Globus Compute: Federated Function as a Service for Research Cyberinfrastructure

Vas Vasiliadis
vas@uchicago.edu

THE UNIVERSITY OF CHICAGO

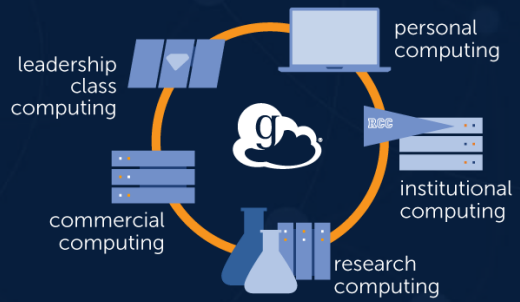
Globus in a nutshell

Managed transfer & sync

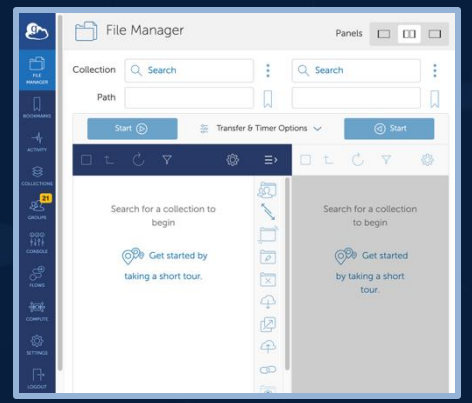
Publication & discovery

Platform-as-a-Service

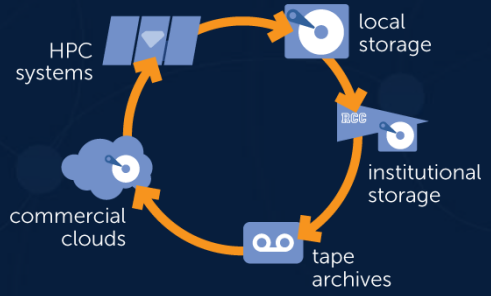
Collaborative data sharing



Managed remote execution



Software-as-a-Service

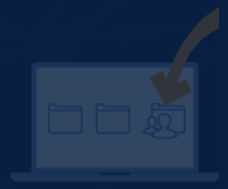


Unified data access

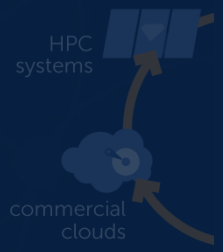
Reliable automation

Globus Compute

Managed tr



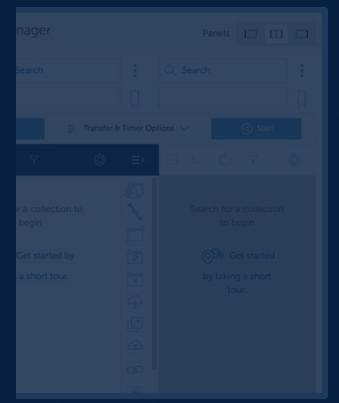
Collaborativ



Unified data access

Managed remote execution

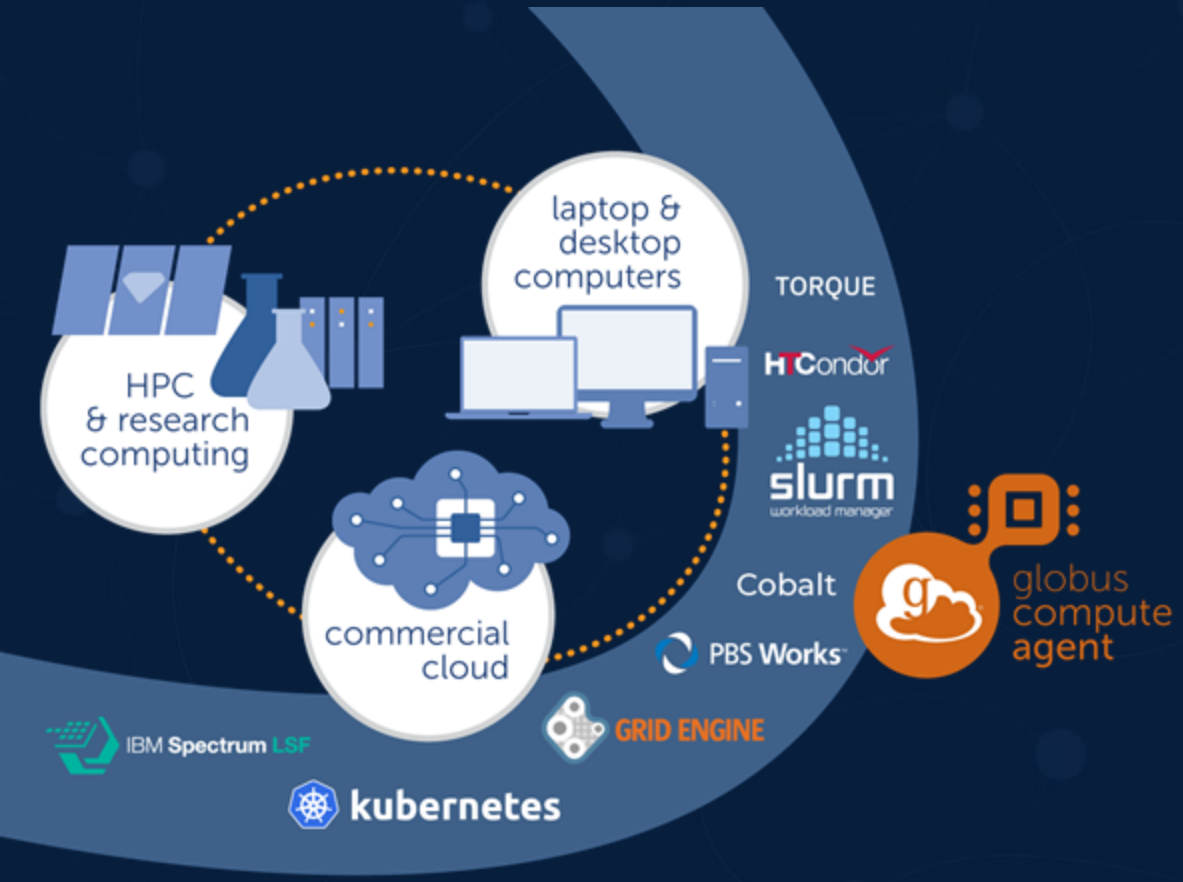
m-as-a-Service



re-as-a-Service

Reliable automation

What is Globus Compute?



- A service that enables programmatic access to compute resources using the FaaS model
  - Secure, reliable orchestration of compute tasks
  - Consistent user interface across diverse resources

How does Globus Compute benefit researchers?

- **Interact with HTC/HPC resources via familiar tools**
  - Jupyter notebooks, Python scripts
- **Reliably manage repeated tasks**
  - Parameter sweeps, Monte Carlo simulations
- **Simplify execution of jobs on diverse compute resources**
  - Scale up/out on different platforms with minimal effort
- **Create fully automated research pipelines**
  - Combine robust data management with computation
- **No need to “learn HPC”**
  - Hide complexity of underlying environment

Interactive computing via familiar tools

- **Define function like any other code**
- **Invocation runs code on compute resource**
- **Wait for results or access later asynchronously**

```python
# Define the function for remote execution
def hello_name(name: str):
    return f'Hello, {name}!'

from globus_compute_sdk import Client, Executor

MY_COMPUTE_ENDPOINT = "ad4f48be-9c03-49bc-9dc4-e240bc599bef"
compute_executor = Executor(endpoint_id=MY_COMPUTE_ENDPOINT)

# Run the function on the remote compute resource
my_result = compute_executor.submit(hello_name, 'you')

