# ALCF Inference Service

Misha Salim

# Inference Service

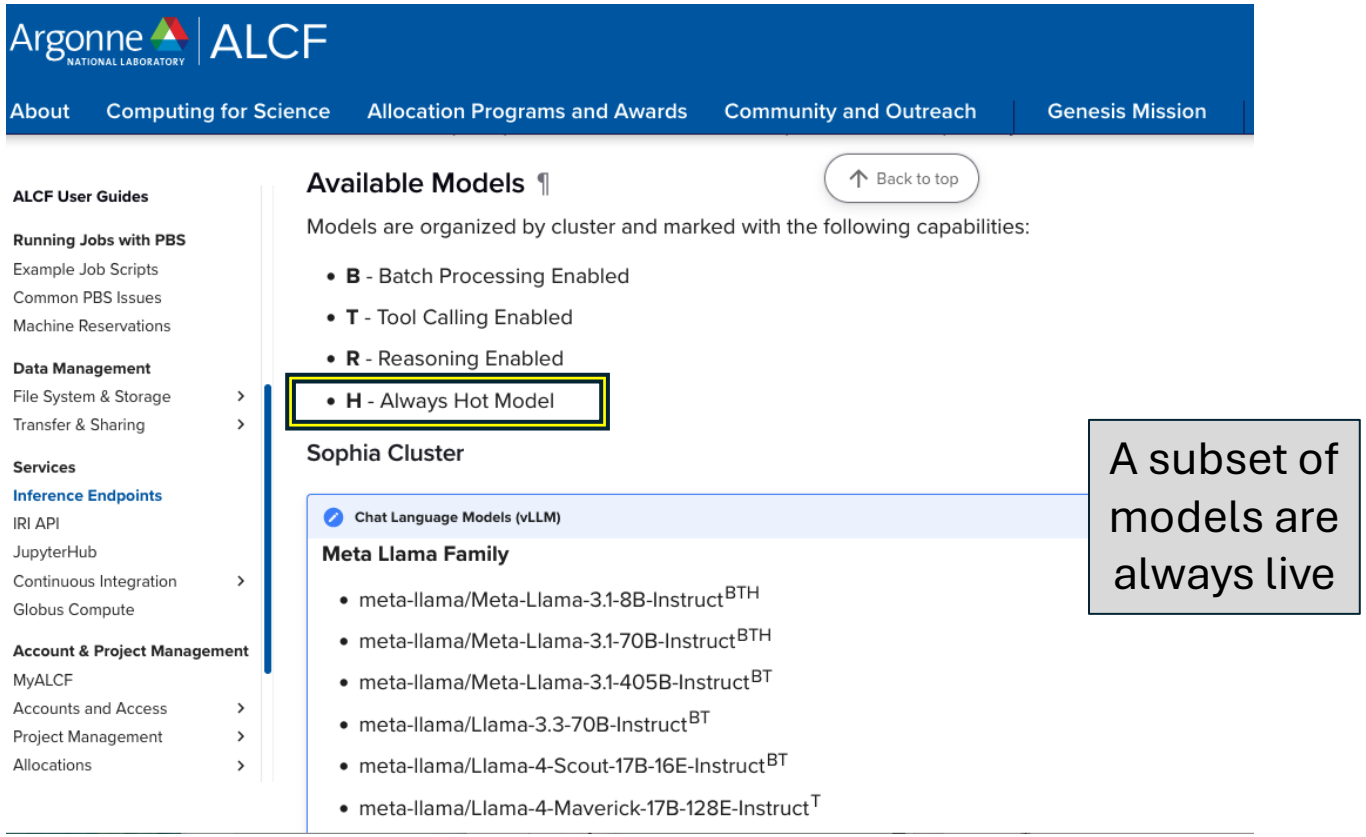
- Always-on, internet-accessible open weight LLMs and science foundation models running @ALCF

- OpenAI- and Anthropic-compatible APIs provide a foundation for agentic workflows and enable researchers to apply AI to their data, simulations, and real-time experiments.

- Shared service means model lifetime is not tied to your own workloads:  just point at https://inference-api.alcf.anl.gov/ and go

# Model Catalog

- ALCF is continually adding newly-released models
- https://docs.alcf.anl.gov/services/inference-endpoints



# Model Catalog

- ALCF is continually adding newly-released models
- https://docs.alcf.anl.gov/services/inference-endpoints


Gemma 4




# Model Catalog

## General-Purpose LLMs

- **Llama (Meta)** — 3.1-8B / 70B / 405B-Instruct, 3.3-70B-Instruct, Llama-4 Scout-17B-16E & Maverick-17B-128E
- **Gemma (Google)** — gemma-3-27b-it; gemma-4 26B-A4B / 31B / E4B-it
- **Mistral** — Large-2407, Large-3-675B-2512, Mixtral-8x22B-v0.1, Devstral-2-123B-2512
- **Nemotron-3 (NVIDIA)** — Super-120B, Ultra
- **gpt-oss (OpenAI)** — 20B, 120B
- **Trinity-Large-Thinking (Arcee)** — W4A16 quantized

## Embedding Models

- embeddinggemma-300m (Google)
- SFR-Embedding-Mistral (Salesforce)
- Mistral-7B-Instruct-v0.3-embed

## Vision & Multimodal

- Llama-3.2-90B-Vision-Instruct — multimodal VLM
- SAM 3 — promptable image segmentation
- DINOv3 — SoTA visual representation

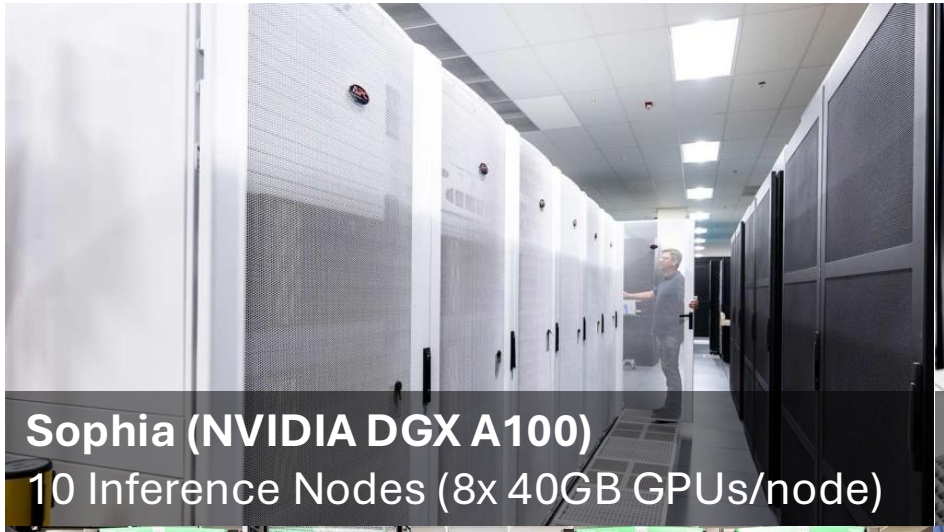
## Scientific Models

- AuroraGPT (Argonne) — IT-v4, Tulu3-SFT, DPO-UFB, KTO-UFB checkpoints
- AstroSage-70B (AstroMLab) — astronomy
- GenSLM — genome-scale language models
- AmSC SUF Partnership — 7 HEP models, Triton backend

**LLM Endpoints**

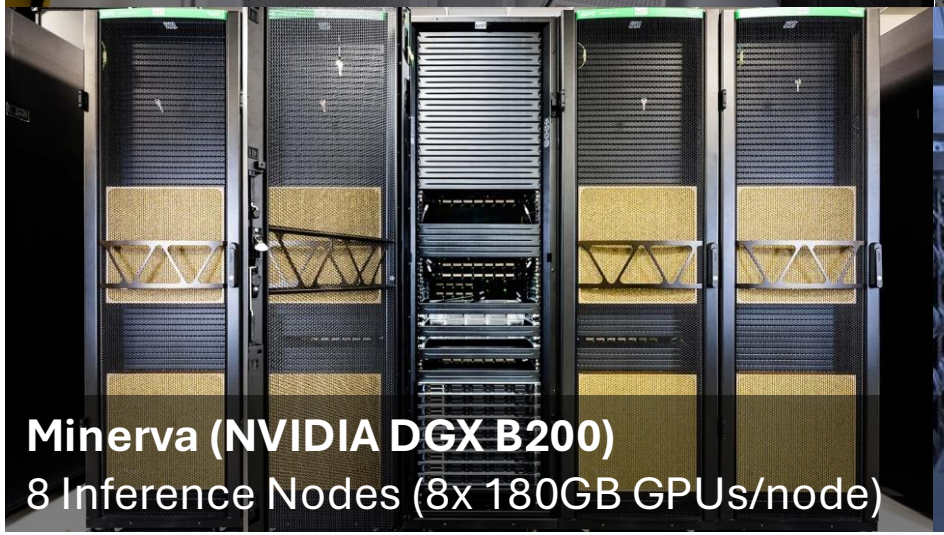
/v1/chat/completions
/v1/responses
/v1/messages
/v1/completions
/v1/embeddings

