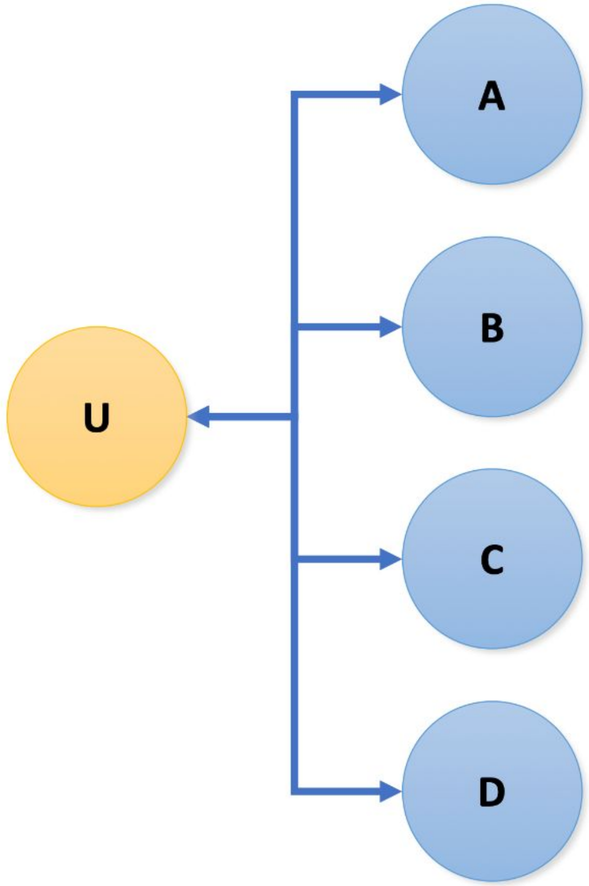
NLP Services



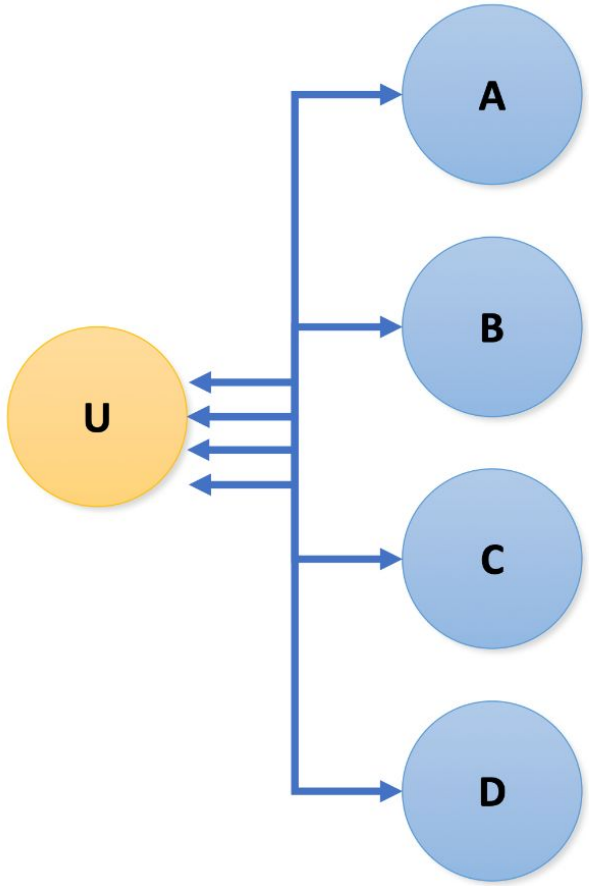
We find the following use patterns



Selection



Competition



Cooperation

Cost Comparison for Translate

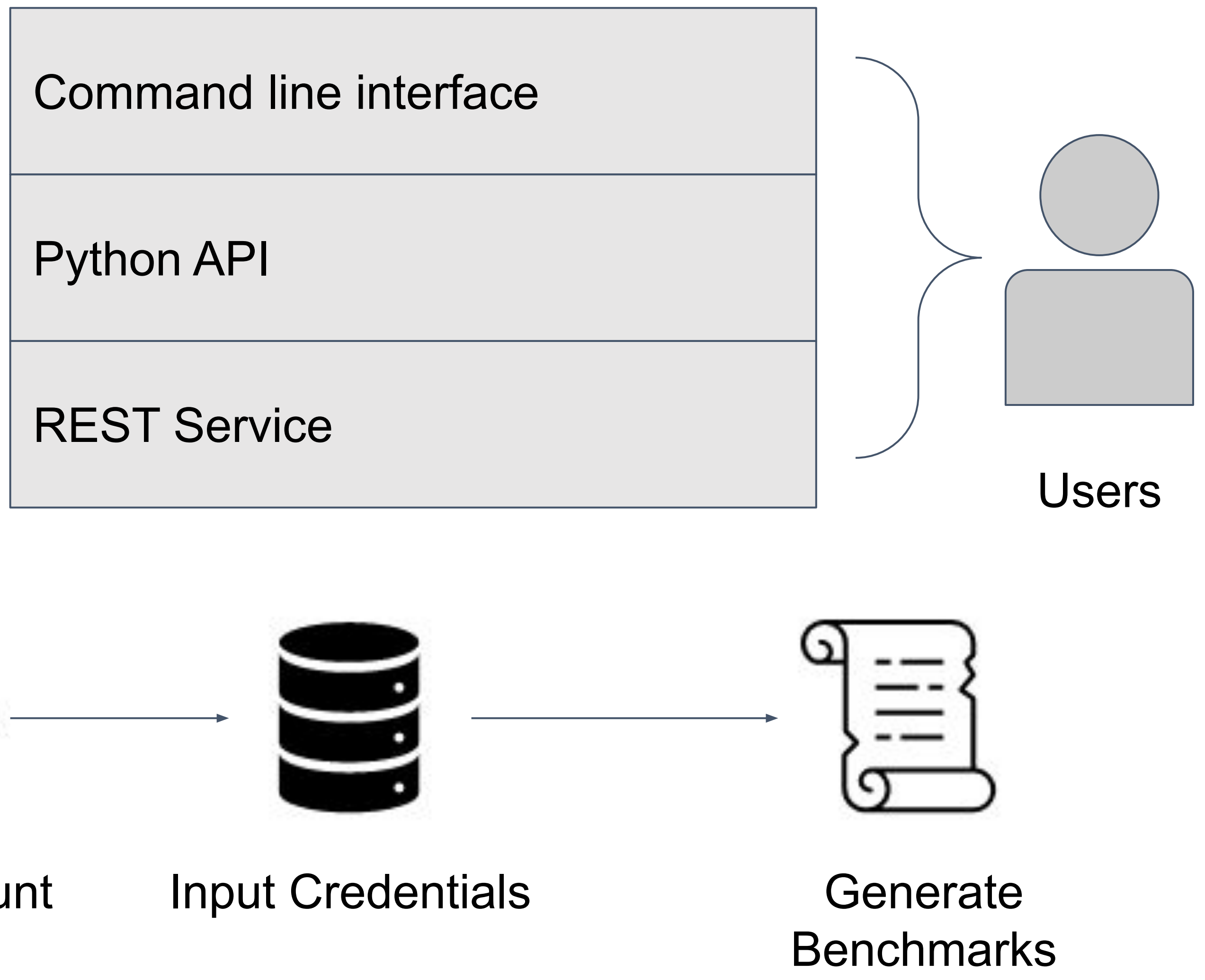
4.7 characters is average word length in english language

Provider	Free Tier Limit	Paid Tier (1 mil characters a day)	Paid Tier (1 mil words a day)
AWS Translate	<= 2 mil characters per month (for 12 months)	$15.00 USD	$70.50 USD (~26K per year)
Azure	<= 2 mil characters per month	$10.00 USD	$47.00 USD (~17K per year)
~~IBM Watson~~	<= 1 mil characters per month	$20.00 USD	$94.00 USD (~$34K per year)
Google	<= 500K characters per month	$20.00 USD	$94.00 USD (~$34K per year)

NLP Services

- Many NLP cloud services
- Condensed to one framework
  - cloudmesh
    - Translation service
    - Sentiment analysis
    - Wordcloud