print(my_result)
```

Reliable repeated task management

```python
# Function that estimates pi
def pi(num_points):
    from random import random
    inside = 0

    for i in range(num_points):
        x, y = random(), random()
        if x**2 + y**2 < 1:
            inside += 1
    return (inside*4 / num_points)

# Register function with  Globus Compute
from globus_compute_sdk import Client

gc_client = Client()
my_function = gc_client.register_function(pi)

# Execute the function N times
N = 1000
estimates = [
    gce.submit(pi, 10**5)
    for _ in range(N)
]

# Get the results and calculate the total
results = [est.result() for est in estimates]
```

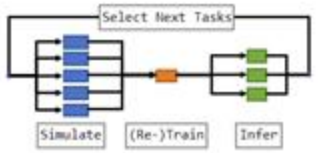
- **No need to manage process on login node**
  - Reliable outsourcing of task management
  - Automated retry on certain failures
- **No special tools required to access compute resource**

Example: AI-enabled bag of tasks

Molecular design ML-in-the-loop workflow

This notebook demonstrates an increasingly commmon ML-in-the-loop molecular design application. We use ML to guide the choice of simulations to perform. The objective of this application is to identify which molecules have the largest ionization energies (IE, the amount of energy required to remove an electron).

IE can be computed using various simulation packages (here we use xTB); however, execution of these simulations is expensive, and thus, given a finite compute budget, we must carefully select which molecules to explore. We use machine learning to predict high IE molecules based on previous computations (a process often called active learning). We iteratively retrain the machine learning model to improve the accuracy of predictions. The resulting ML-in-the-loop workflow proceeds as follows.



In this notebook, we use Globus Compute to execute functions (simulation, model training, and inference) in parallel on remote computers. We show how Globus Compute's use of (i.e., `concurrent.futures`) allows applications to be easily written that dynamically respond to the completion of asynchronous tasks.