# The Hardware

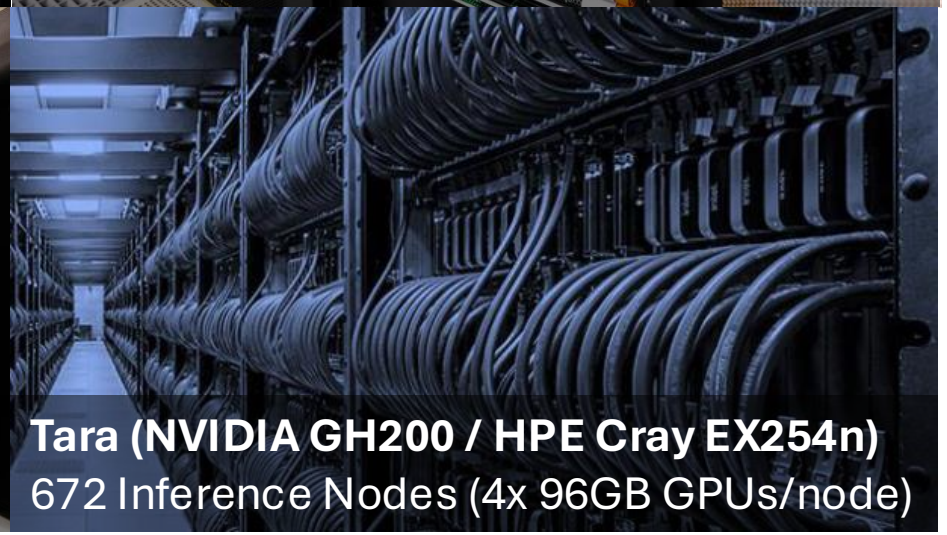


**Sophia (NVIDIA DGX A100)**
10 Inference Nodes (8x 40GB GPUs/node)


**Metis (SambaNova SN40L)**
6 Inference Nodes (16 accelerators/node)

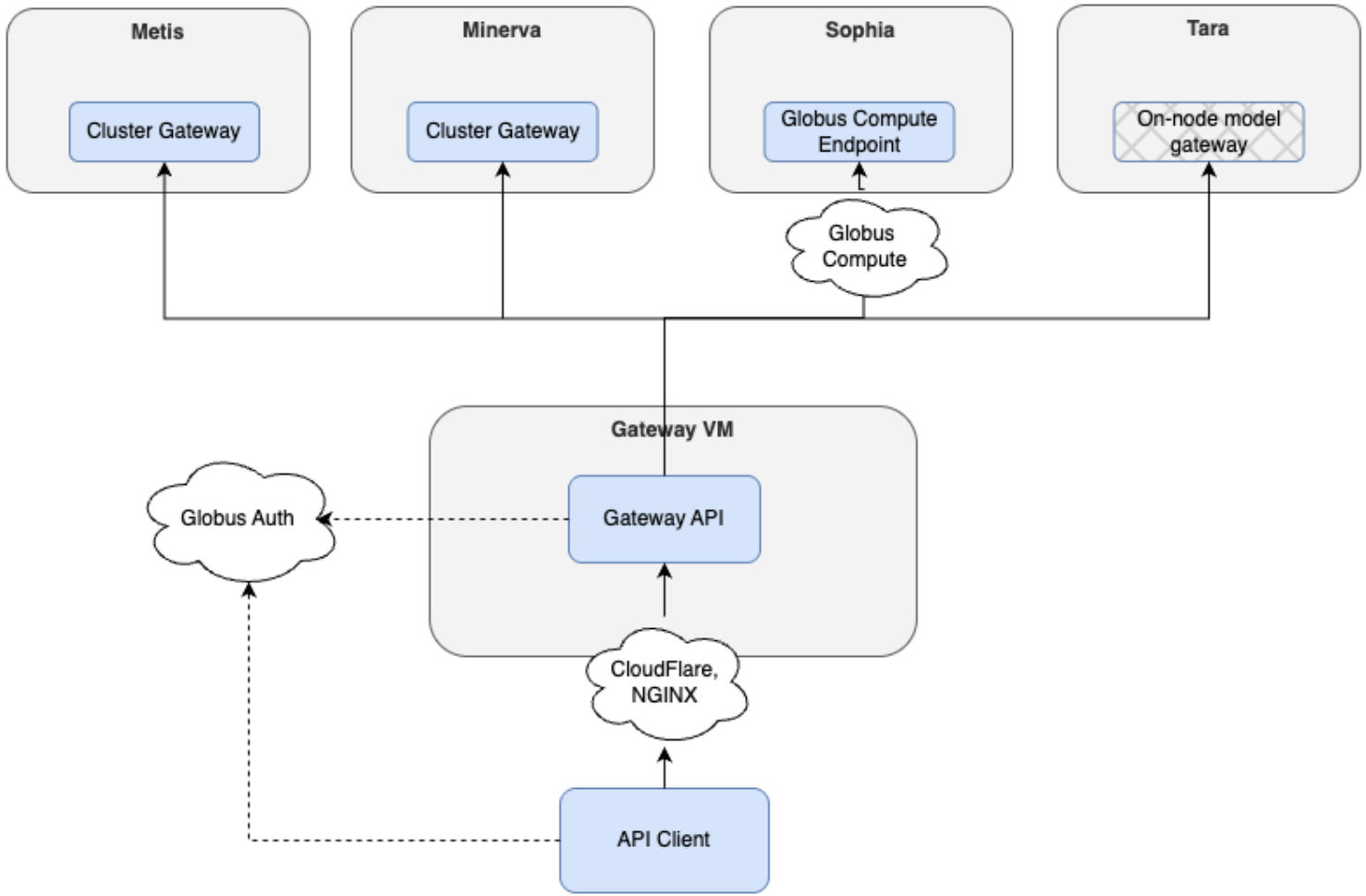


**Minerva (NVIDIA DGX B200)**
8 Inference Nodes (8x 180GB GPUs/node)



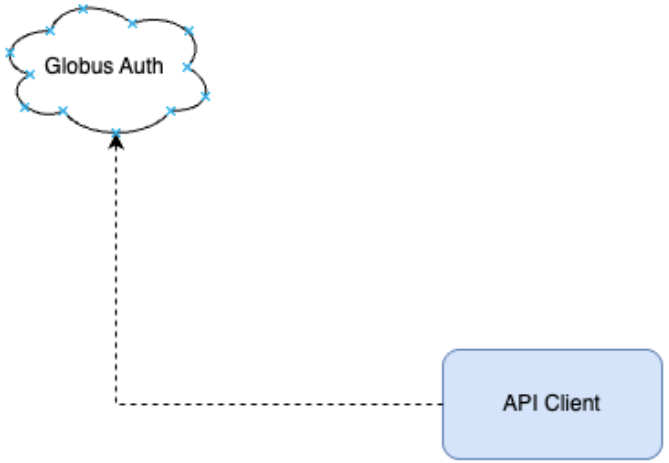
**Tara (NVIDIA GH200 / HPE Cray EX254n)**
672 Inference Nodes (4x 96GB GPUs/node)

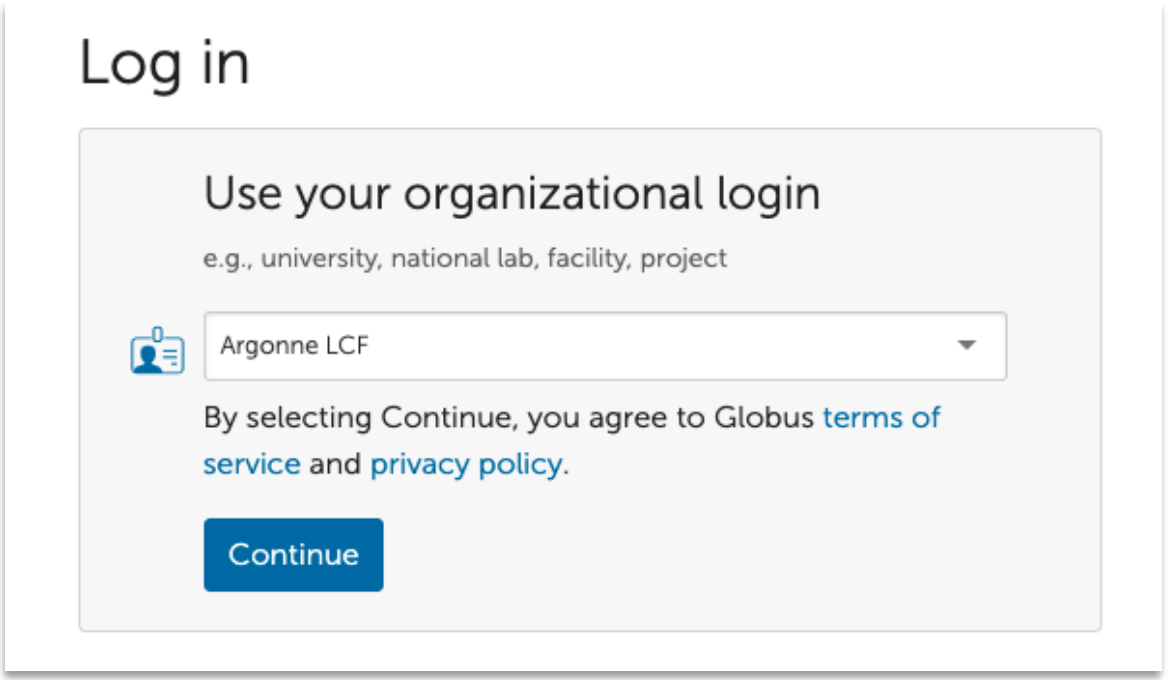
# Gateway Overview




- Globus OAuth2 flows for API Clients and Web UI
- Multi-backend integration
- Live model registry
  - Always-on
  - Autoscaling / scale-to-zero

# Globus Auth

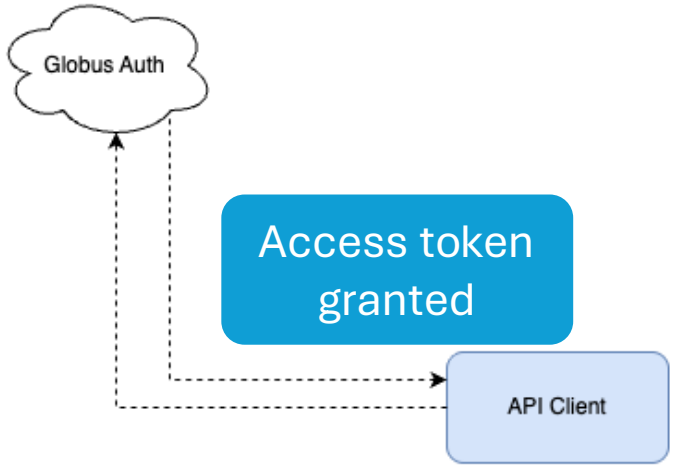



- User-initiated OAuth2 login
- Globus federates w/ 1100+ identity providers
- Currently open to all users with ANL or ALCF credentials