Architecture

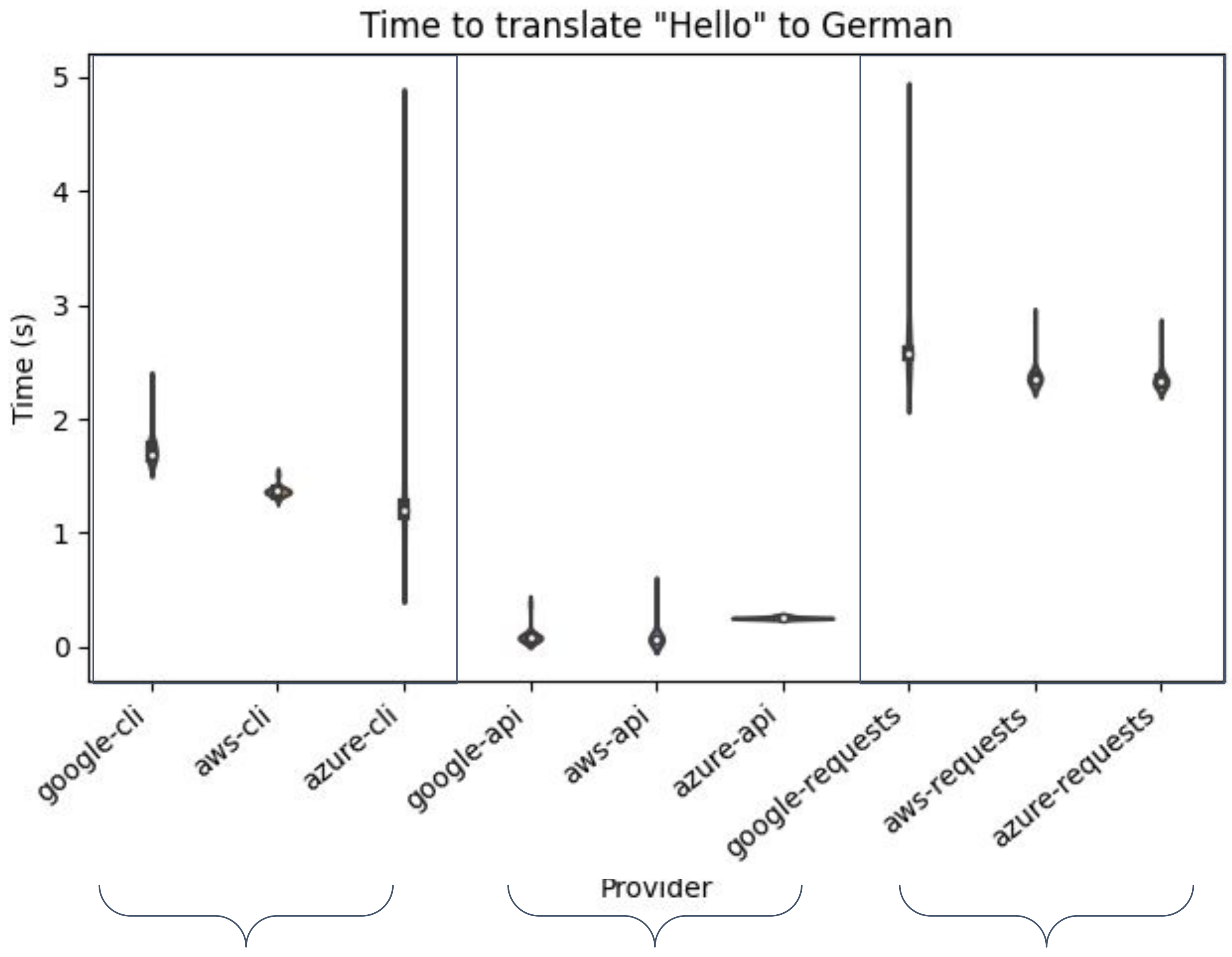


cloudmesh-nlp Interfaces

Python API (`from cloudmesh.nlp.provider import Translate`)

FastAPI Server (`requests`)

REST Service (`cms nlp`)



cloudmesh command line

cloudmesh python API

cloudmesh service

IBM Watson was suddenly deactivated without reasonable explanation, and no help support. Therefore, no benchmarks could be conducted.

This was reported also by others.

```
$ cms nlp translate --provider=google --from=en --to=de hello

$ cms nlp translate --provider=aws --from=en --to=de hello

$ cms nlp translate --provider=local --from=en --to=de hello
```

```
{'date': '05/02/2022 14:45:45',
'input': 'hello',
'input_language': 'en',
'output': 'Hallot',
'output_language': 'de',
'provider': 'aws',
'time': 0.2641}
```

Quickly evolving superpowerful-services

GPT3: https://beta.openai.com/

- large model training
- out of scope for many
- service offered for $
- continuously evolving every year