In [ ]:
```
# Set this ID to your Globus Compute endpoint
compute_endpoint = ''
```

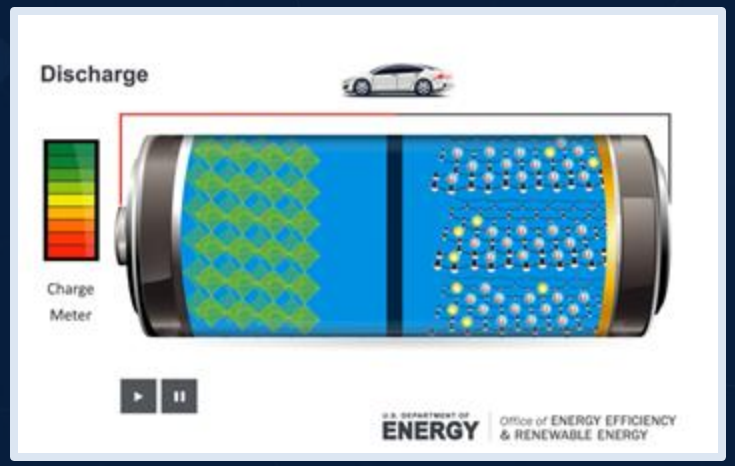
In [ ]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
from concurrent.futures import as_completed
from matplotlib import pyplot as plt
from tqdm.notebook import tqdm
from time import monotonic
```

github.com/funcx-faas/molecular-design

Aim: Identify high value molecules (high ionization energy) among a search space of billions of candidates

Problem: Simulation is expensive

Solution: Create an active learning loop, coupling simulation with ML to simulate only high value candidates



Execution on different systems

- **Simplified workload migration**
  - Register function once with Globus Compute
  - One-time setup of runtime environment on different systems
- **Same interface for function invocation**

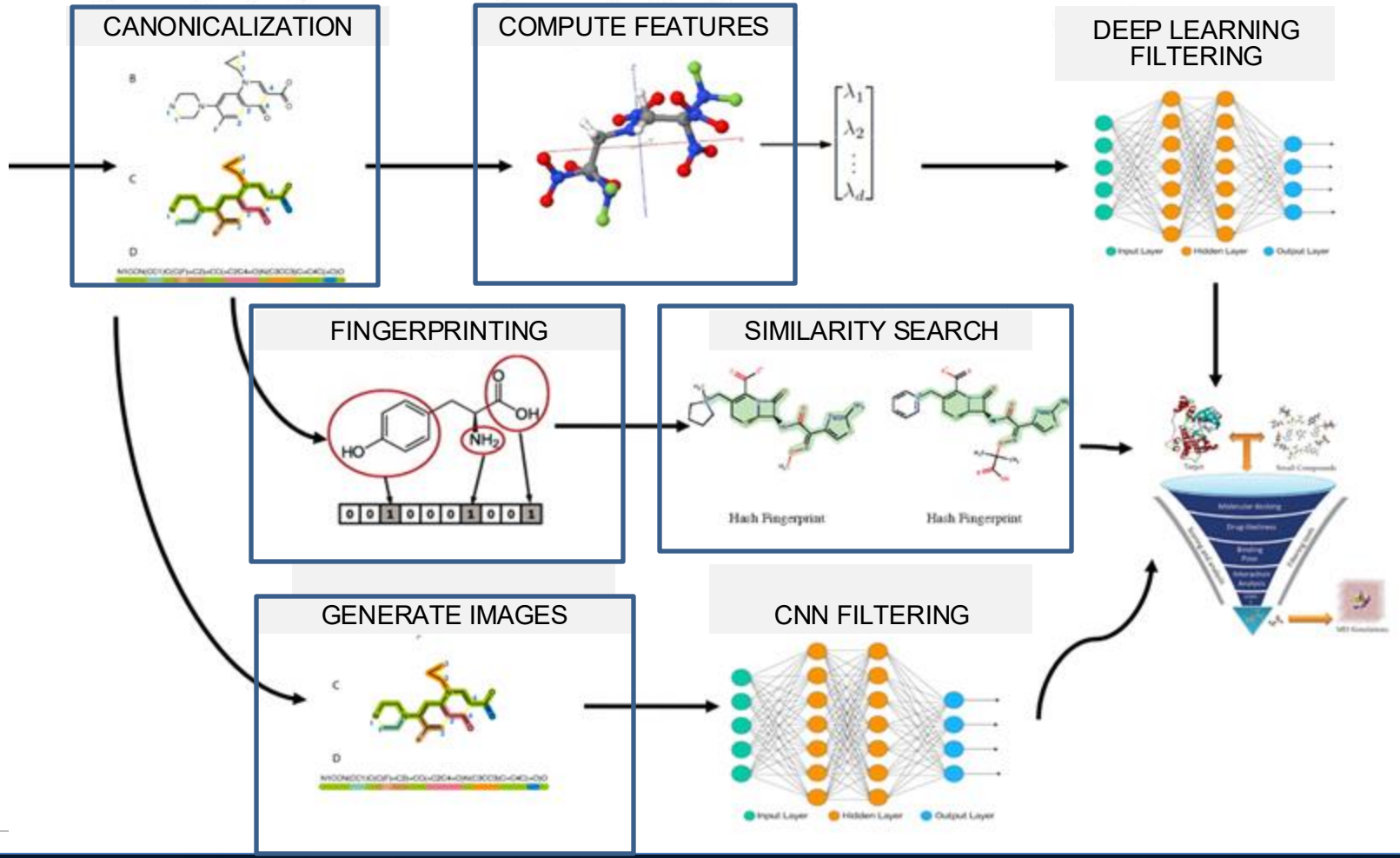
```python
def process_image(input_file):

    import os
    from PIL import Image

    image = Image.open(file)
    image.thumbnail((200, 200))
    image.save(f"{os.path.basename(file)}")