# Globus Auth

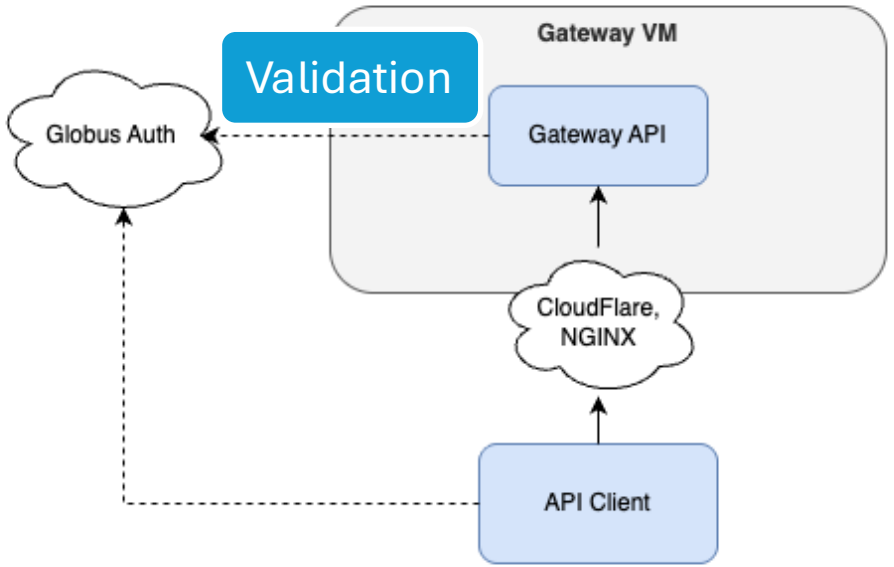
```
$ uvx alcf-ai auth login

Please authenticate with Globus here:
------------------------------------
https://auth.globus.org/v2/oauth2/authorize?client_id=5
2Fauth.globus.org%2Fv2%2Fweb%2Fauth-code&scope=urn%3Agl
Ahttps%3A%2F%2Fauth.globus.org%2Fscopes%2F96c7390b-a3e8
%2Fscopes%2F681c10cc-f684-4540-bcd7-0b4df3bc26ef%2Facti
YPaoCLdoHkn6Irlxs6xI-2i2TwPFlKip43iMTI&code_challenge_m
app+on+Mac&session_required_policies=83732ff2-9c42-4548
------------------------------------

Enter the resulting Authorization Code here:
```




- On completion of Authorization code flow, the client receives access tokens for the Inference Service

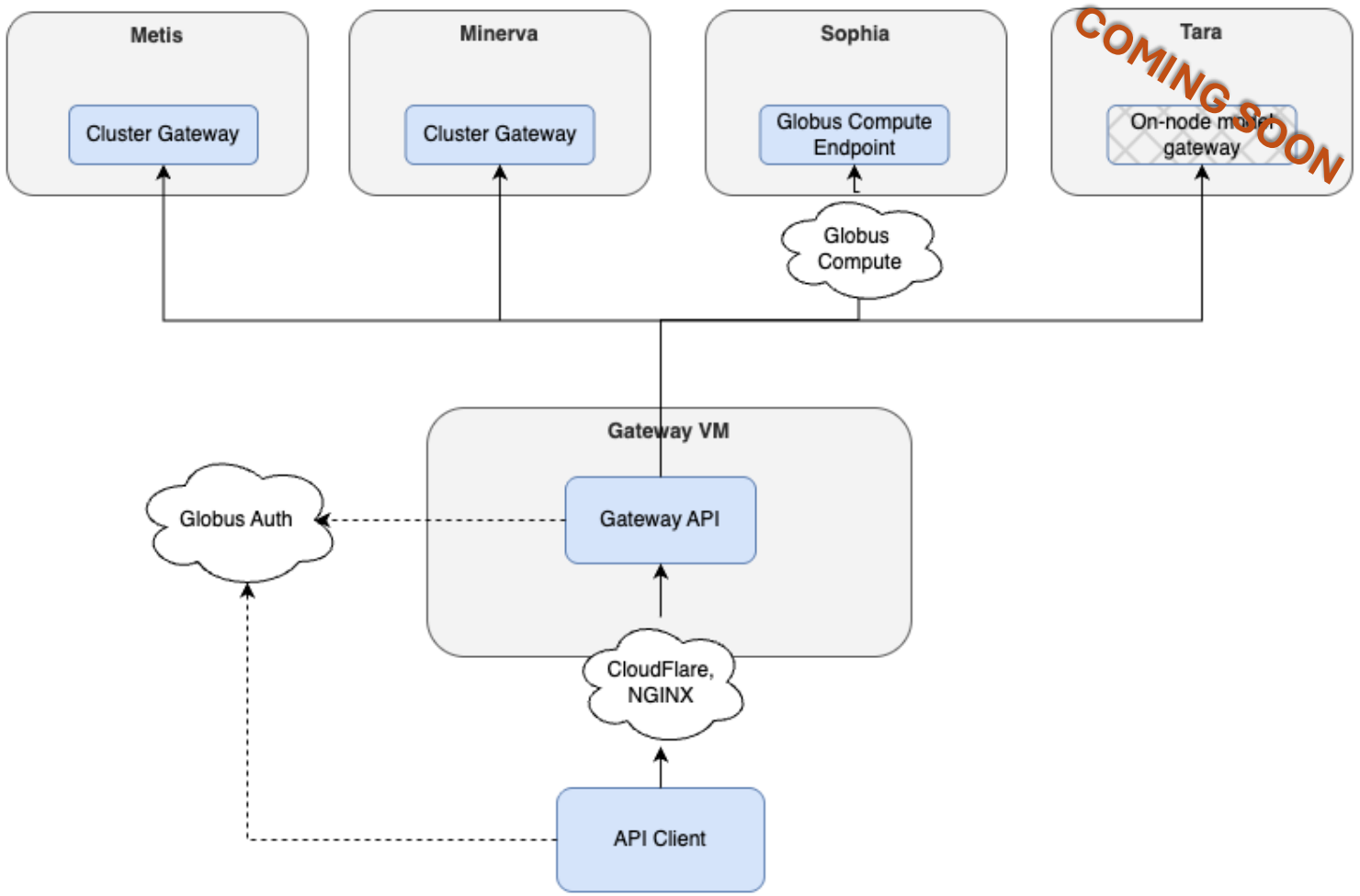
# Globus Auth

```python
from openai import OpenAI
from inference_auth_token import get_access_token

# Get your access token
access_token = get_access_token()

client = OpenAI(
    api_key=access_token,
    base_url="https://inference-api.alcf.anl.gov/resource_server/metis/api/v1"
)

response = client.chat.completions.create(
    model="gpt-oss-120b",
    messages=[{"role": "user", "content": "Explain quantum computing in simple terms."}]
)
```



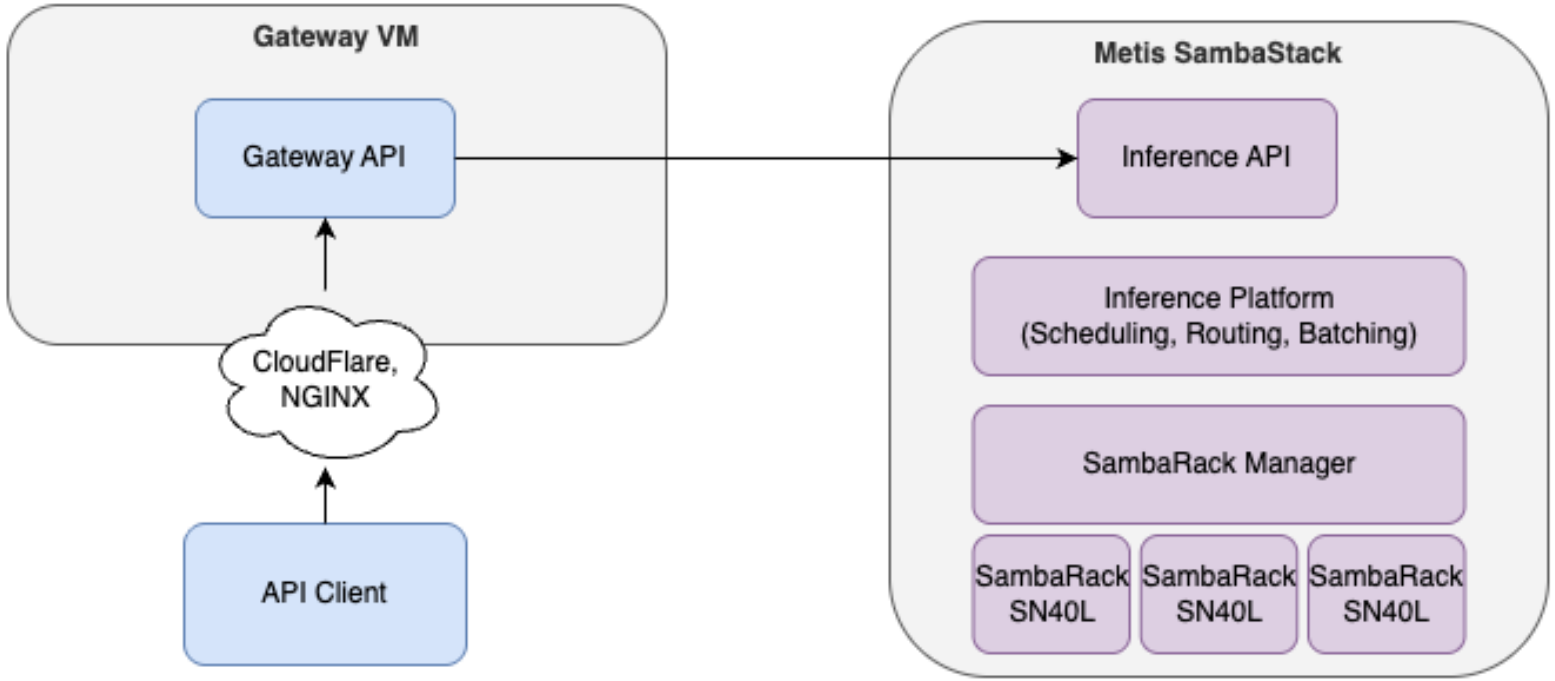

- Access tokens sent in HTTP Authorization Header on every request to Gateway API
- Gateway validates tokens and introspects user information to authorize request