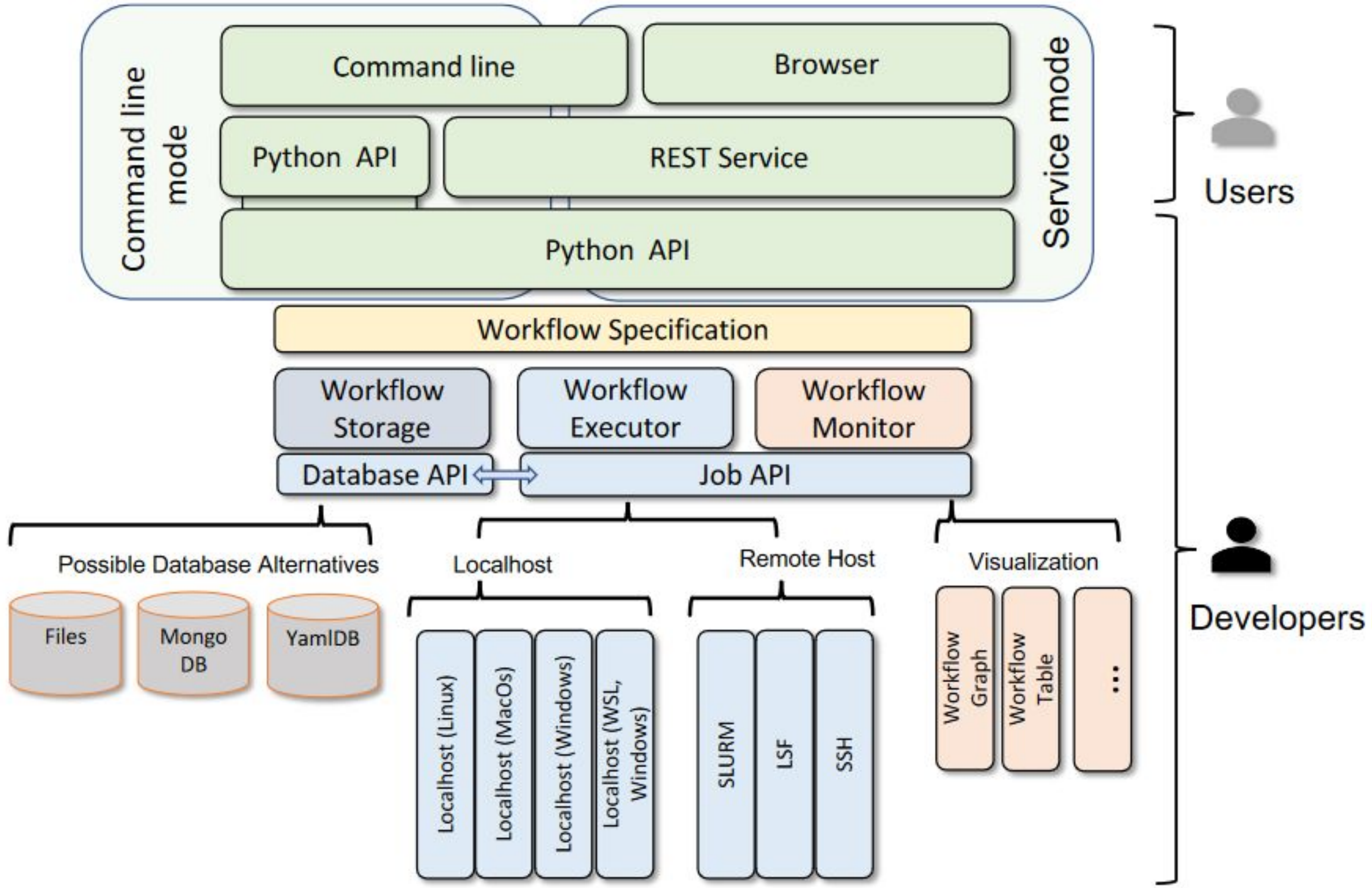
But

- dependence on services
- can a system fool others to give a believable result so others can be fooled into believing the service is useful.

Cloudmesh Compute Cluster (`cloudmesh-cc`)

- Cross-platform software for running workflows
- Workflows consist of jobs
  - SSH Job
  - SLURM Job
  - Local Job
- Supports shell, python, and jupyter jobs
- Workflows are defined by YAML configuration files

