Infrastructure

- Accounts
- Wiki
- Compute and Storage
- Project management platform
- Snipe-IT
- Zulip
- LiteLLM
- Monitoring
- Managing SSH keys with Kanidm

Accounts

Your lab account

You can use your lab account to log into the following services:

- This wiki.
- The lab's project management platform
- The lab's Zulip instance.

Other services may be added in the future.

Adding passkeys/changing password

If you need to add new passkeys or change your password, you can do so by visiting https://idm.lab.pyarelal.xyz .

Wiki

Purpose

We will use this wiki as a place to put the following:

- Lab news
- Lab policies
- Lab procedures
- Lab resources
- Lab infrastructure
- Lab member profiles
- Lab seminar (aka journal club/reading group) schedules
- Public-facing project pages
- And more...

What does **not** go into this wiki:

- Credentials (usernames/passwords)
  - These should be shared via Stache.
- Correspondence

Organization

The wiki is based on Bookstack. The user documentation for Bookstack can be found here.

Public vs. Private content

Bookstack allows fine-grained visibility controls. Only lab members (i.e., people with lab accounts) have the ability to edit pages on this wiki. However, there are a couple of Books that are viewable by the public:

- Public
- Lab Manual

The 'Public' book is meant to act like a kind of 'landing page' for the public, and has things like lab news, etc.

The 'Lab Manual' book contains lab policies and procedures, and is public since we want to share this information with prospective students, etc.

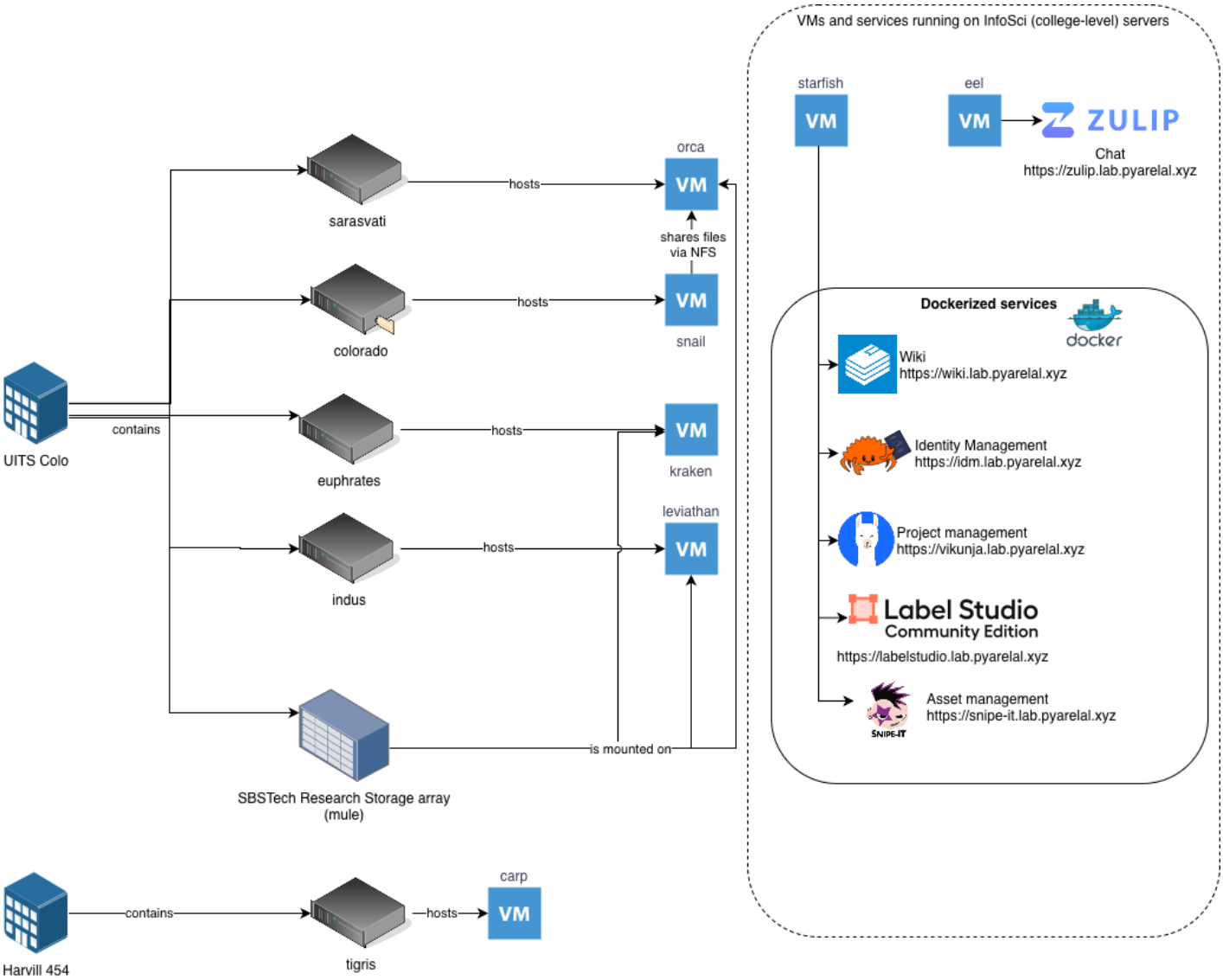
You are welcome to create additional books, pages, and chapters as you see fit. These are subject to being curated by the lab PIs, especially if they are public.