# Backend Integrations

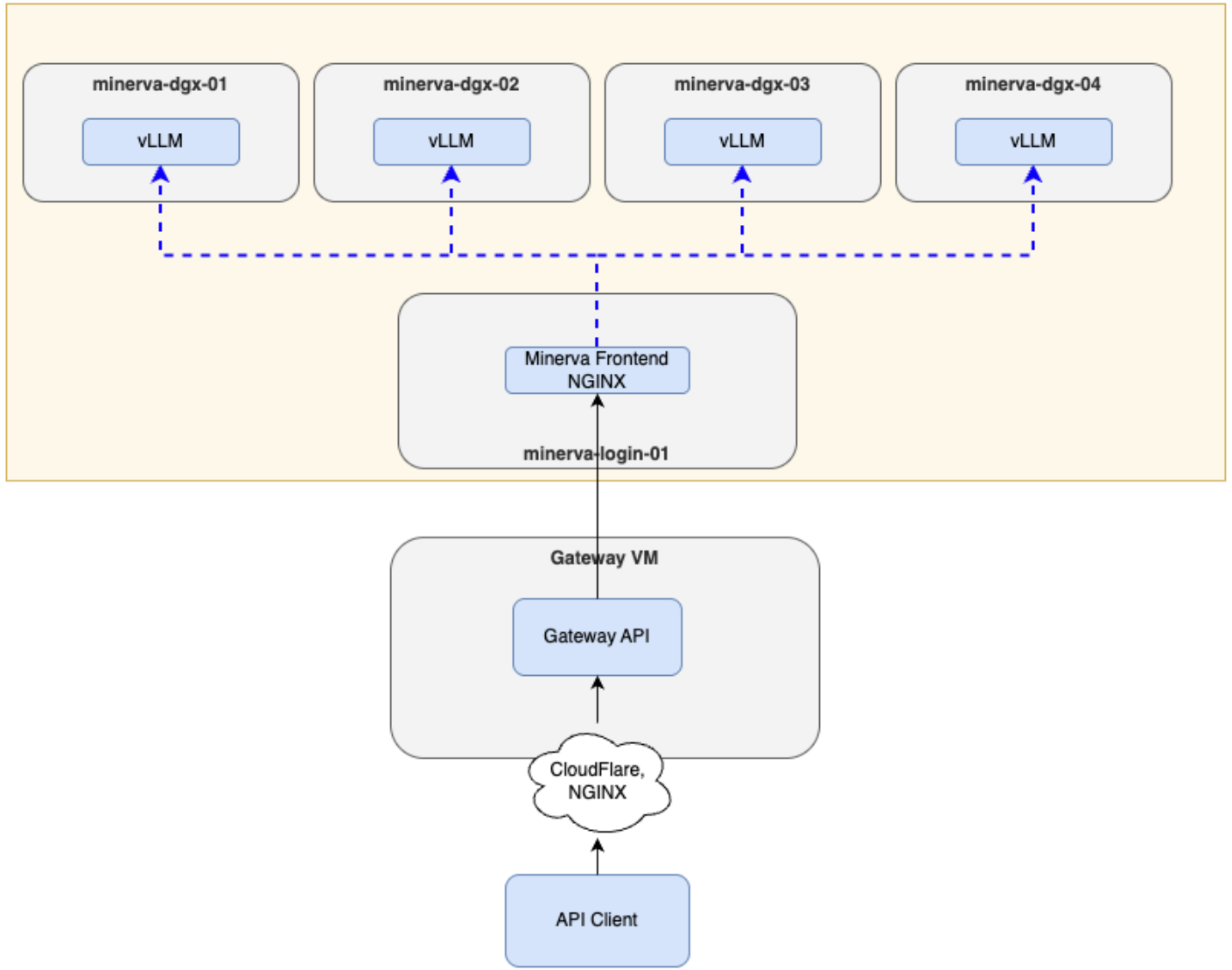



- **Metis:** Direct HTTPS proxy to SambaStack frontend
- **Minerva:** Direct HTTPS proxy to cluster frontend
- **Sophia:** inference requests invoked through Globus Compute

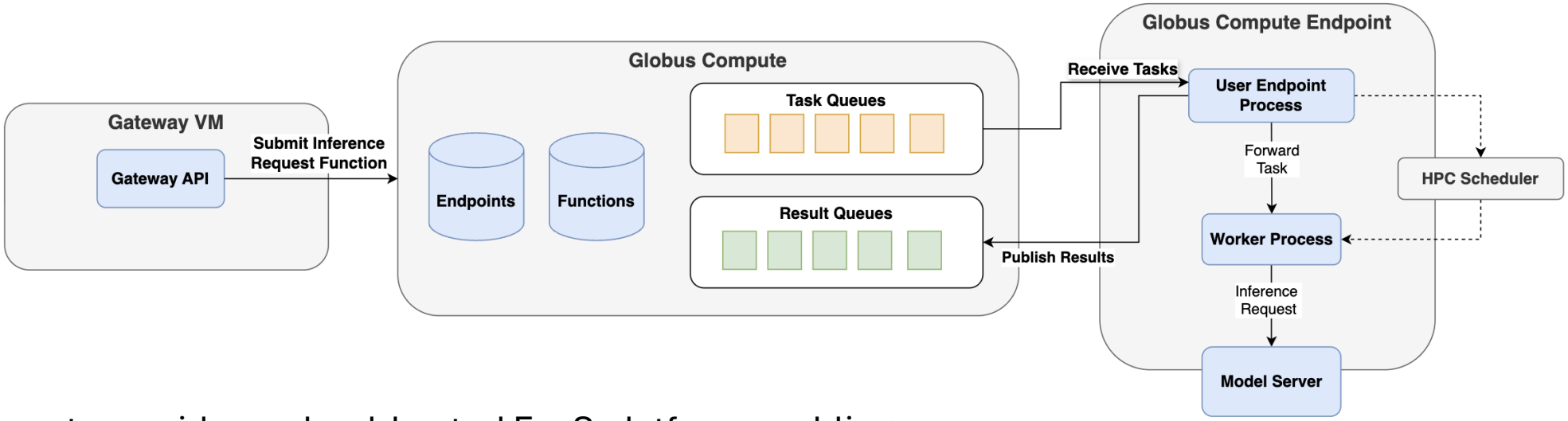
# Metis


Gateway VM
Gateway API
CloudFlare,
NGINX
API Client
Metis SambaStack
Inference API
Inference Platform
(Scheduling, Routing, Batching)
SambaRack Manager
SambaRack
SN40L
SambaRack
SN40L
SambaRack
SN40L

# Minerva


minerva-dgx-01
vLLM
minerva-dgx-02
vLLM
minerva-dgx-03
vLLM
minerva-dgx-04
vLLM
Minerva Frontend NGINX
minerva-login-01
Gateway VM
Gateway API
CloudFlare, NGINX
API Client