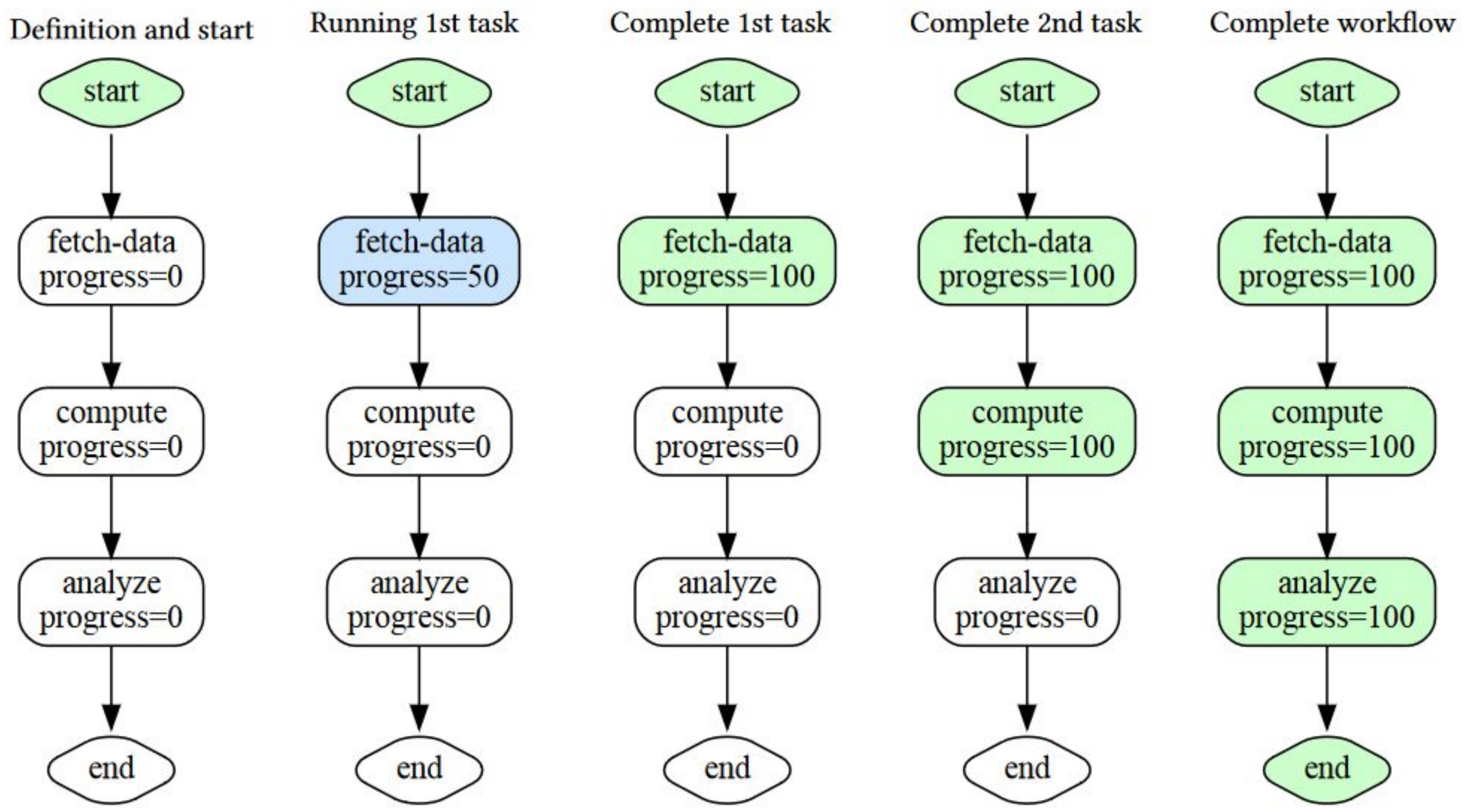
Architecture



Example of YAML workflow config file

```yaml
workflow:
    nodes:
        start:
            name: start
        fetch-data:
            name: fetch-data
            user: gregor
            host: localhost
            kind: local
            status: ready
            label: ‘{name}\nprogress={progress}’
            script: test-fetch-data.sh
        compute:
            ...
        analyze:
            ...
        end:
            ...
```

Example of workflow



Remote workflow

- Cloudmesh-cc can run on remote HPC environments
    - Rivanna, supercomputer at UVA
        - Rivanna has 603 nodes
        - 20476 cpu cores
- Workflows can take advantage of HPC resources for AI benchmarking

Table: Rivanna HPC

GPU	GPU devices per node	Number of nodes	Total Cores TC: Tensor Cores C: CUDA Cores
A100 40G	8	2	TC: **432** * 2 * 8 = 6912 C: **6912** * 2 * 8 = 110592
A100 80G	8	11	TC: **432** * 11 * 8 = 38016 C: **6912** * 11 * 8 = 608256
V100 32G	4	12	TC: **640** * 12 * 4 = 30720 C: **5120** * 12 * 4 = 245760
P100	4	4	TC: N/A C: **3584** * 4 * 4 = 57344
K80	8	8	TC: N/A C: **4992** * 8 * 8 = 319488
RTX 2080 Ti	10	2	TC: **544** * 10 * 2 = 10880 C: **4352** * 10 * 2 = 87040