# Register the function with the Globus Compute service
from globus_compute_sdk import Client

gc_client = Client()
my_function = gc_client.register_function(process_images)

campus_cluster = "ad4f48be-9c03-49bc-9dc4-e240bc599bef"
purdue_anvil_access = "e93b4289-35c1-4de1-838a-0c0512cdf61e"

# Run code on Campus Cluster
my_campus_task = gc_client.run('image.png', my_function, campus_cluster)

# Run code on Anvil supercomputer at Purdue
my_anvil_task = gc_client.run('image.png', my_function, purdue_anvil_access)
```

Application: Using AI and supercomputers to accelerate drug development

CHEMICAL LIBRARY DATABASE

4B known molecules

Enamine
DRUGBANK
BindingDB GDB
eMolecules
cureFFI MOSES
ZINC15
LINCS
SureChEMBL
PubChem
AND MORE

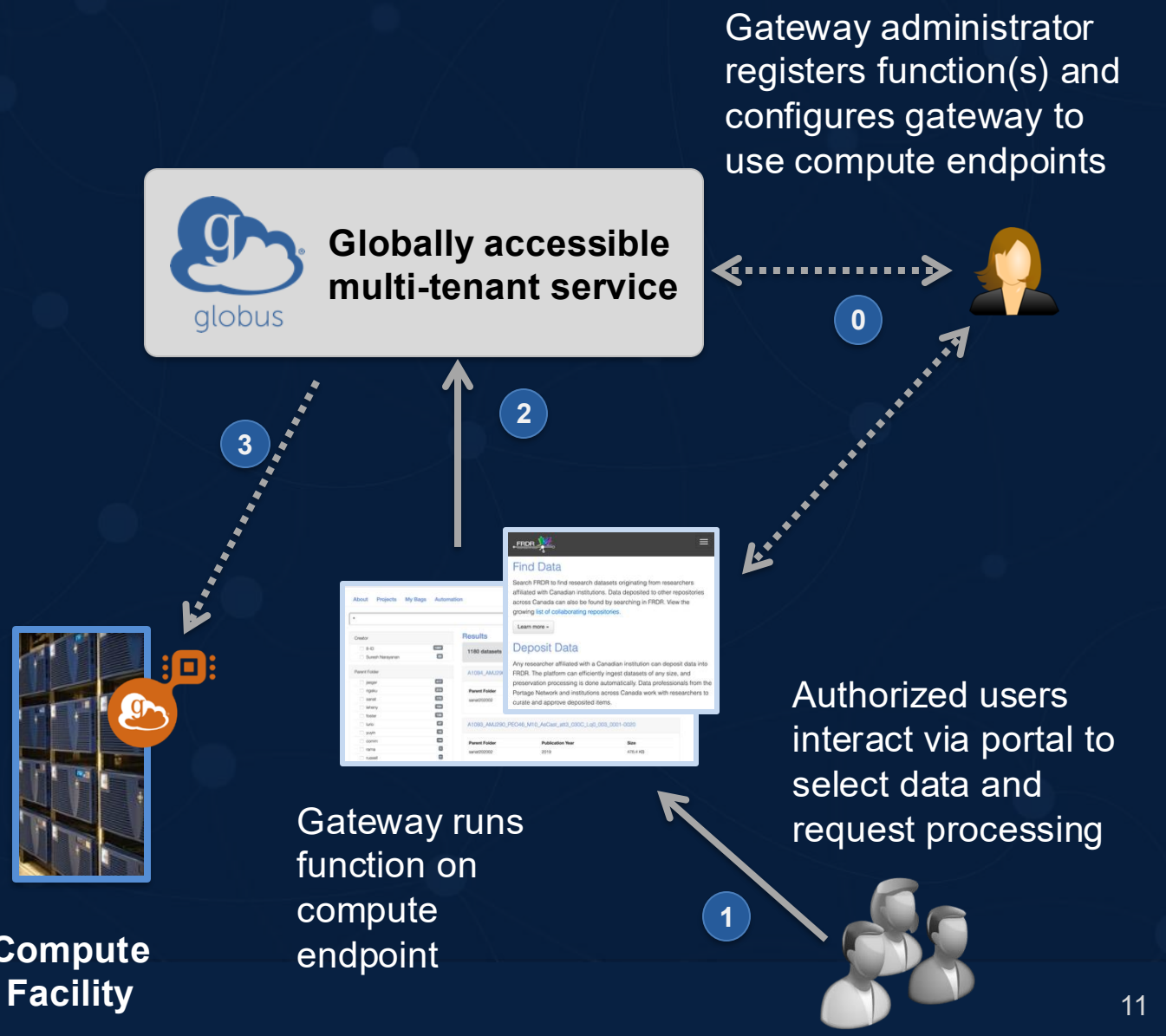


COMPUTING RESOURCES

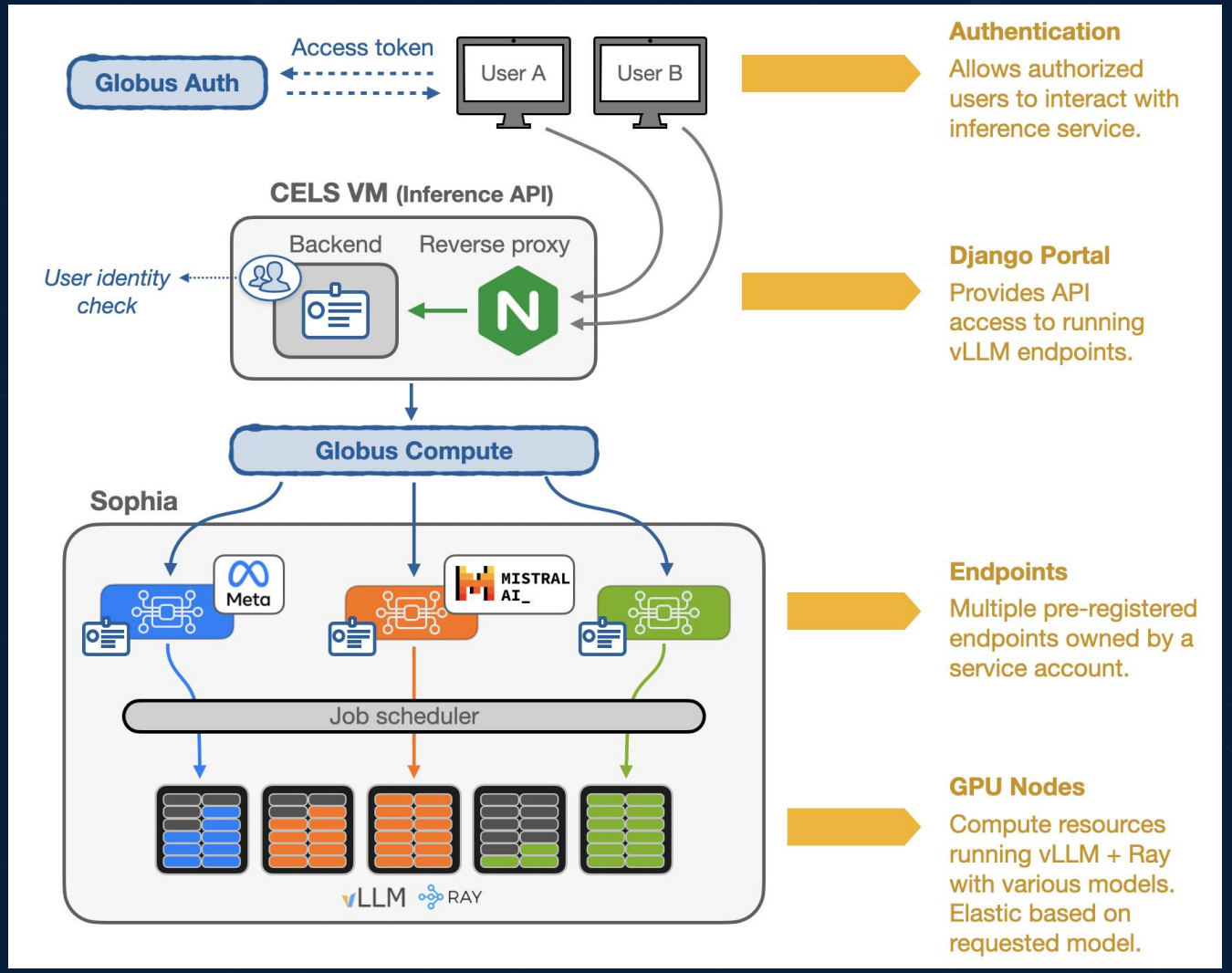
Bahuji et al. Targeting SARS-CoV2 with AI- and HPC-enabled Lead Generation: A First Data Release.

Compute access from Science Gateways/Portals

- **Secure access to execution environment from science gateways/portals**
- **Support for community account or service account model**
- **Authentication & authorization policies**
  - Allowed users
  - Permitted functions

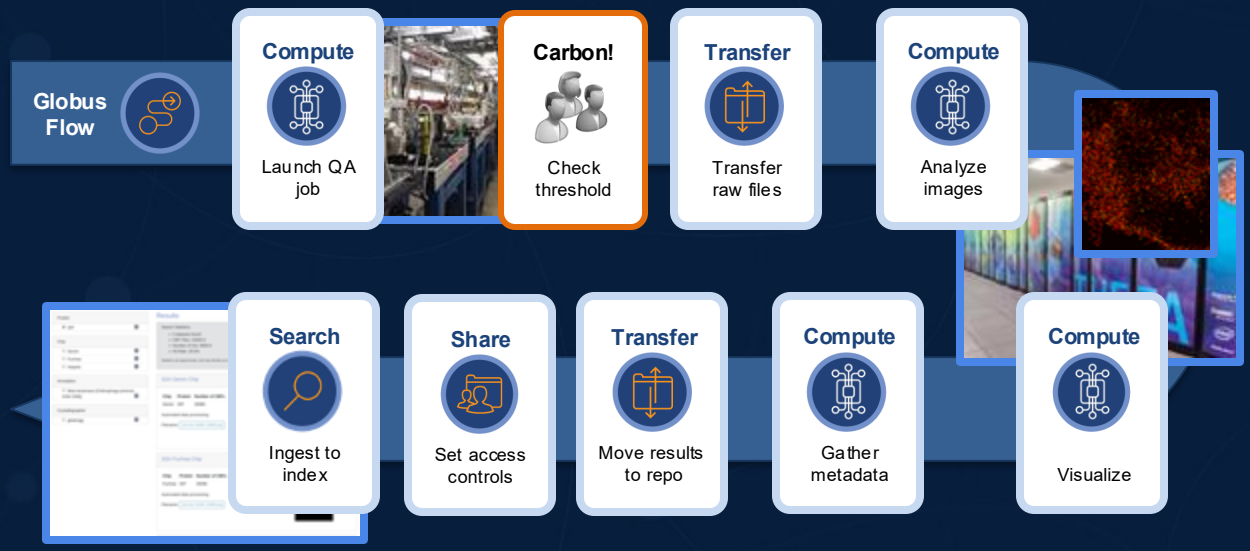


Argonne Inference Service



Enabling (and expanding) automated pipelines