# Sophia

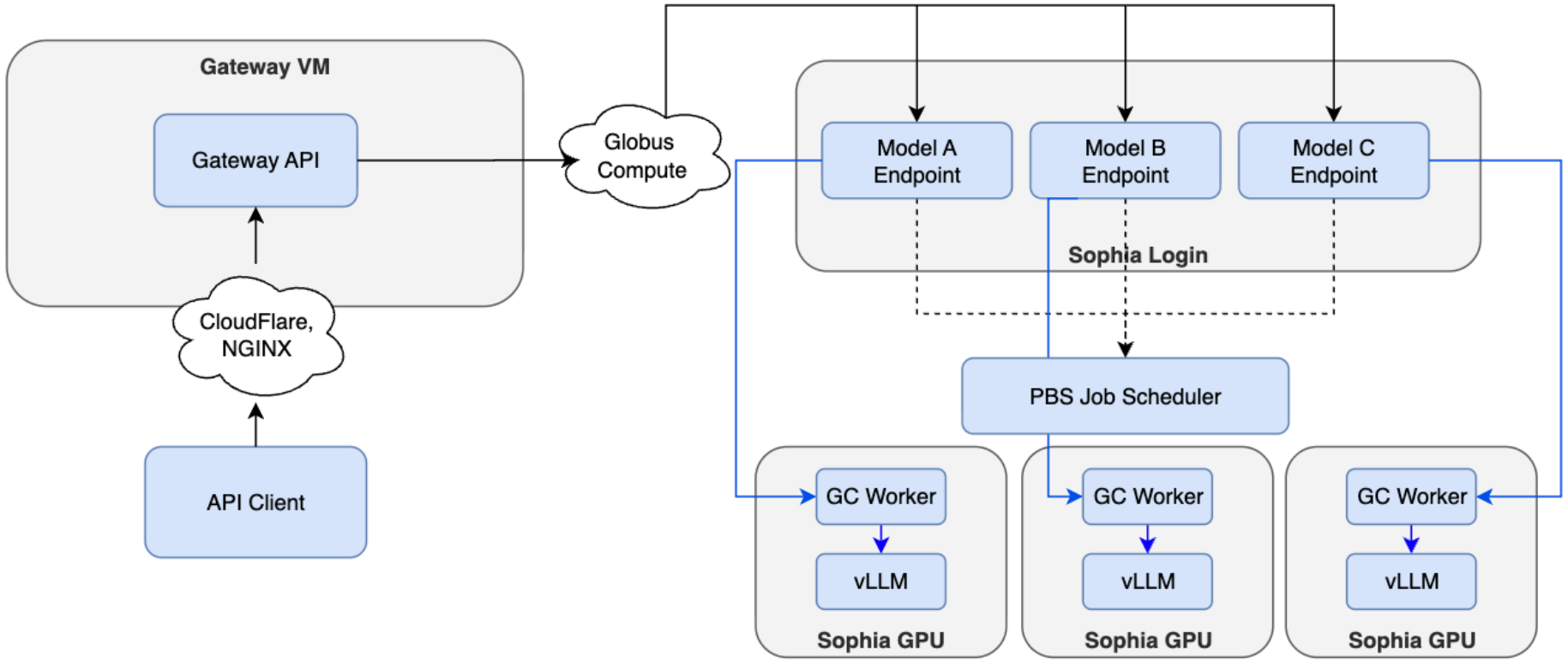



- Globus Compute provides a cloud-hosted FaaS platform enabling distributed Python function execution across **endpoints** running anywhere

- The Inference Service adopts this platform to simultaneously provision models, autoscale, and route inference traffic on the Sophia cluster

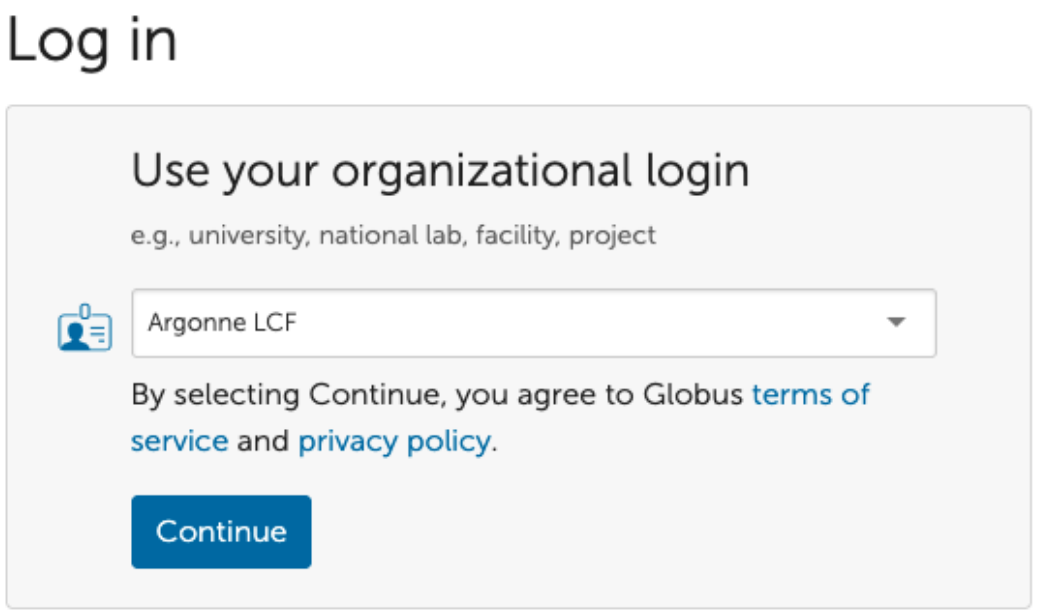
https://globus-compute.readthedocs.io/

# Sophia


Gateway VM
Gateway API
CloudFlare, NGINX
API Client
Globus Compute
Model A Endpoint
Model B Endpoint
Model C Endpoint
Sophia Login
PBS Job Scheduler
GC Worker
vLLM
Sophia GPU
GC Worker
vLLM
Sophia GPU
GC Worker
vLLM
Sophia GPU

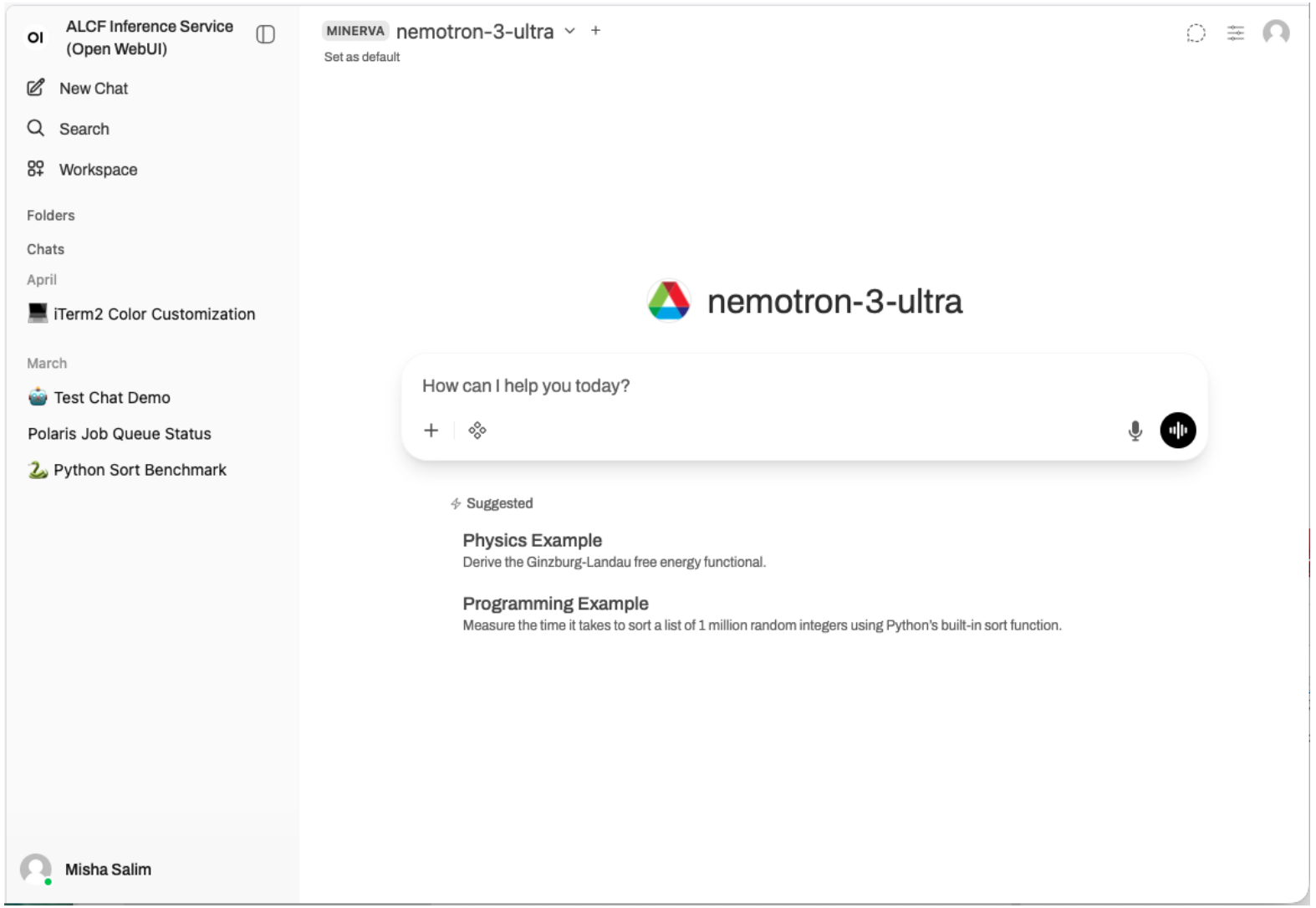
# Web Chat UI

- https://inference.alcf.anl.gov
  - Sign in with Globus, providing your ALCF credentials