If you want to create a new page and are not sure where to put it, you can put it in the 'Miscellaneous' book (visible only to lab members), and move it elsewhere in the future if it makes sense.

Editing

The wiki has support for both WYSIWIG and Markdown editing. Feel free to use whichever one you prefer. However, if you are copying content from a PDF file to a wiki page, please use the Markdown editor, in order to prevent the creation of spurious HTML markup.

Compute and Storage



Compute

The ML4AI lab has the following compute VMs:

VM Name	CPU	RAM	GPUs
kraken	AMD EPYC 7662 64-Core Processor (2.0 GHz)	720 GB	2x NVIDIA A100 (40 GB)

VM Name	CPU	RAM	GPUs
leviathan	AMD EPYC 7763 64-Core Processor (2.45 GHz)	720 GB	6x NVIDIA RTX A6000
carp	AMD EPYC-Rome Processor	95 GB	1x NVIDIA GeForce RTX 3090
orca	AMD EPYC 9474f, 48-core, 3.60 GHz, 256MB cache	1.5 TB (tentative)	2x NVIDIA H100 NVL (94GB hbm3, PCIE 5.0 x16)

TODO:

- Add information about venti

Storage

The lab has a 20 TB NFS share mounted at /media/mule on the kraken, leviathan, and orca VMs.

There is a 90 TB NFS share mounted at /media/snail-ssd on the orca VM.

TODO:

- Add information about timelord

TODO:

- Add information about other legacy SISTA systems that are still operational.

Backup

- VMs running on InfoSci servers (e.g., eel, starfish) are backed up every 6 hours
- As of 2025-03-06, VMs running on lab servers (e.g., kraken, leviathan) are not backed up, but the plan is to include them in the backup system in the future.

Project management platform

We have a self-hosted instance of the Vikunja project management app running at https://vikunja.lab.pyarelal.xyz.

Snipe-IT

We have an instance of Snipe-IT at https://snipe-it.lab.pyarelal.xyz that we will use to keep track of our equipment and consumables.

You can log into the Snipe-IT instance using your lab credentials (username and Unix password).

Zulip

We have an instance of Zulip set up at https://zulip.lab.pyarelal.xyz, to enable efficient communication.

LiteLLM

The lab runs a LiteLLM proxy that gives you access to large language models running on the lab's GPU server (orca), using an OpenAI-compatible API. This lets you use tools like Python scripts, curl, and Claude Code with local open-source models without needing an external API account.

Getting access

Email adarsh@arizona.edu to request an API key. Include a brief description of how you plan to use it.

The API base URL is: `https://litellm.lab.pyarelal.xyz`

Once you have a key, set it as an environment variable so it persists across sessions. Add this to your shell config file (e.g. `~/.bashrc`, `~/.zshrc`):

```
export LITELLM_API_KEY=sk-...
```

Then reload your shell: `source ~/.bashrc` (or open a new terminal).

Available models

To see which models are currently available:

```
curl https://litellm.lab.pyarelal.xyz/models \
  -H "Authorization: Bearer $LITELLM_API_KEY"
```

Models are named `<family>:<size>[-a<N>b]-<quant>`, e.g. `qwen3.6:35b-a3b-q8_0`. The name tells you three things: total parameter count (`35b`), whether it's a mixture-of-experts model (`a3b` = only 3B parameters active per token; no `a`-suffix means dense), and the quantization level (`q8_0` = 8-bit, near-lossless; `q4_k_m` = 4-bit). Dense models are generally stronger per total parameter; MoE models generate faster for their size.