- **Incorporate compute tasks into Globus Flows**
- **Perform actions that don't have an action provider**
- **Automate execution of different workloads on suitable compute resources**



Pipeline with diverse computing tasks (serial crystallography)

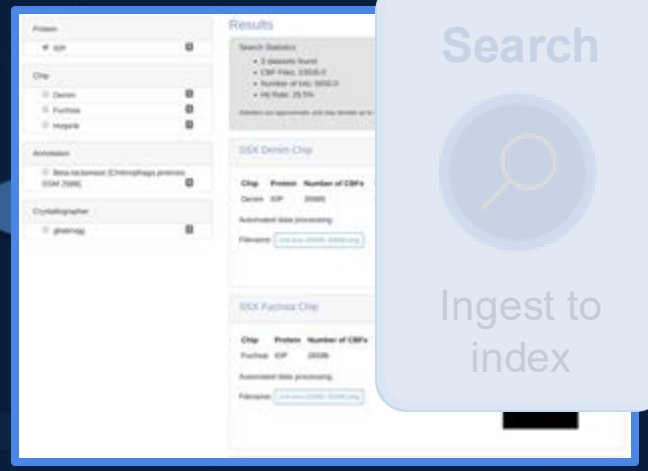
Globus Flow

Compute
Run code on **adjacent server**
Launch QA job

Carbon!
Check threshold

Transfer
Transfer raw files

Compute
Run code on **supercomputer**
Analyze images



Search
Ingest to index

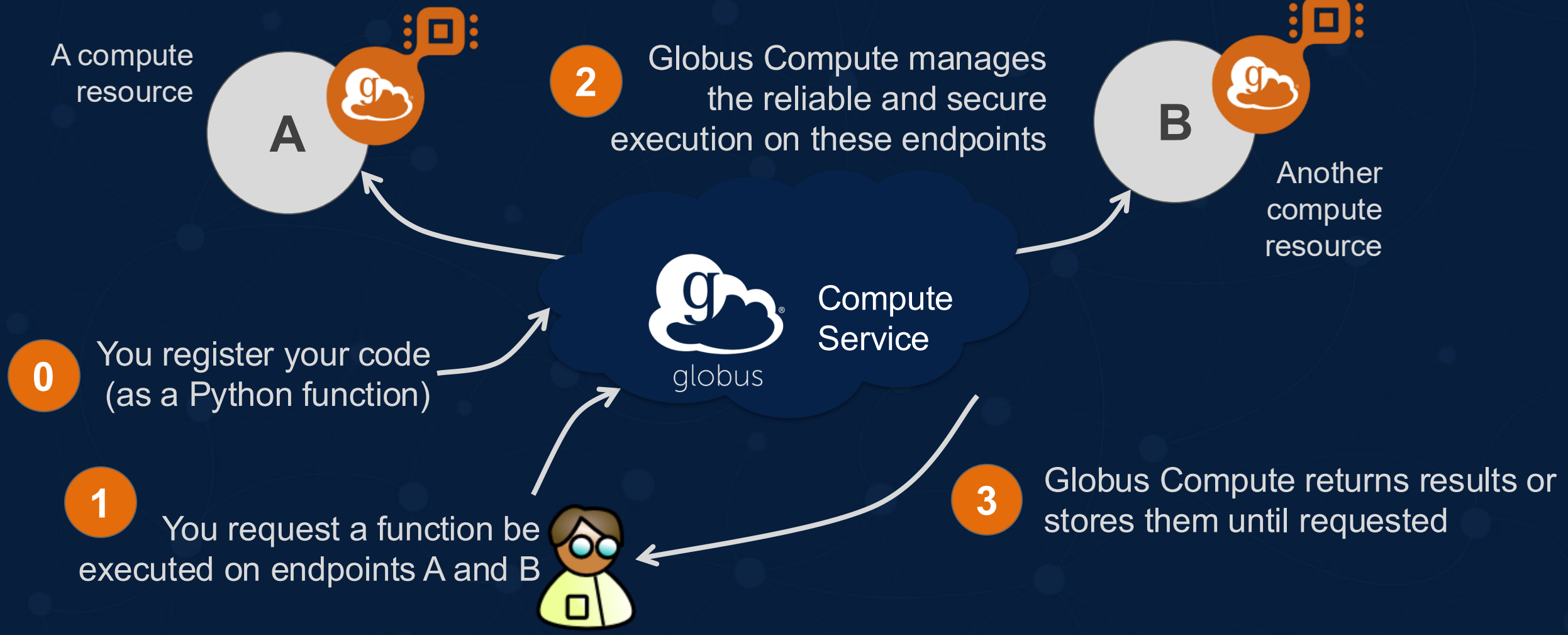
Share
Set access controls

Transfer
Move results to repo

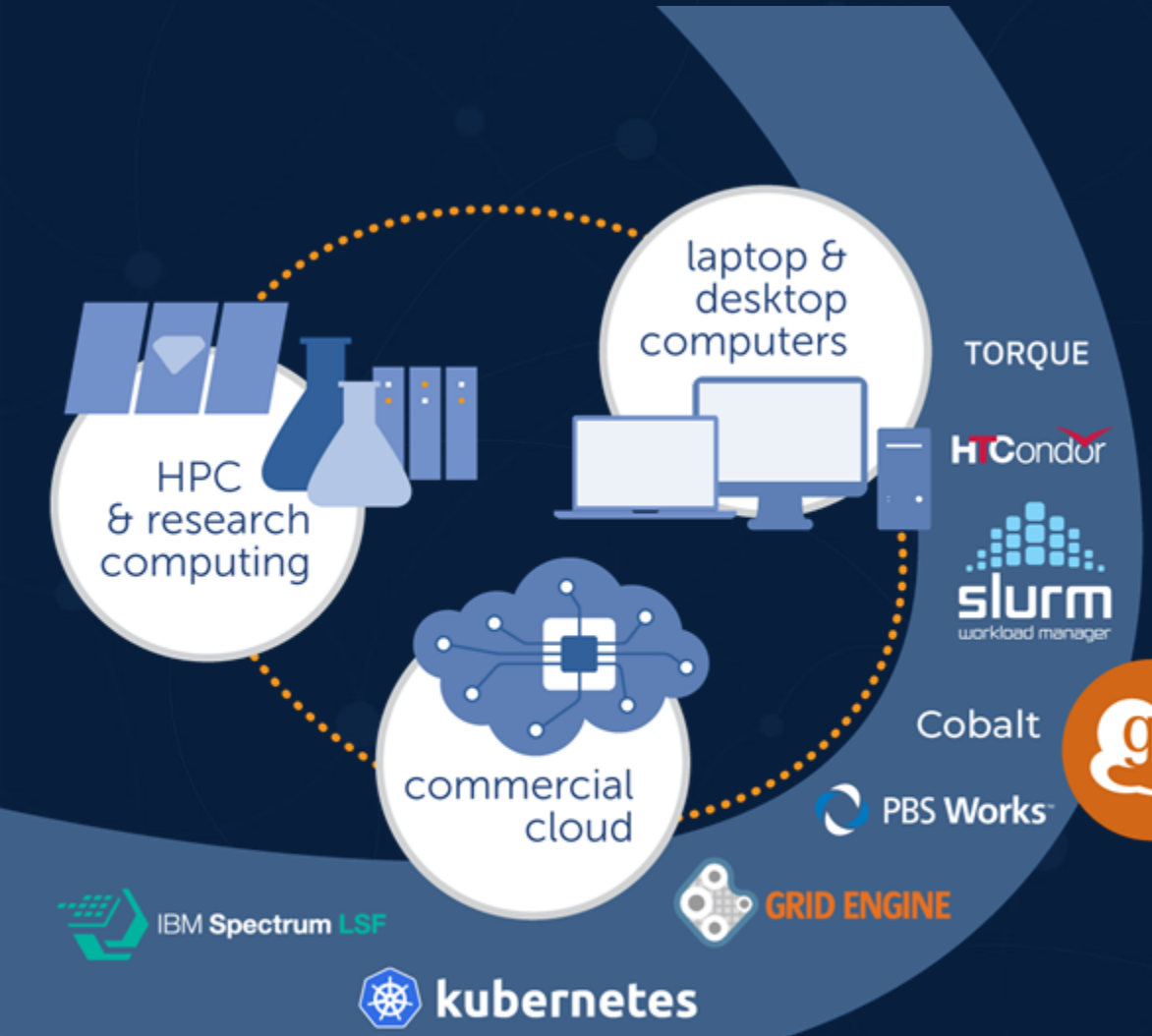
Compute
Access **custom metadata extraction service**