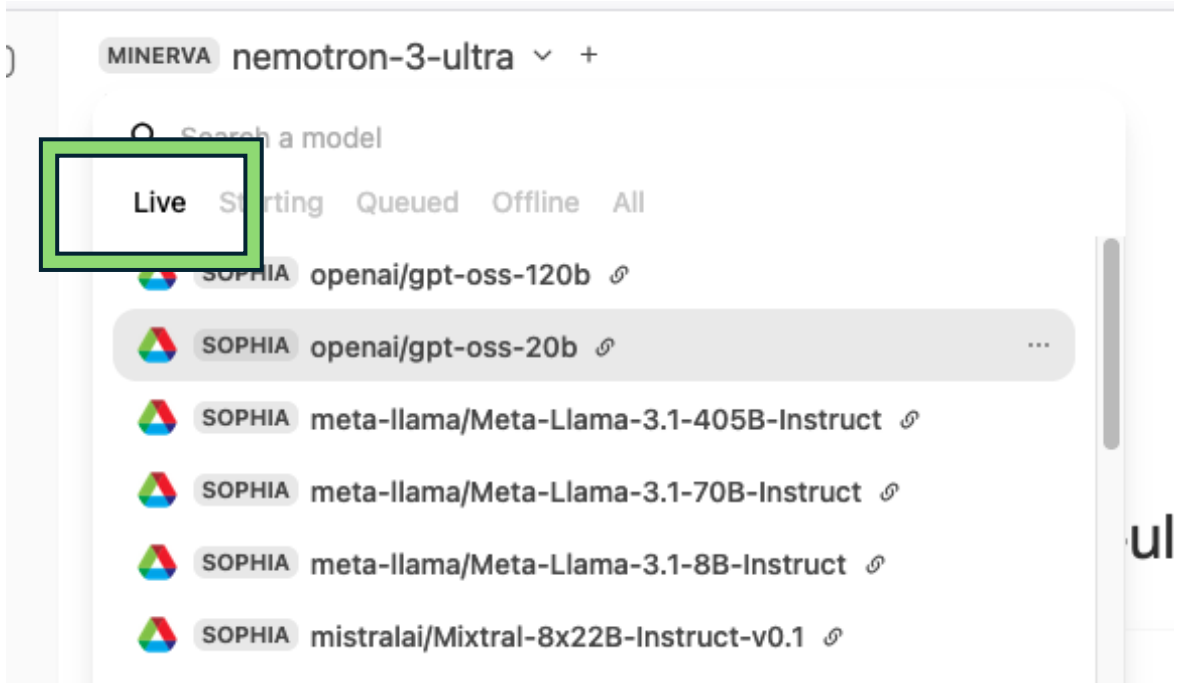
https://inference.alcf.anl.gov

# Web Chat UI

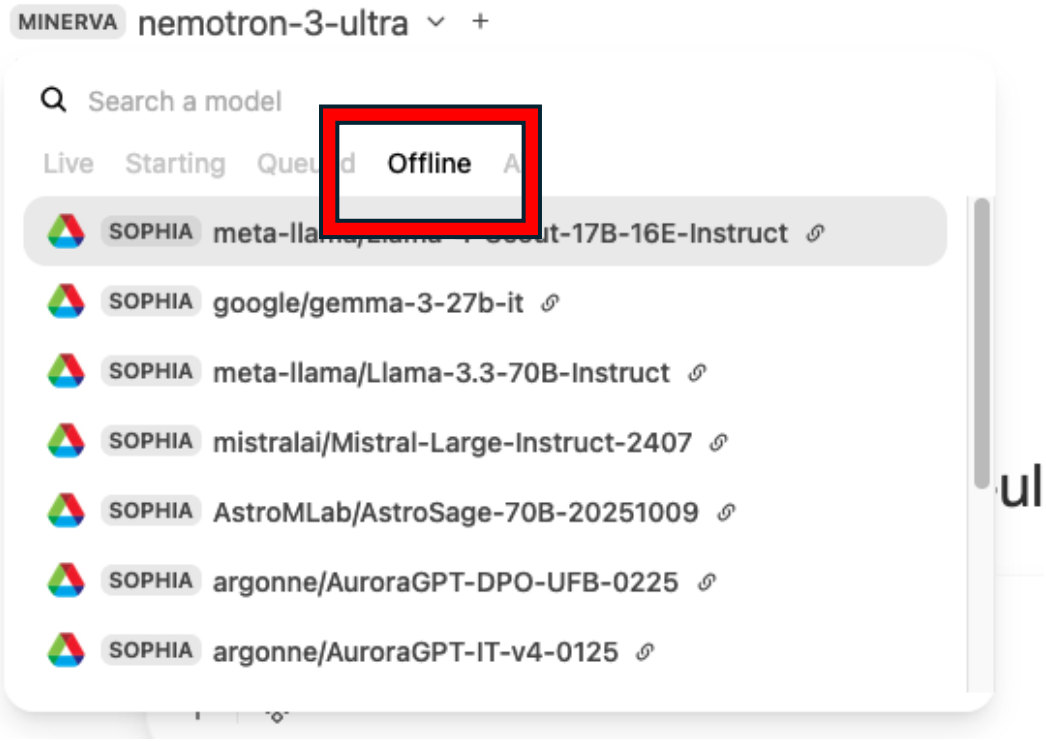


https://inference.alcf.anl.gov

# Web Chat UI



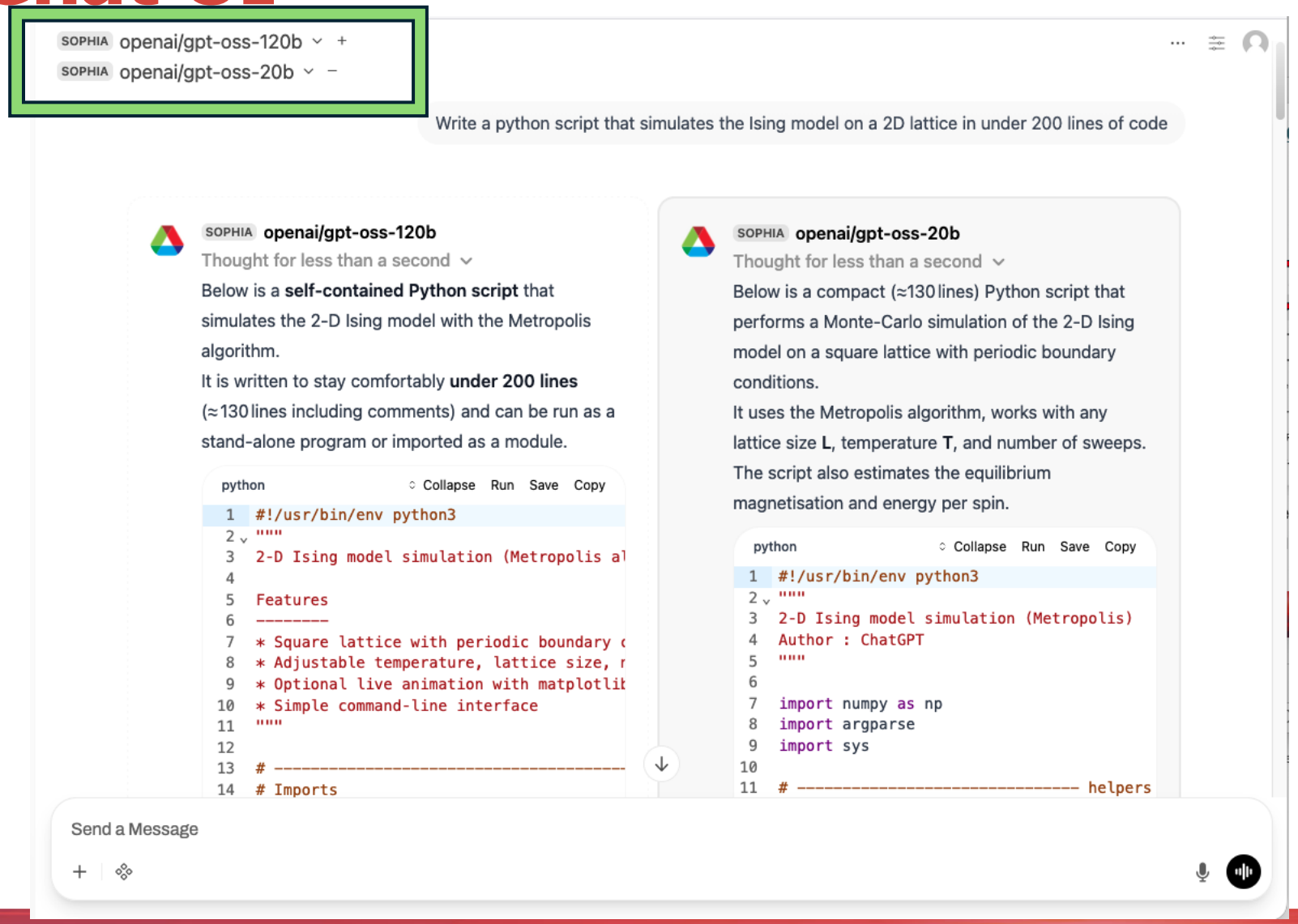

**Live models**: immediate response




**Offline Models**: queued for cold-start upon request

https://inference.alcf.anl.gov

# Web Chat UI



# API Access: Resources

https://docs.alcf.anl.gov/services/inference-endpoints

Back to top

## API Access ¶

For programmatic access, you can use the API endpoints directly.

**1. Setup Your Environment**

You can run the following setup from anywhere (your local machine, or an ALCF machine).

```
# Create a new Conda environment
conda create -n globus_env python==3.11.9 --y
conda activate globus_env

# Install necessary packages
pip install openai globus_sdk
```