Using with curl

```
curl -X POST https://litellm.lab.pyarelal.xyz/chat/completions \
  -H "Authorization: Bearer $LITELLM_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "qwen3.6:27b-q8_0",
    "messages": [{"role": "user", "content": "Hello!"}]
  }'
```

Using with Python

Install the OpenAI SDK if you don't have it: `pip install openai`

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ["LITELLM_API_KEY"],
    base_url="https://litellm.lab.pyarelal.xyz",
)

response = client.chat.completions.create(
    model="qwen3.6:27b-q8_0",
    messages=[{"role": "user", "content": "Hello!"}],
)
print(response.choices[0].message.content)
```

Using with Claude Code

You can use Claude Code with the lab's models by pointing it at LiteLLM instead of Anthropic's API. Set these environment variables before running `claude`:

```
export ANTHROPIC_API_KEY=$LITELLM_API_KEY
export ANTHROPIC_BASE_URL=https://litellm.lab.pyarelal.xyz
claude
```

Then switch to a lab model inside Claude Code with the `/model` command:

```
/model qwen3.6:27b-q8_0
```

Note: open-source models have different capabilities than Claude — some Claude Code features (e.g. complex tool use) may not work as well.

Monitoring

Overview

The lab uses a self-hosted monitoring stack to track CPU, GPU, memory, disk, network, and per-process resource usage across all lab servers. Metrics are visualised in Grafana, which is available at https://grafana.lab.pyarelal.xyz. Log in with your lab account via the **Sign in with Kanidm** button.

What is monitored

- CPU usage (by type: user, system, iowait, etc.)
- RAM usage (used, cached, buffers)
- Network traffic (sent and received)
- Disk I/O (read and write)
- GPU utilisation, memory, temperature, and power draw (on GPU-equipped hosts)
- Top processes by CPU and memory

Monitored hosts

Host	GPU monitoring
orca	Yes (NVIDIA)
kraken	Yes (NVIDIA)
leviathan	Yes (NVIDIA)
starfish	No
eel	No

Using the dashboard

After logging in, open the **Infrastructure Overview** dashboard. Use the **Host** dropdown at the top to switch between servers. The time range selector in the top right controls how far back the graphs show.

The dashboard is divided into three sections:

- **System** — CPU, RAM, network, and disk panels visible for all hosts
- **GPU** — GPU panels, populated only for GPU-equipped hosts
- **Processes** — top 10 processes by CPU and memory usage

Managing SSH keys with Kanidm

> **For lab members who joined before 2026-05-22.** orca's logins are now managed centrally through Kanidm for new members. If you already had an orca account before that date, your existing access still works and you don't need to follow this.

This page covers SSH access to **orca**, whose logins are managed through the lab's identity server, Kanidm. You register your SSH **public key** with Kanidm once, then you can log in. (For web services and passwords, see Accounts.) Other lab machines are still being migrated — ask Adarsh for access to those.

1. Make sure you have an SSH key

On your laptop:

```
ls ~/.ssh/id_ed25519.pub
```

If that file exists, skip to step 2. Otherwise create one:

```
ssh-keygen -t ed25519 -C "your-name@arizona.edu"
```

Press Enter for the default location; a passphrase is recommended. This creates a **private** key (`~/.ssh/id_ed25519` — never share it) and a **public** key (`~/.ssh/id_ed25519.pub` — safe to share).

2. Copy your public key

```
cat ~/.ssh/id_ed25519.pub
```

Copy the whole line — it starts with `ssh-ed25519`.

3. Register it with Kanidm

You have two options for doing this: via the web (easier) or via the command line.

3.1 Web (easier)

Sign in at idm.lab.pyarelal.xyz → Profile → Credentials -> **SSH Keys** → **Add SSH Key**, paste the public key from step 2, and give it a Title (e.g. `my-macbook-pro`).

3.2 Command line

If you have the `kanidm` client tools:

```
kanidm login -D <your-username>
kanidm person ssh add-publickey <your-username> <label> "ssh-ed25519 AAAA... your-name@arizona.edu"
```

(The label, e.g. `my-macbook-pro`, just lets you tell keys apart — use one per device.)

4. Log into orca

```
ssh <your-username>@orca.infosci.arizona.edu
```

The first time, your shell may take a moment while your home directory is created.

Troubleshooting

- **`Permission denied (publickey)`** — the key registered in Kanidm doesn't match the one your laptop is offering. Recheck steps 2–3, or run `ssh -v <your-username>@orca.infosci.arizona.edu` to see which key is being tried.
- **Asked for a password** — your key isn't being found; confirm it's added in Kanidm and that you're connecting as the right username.
- Still stuck? Contact **Adarsh**.