Gather metadata

Compute
Run code on **GPU cluster**
Visualize

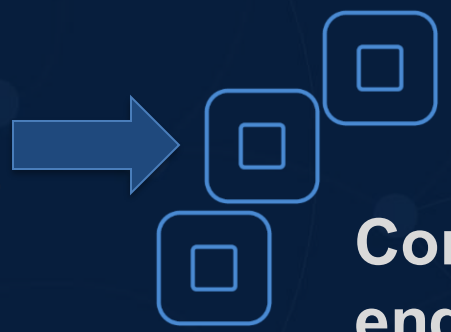
From a researcher's PoV…



Enabling compute at your institution



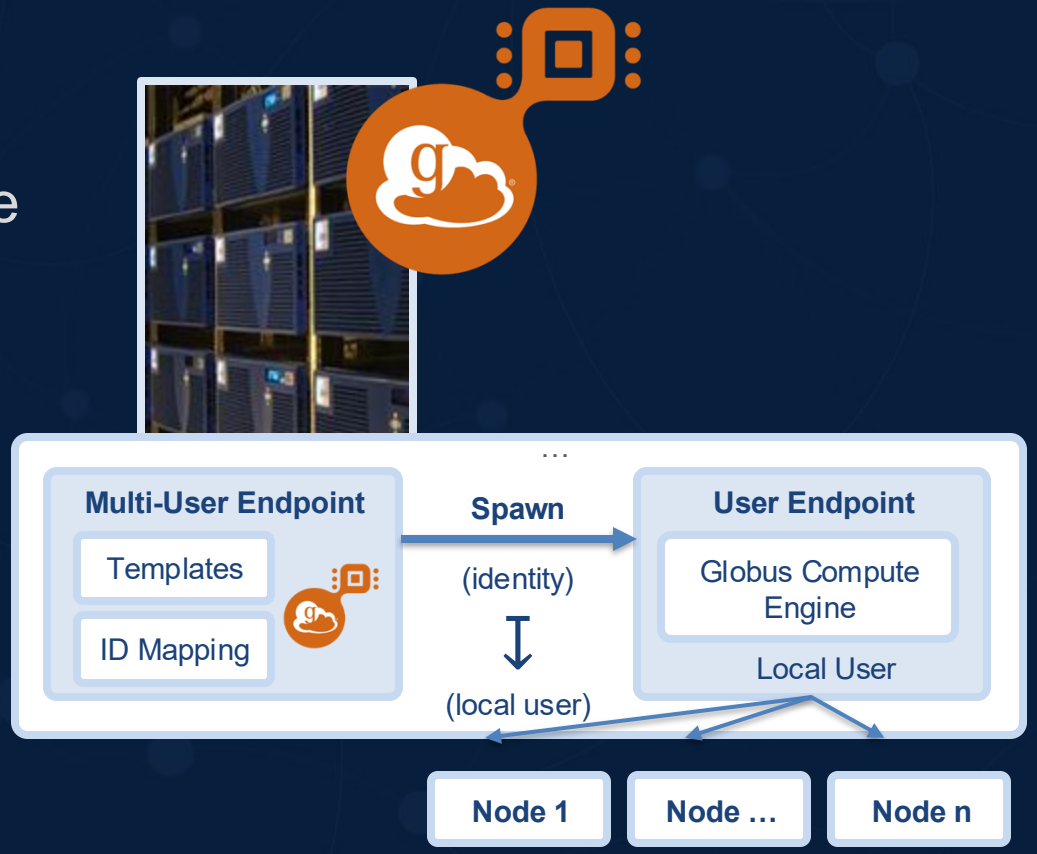
Deploy Globus Compute multi-user agent to connect to ecosystem



Compute endpoint

From a system administrator's PoV…

- **Install compute agent using Linux package manager**
  - Campus cluster, cloud/other shared resource ➔ Multi-user endpoint
  - Facility/instrument-adjacent cluster ➔ Single-user endpoint
- **Define authentication/authorization policies**
  - User identity mapping for authentication behaves like Globus Connect
- **Configure executor template (Slurm, PBS, LSF, Cobalt, K8s, *et al*)**
  - Fixed values, defaults for common job parameters
  - Parameters that may be overridden by user



Resources

- **Getting started:**

jupyter.demo.globus.org → Compute_Introduction.ipynb

globus-compute.readthedocs.io/en/stable/quickstart.html

- **Endpoint configuration and sample templates:**

globus-compute.readthedocs.io/en/stable/endpoints/multi_user.html

- **Our amazing support team: support@globus.org**

Ask us for help!

- **Guidance on best practices**
- **Sounding board for your design/implementation**
- **Assistance with configuring endpoint, templates, etc.**
- **All at no cost to you …just reach out**