**2. Authenticate**

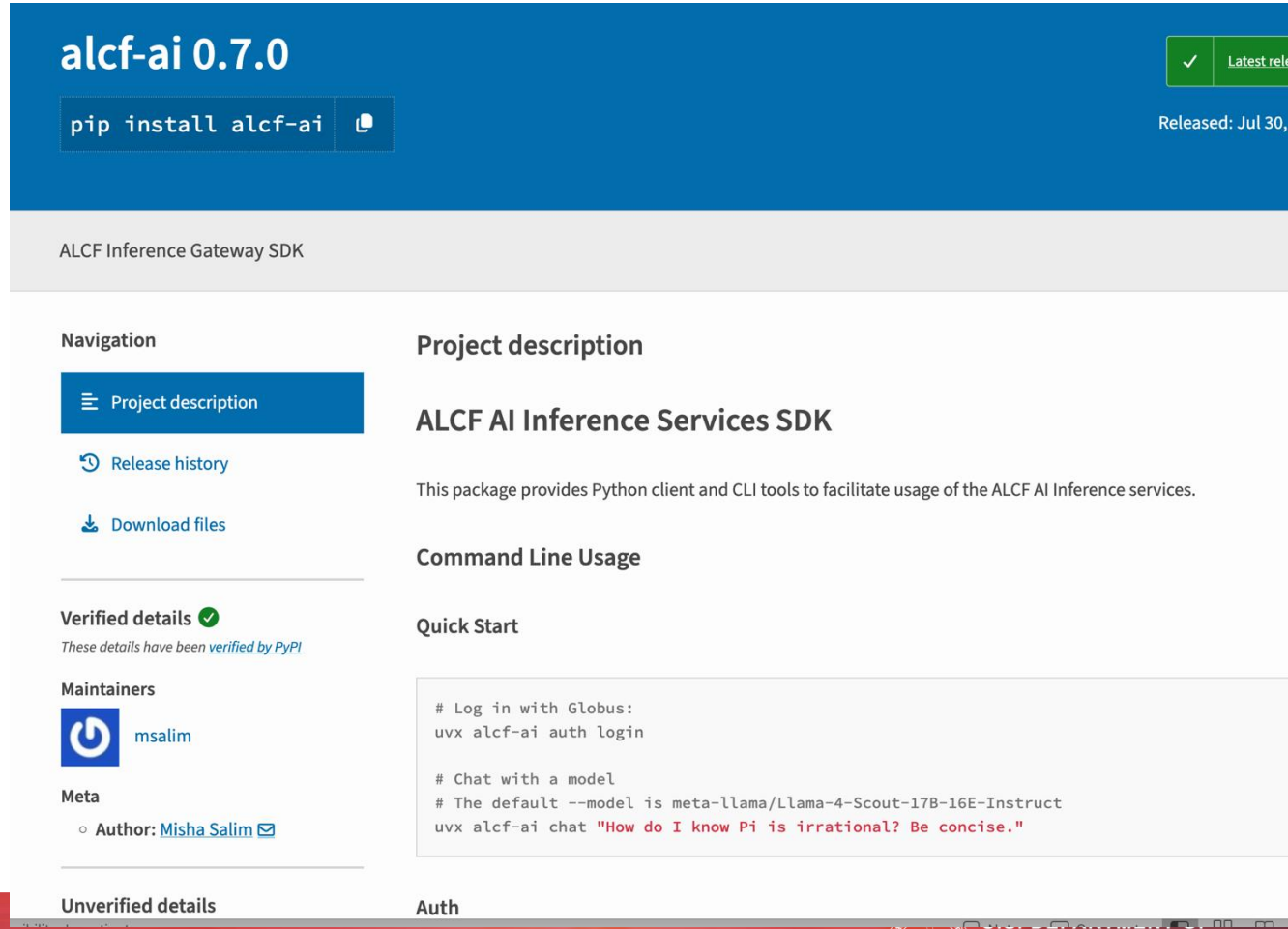
To access the endpoints, you need an authentication token.

```
# Download the authentication helper script
wget https://raw.githubusercontent.com/argonne-lcf/inference-endpoints/refs/heads/main/inference

# Authenticate with your Globus account
python inference_auth_token.py authenticate
```

This will generate and store access and refresh tokens in your home directory. To see how much ti
before your access token expires, type the following command ( `units` can be seconds, minutes, (

```
python inference_auth_token.py get_time_until_token_expiration --units seconds
```

https://pypi.org/project/alcf-ai/



# API Access: CLI

**alcf-ai** package provides self-contained CLI and Python SDK

## uv package manager

```
$ uvx alcf-ai auth login

Please authenticate with Globus here:
------------------------------------
https://auth.globus.org/v2/oauth2/authorize?client_id=58f
scope=https%3A%2F%2Fauth.globus.org%2Fscopes%2F681c10cc-f
d4-a327-1af7d143283e%2Fhttps+urn%3Aglobus%3Aauth%3Ascope%
B3EkslugnWt02mPuHJOyNLpjDeGS6Xs&code_challenge_method=S25
42-4548-b5ce-17e498c84f6a
------------------------------------

Enter the resulting Authorization Code here:
```

## venv / pip

```
$ python3.12 -m venv test_env
$ source test_env/bin/activate
((test_env) ) $ pip install alcf-ai
```

```
((test_env) ) $ alcf-ai auth login
```

# API Access: CLI

- **alcf-ai ls-endpoints:** model discovery
- **alcf-ai ls-jobs <cluster>:** current model status

```
$ uvx alcf-ai ls-jobs sophia
{
    'running': [
        {
            'Models': 'google/gemma-4-E4B-it',
            'Framework': 'vllm',
            'Cluster': 'sophia',
            'Model Status': 'running',
            'Job ID': '168029.sophia-pbs-01.lab.alcf.anl.gov',
            'Job State': 'R',
            'Host Name': 'sophia-gpu-14/0*256',
            'Job Comments': 'Job run at Mon Jul 20 at 14:56 on (sophia-gpu-14:ncpus=256:ngpus=8:mem=1006632960kb)',
            'Nodes Reserved': '1',
            'Walltime': '267:34:41'
        },
        {
            'Models': 'google/gemma-4-31B-it',
            'Framework': 'vllm',
            'Cluster': 'sophia',
            'Model Status': 'running',
            'Job ID': '168030.sophia-pbs-01.lab.alcf.anl.gov',
            'Job State': 'R',
            'Host Name': 'sophia-gpu-15/0*256',
            'Job Comments': 'Job run at Mon Jul 20 at 14:56 on (sophia-gpu-15:ncpus=256:ngpus=8:mem=1006632960kb)',
            'Nodes Reserved': '1',
            'Walltime': '267:34:45'
        },
```

# API Access: CLI

- **alcf-ai chat:** send Chat Completions requests

```
$ uvx alcf-ai chat --model gpt-oss-120b --cluster metis "Test; what model are you?"
I'm ChatGPT, a large language model built on OpenAI's GPT-4 architecture.
$
```

```
$ cat test.py | uvx alcf-ai chat --model gpt-oss-120b --cluster metis -s "Summarize what this code does."
Summary

The script runs a batch of Triton inference jobs in parallel, handling all the necessary data-movement steps automatically.

 1 Setup
    • Imports a thread-pool executor, Path, the InferenceClient from the alcf_ai SDK, and a constant STAGING_COLLECTION_ROOT.
    • Instantiates an InferenceClient.
    • Defines a directory that holds sample input files (sample_dir), a list of model names (models), and the ID of the source
 2 run_inference function
    • Stage-in – Copies a local input file (input_path) into the remote staging collection using client.stage_in. The remote pa
      STAGING_COLLECTION_ROOT and the destination returned by the stage-in call.
    • Prepare output path – Derives a remote output filename by replacing the input file's extension with .output.npz.
    • Submit inference – Calls client.d3_triton.submit with the model name, remote input path, and remote output path. The call
    • Poll for completion – Waits for the Triton task to finish with client.d3_triton.poll_task_result.
    • Stage-out – Copies the remote result file back to the local filesystem, naming it with the same base as the input but wit
    • Returns the result dictionary (which includes at least the output_path and other metadata).
 3 Parallel execution
```

# API Access: SDK

```python
from alcf_ai import InferenceClient
from rich import print

# Automatically uses cached refresh tokens from previous login:
client = InferenceClient()

# Programmatically discover endpoints:
print(client.list_endpoints()["clusters"]["sophia"])

# Get an OpenAI API client for an ALCF cluster:
oai = client.clusters("sophia").openai
print(
    oai.chat.completions.create(
        model="openai/gpt-oss-120b",
        messages=[{"role": "user", "content": "Hello there!"}],
    )
)
```

```
$ uv run --with alcf-ai sdk.py
```

```
ChatCompletion(
    id='chatcmpl-8a534e417ff4ef77',
    choices=[
        Choice(
            finish_reason='stop',
            index=0,
            logprobs=None,
            message=ChatCompletionMessage(
                content="Hey there! 👋\n\nWhat's on your mind today? I'm hap
topic, brainstorming ideas, or just chatting. Let me know!",
                refusal=None,
                role='assistant',
                annotations=None,
                audio=None,
                function_call=None,
                tool_calls=[],
                reasoning='We need to respond as ChatGPT. The user says "Hel
about how can help.'
            ),
            stop_reason=None,
            token_ids=None
        )
    ],
```

# API Access: SDK

```python
from alcf_ai import InferenceClient
from rich import print
from langchain_openai import ChatOpenAI

client = InferenceClient()
oai = client.clusters("sophia").openai

llm = ChatOpenAI(
    model="openai/gpt-oss-120b",
    api_key="unused",
    client=oai.chat.completions,
)

print(llm.invoke("Say hello, world!").content)
```

```
$ uv run --with alcf-ai,langchain-openai sdk.py
Hello, world!
```