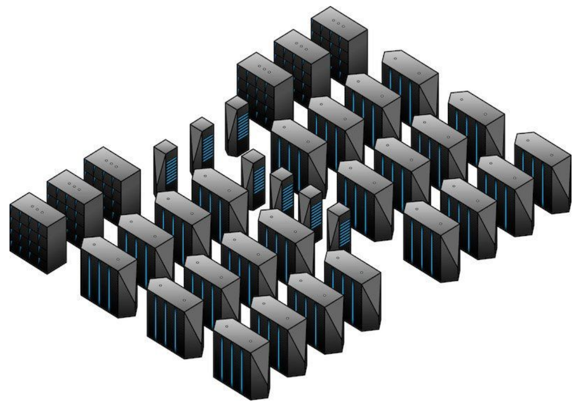
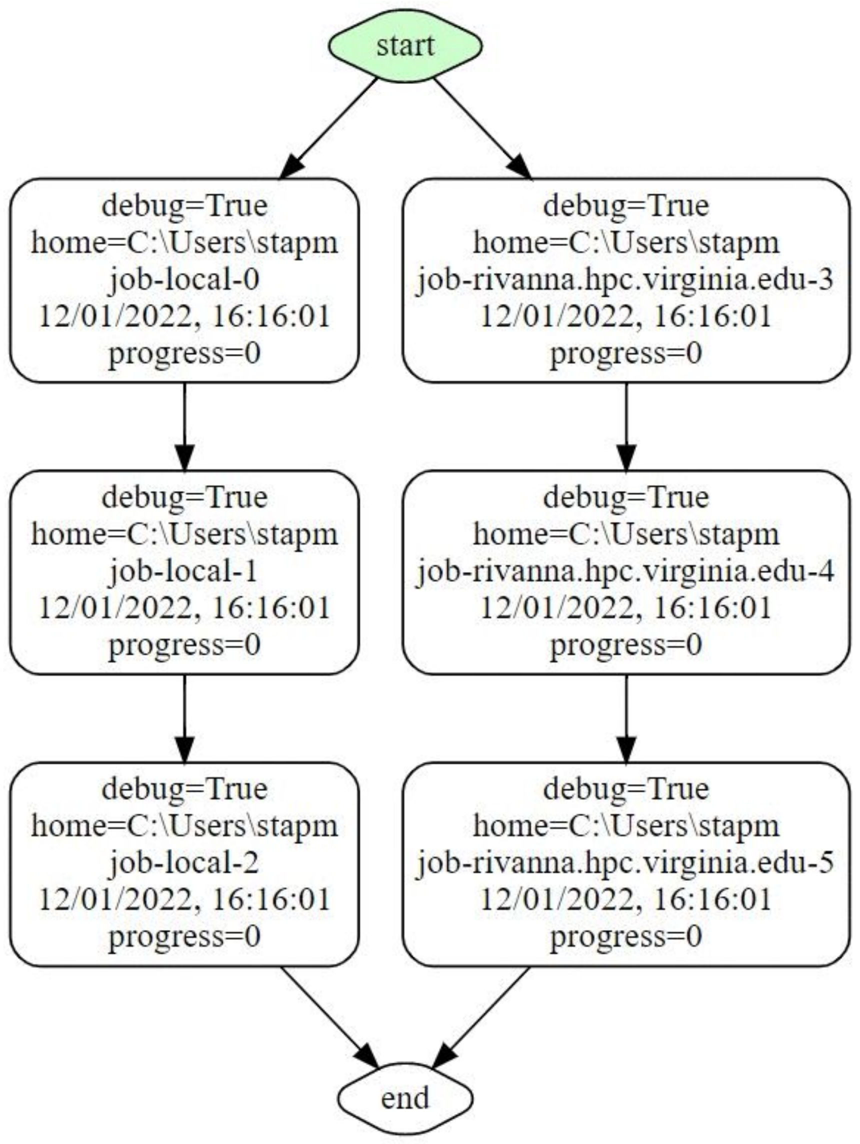


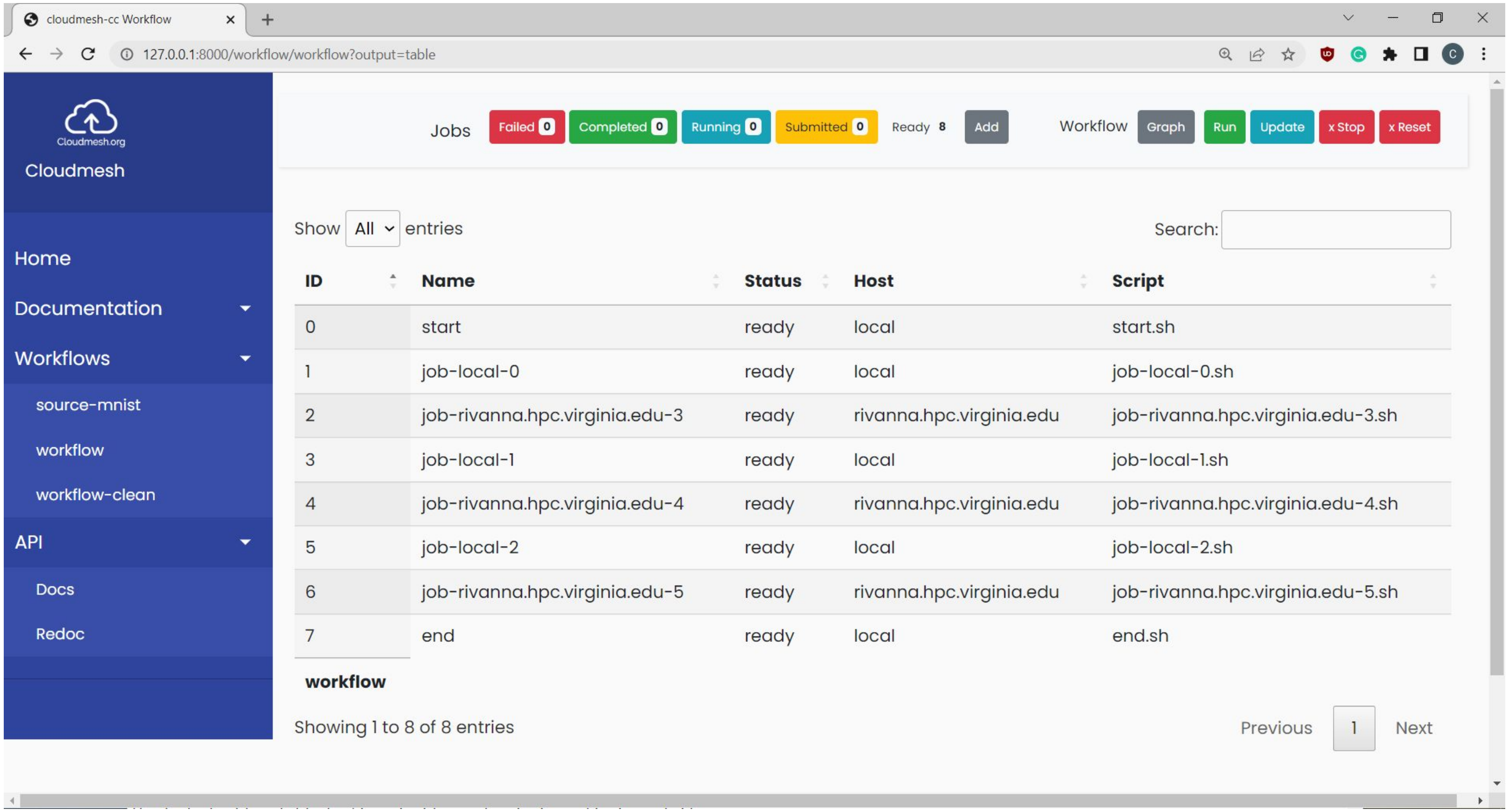
Figure: Diagram of Rivanna at UVA.

Live progression of workflow

- graphviz creates the graph
- svg file automatically updates
  - Can be seen in web and CLI interface



Workflow - Web GUI View



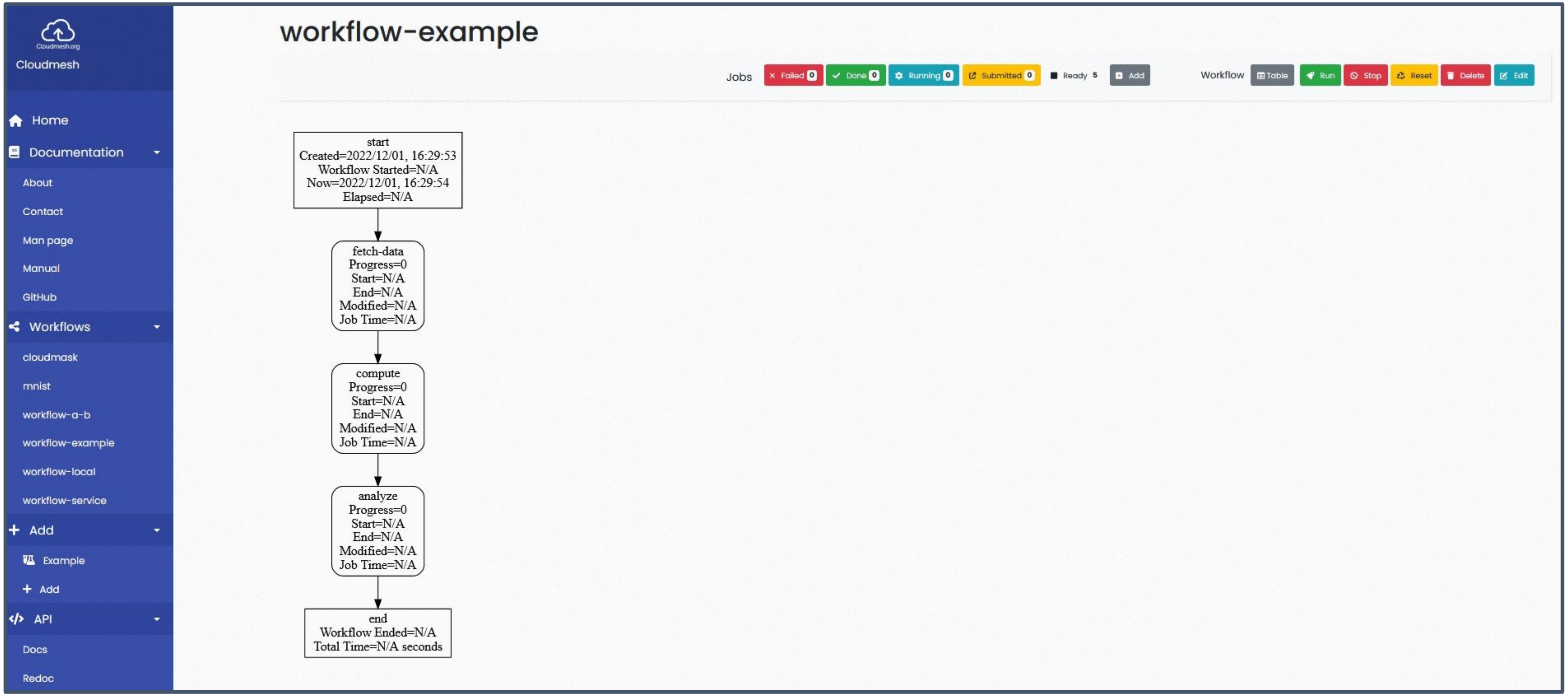
ID	Name	Status	Host	Script
0	start	ready	local	start.sh
1	job-local-0	ready	local	job-local-0.sh
2	job-rivanna.hpc.virginia.edu-3	ready	rivanna.hpc.virginia.edu	job-rivanna.hpc.virginia.edu-3.sh
3	job-local-1	ready	local	job-local-1.sh
4	job-rivanna.hpc.virginia.edu-4	ready	rivanna.hpc.virginia.edu	job-rivanna.hpc.virginia.edu-4.sh
5	job-local-2	ready	local	job-local-2.sh
6	job-rivanna.hpc.virginia.edu-5	ready	rivanna.hpc.virginia.edu	job-rivanna.hpc.virginia.edu-5.sh
7	end	ready	local	end.sh