# API Access: any HTTPS client!

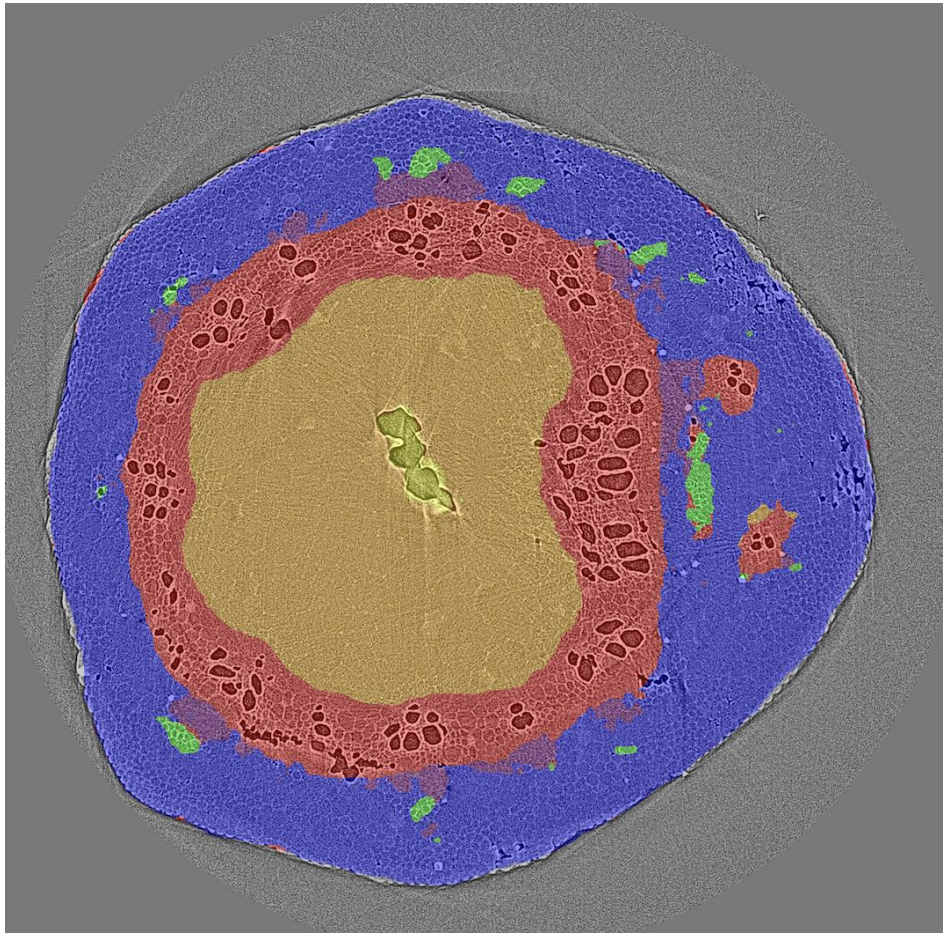
```bash
#!/bin/bash

# Get your access token
access_token=$(uvx alcf-ai auth get-access-token)

curl -X POST "https://inference-api.alcf.anl.gov/resource_server/metis/api/v1/chat/completions" \
     -H "Authorization: Bearer ${access_token}" \
     -H "Content-Type: application/json" \
     -d '{
            "model": "gpt-oss-120b",
            "messages":[{"role": "user", "content": "Explain quantum computing in simple terms."}]
         }'
```

# API Access: Vision Models

- Segmentation Models like **SAM3** and **DINOv3** enable *real-time* AI-assisted segmentation of images generated at DOE light sources

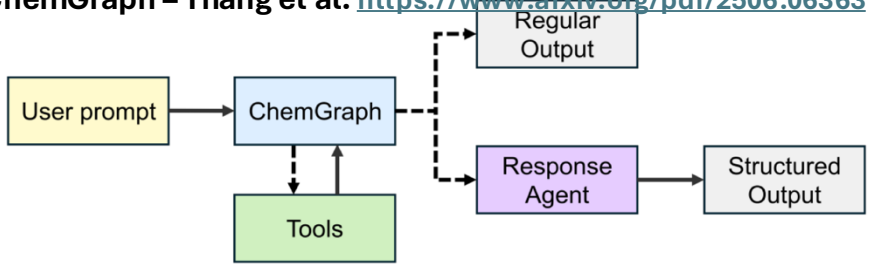
- **alcf-ai** orchestrates bulk image transfers; staging data to/from Inference Service

```
$ uvx alcf-ai dinov3 submit \
>   --origin-collection-id $SOURCE_COLLECTION \
>   --save-overlay \
>   /datascience/msalim/sam3_datasets/APS/SYNAPS-I/example/raw
```



# Highlighted Use Cases

**ChemGraph – Thang et al.** https://www.arxiv.org/pdf/2506.06363

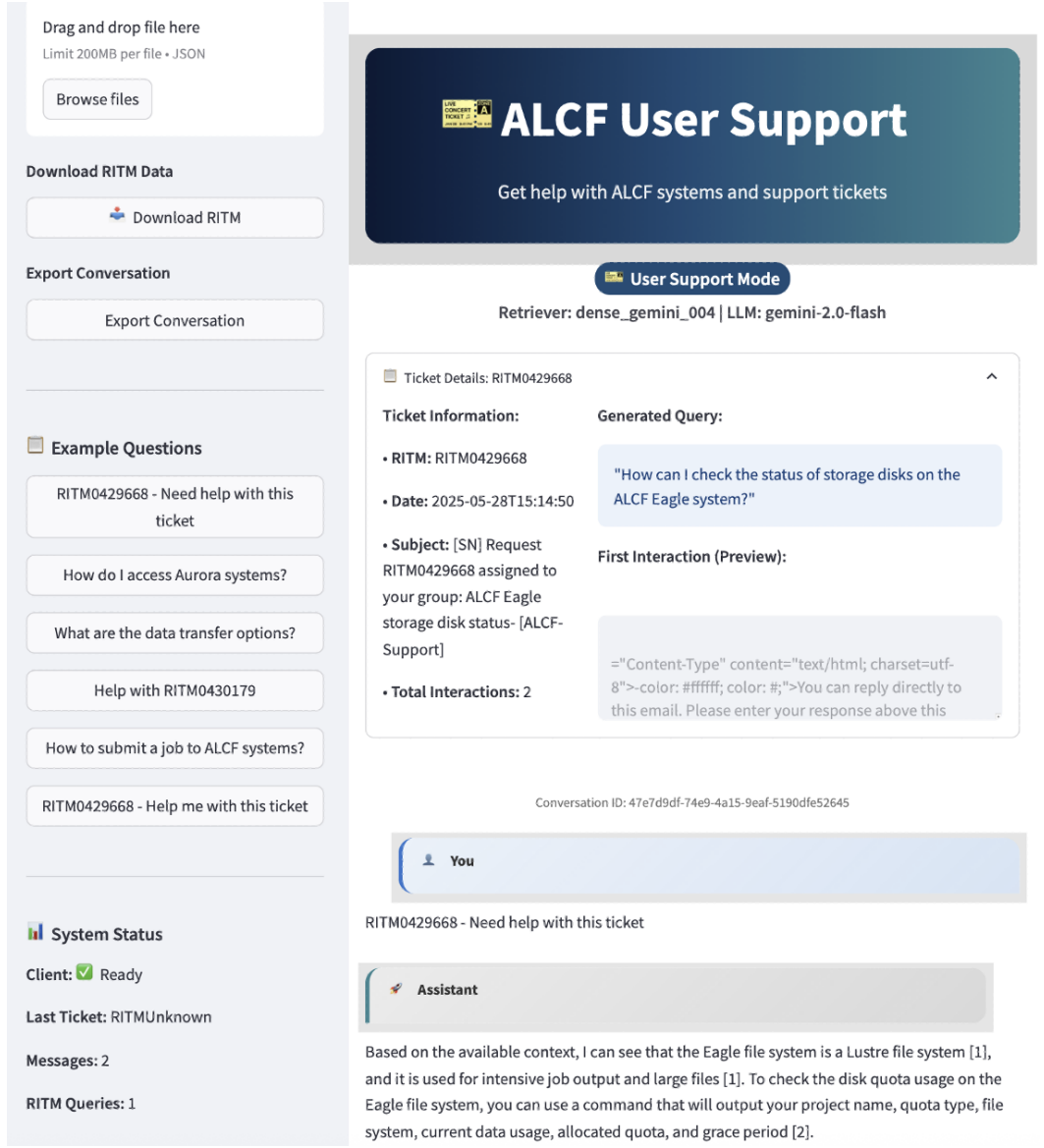



**Workflow and Cheminformatics**

- LangGraph
- ASE
- RDKit
- PubChemPy

**Simulation Backends**

| Semi-empirical | Ab initio | ML Potentials |
|---|---|---|
| - xTB | - NWChem | - MACE |
| - EMT | - ORCA | - UMA |



# Looking Ahead

- To meet rapidly increasing demand from agentic workflows and Genesis Mission projects, scaling capacity is a key focus in 2026
- Continually updating model catalog and expanding support for workflows using LLM, vision, and scientific foundation models

# Looking Ahead

- The Inference Service APIs provide compatibility with:
  - OpenAI Chat Completions
  - OpenAI Responses
  - Anthropic Messages

- Most agents interoperate with one of these APIs: simply point your agent at the desired base URL https://inference-api.alcf.anl.gov/resource_server/
  - Caveat: we are actively working to improve *streaming* support, which is critical to reliable operation and avoiding proxy timeouts with long agentic traces

# Looking Ahead

https://pi.dev/

```
28     "alcf-minerva": {
29       "baseUrl": "https://inference-api.alcf.anl.gov/resource_server/minerva/api/v1",
30       "api": "openai-completions",
31       "apiKey": "!uvx alcf-ai auth get-access-token",
32       "compat": {
33         "supportsDeveloperRole": false,
34         "supportsReasoningEffort": false
35       },
36       "models": [{ "id": "inkling-bf16" }]
37     }
38   }
39 }
".pi/agent/models.json" line 39 of 39 --100%-- col 1
```

# Looking Ahead

https://pi.dev/

Stay tuned for upcoming sessions on Agentic Workflows and Coding!

```
$ pi --provider alcf-minerva --model inkling-bf16

pi v0.79.6
escape interrupt · ctrl+c/ctrl+d clear/exit · / commands · ! bash · ctrl+o more
Press ctrl+o to show full startup help and loaded resources.

Pi can explain its own features and look up its docs. Ask it how to use or extend Pi.


Update Available
New version 0.83.0 is available. Run pi update
Changelog: open changelog


Write a short python script that prints 'hello world' in the file hello.py

The user wants a short Python script that prints 'hello world' in a file called hello.py. I'll create the file with the
appropriate content.

write ~/hello.py

print('hello world')

The file has been written successfully. I should confirm briefly.
```

# Acknowledgements


- ALCF
  - Operations
  - Cybersecurity
  - Data Services and Workflows Team
  - AI/ML Team
- Benoit Côté
- Ethan Wong
- Nathan Nichols

# ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

Produced by Argonne National Laboratory, a U.S. Department of Energy Laboratory managed by UChicagoArgonne, LLC under contract DE-AC02-06CH11357.

Special thanks to the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF) for the use of their resources during the training event.