workflow

Showing 1 to 8 of 8 entries

Web interface

Labels can be custom specified



workflow-example

```
start
Created=2022/12/01, 16:29:53
Workflow Started=N/A
Now=2022/12/01, 16:29:54
Elapsed=N/A

fetch-data
Progress=0
Start=N/A
End=N/A
Modified=N/A
Job Time=N/A

compute
Progress=0
Start=N/A
End=N/A
Modified=N/A
Job Time=N/A

analyze
Progress=0
Start=N/A
End=N/A
Modified=N/A
Job Time=N/A

end
Workflow Ended=N/A
Total Time=N/A seconds
```

Practical uses

- Artificial intelligence staging, training, and benchmarking all in one workflow

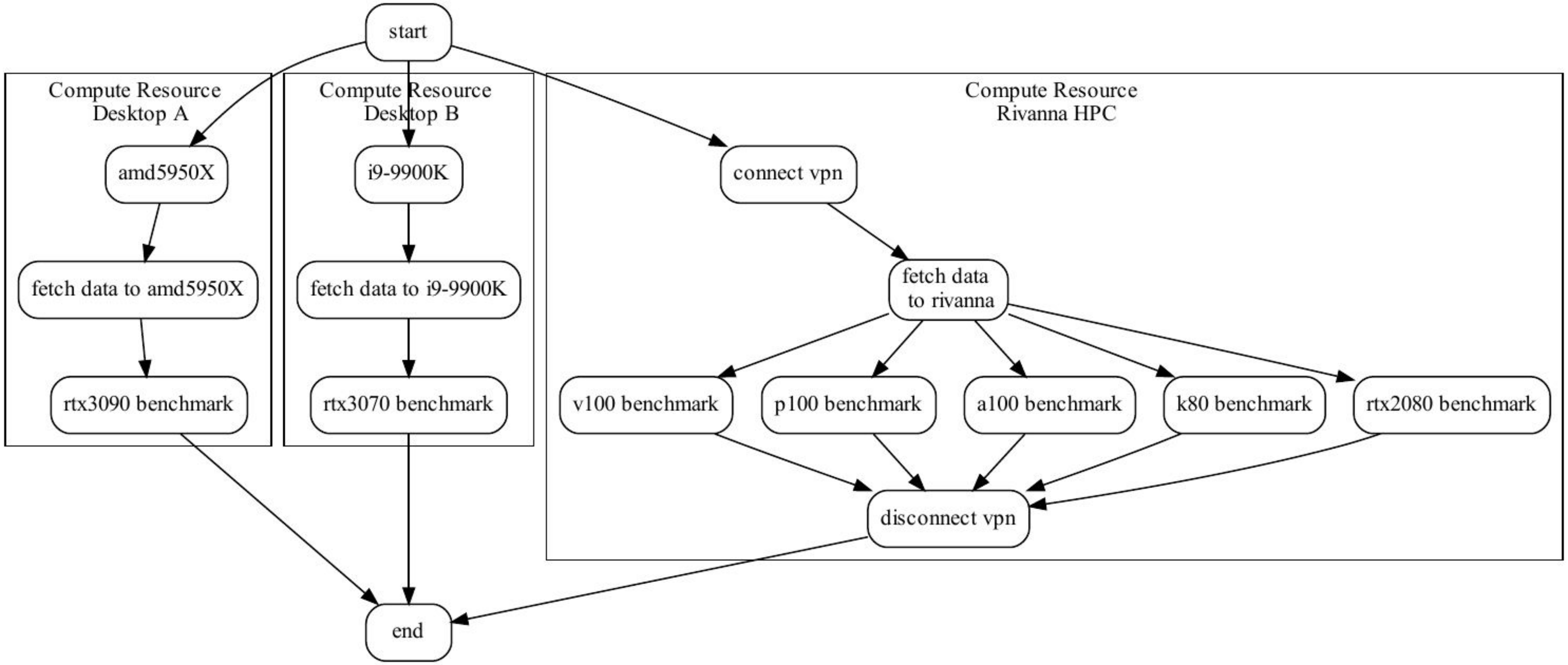


Figure: cloudmask workflow that categorizes satellite images of the sky using machine learning.

Conclusion

- We have created a hybrid cross-platform analytics system with several interfaces, including web browser GUI
- Developed codebase is minimal and small compared to other workflow programs
- Suitable for benchmarking and research projects