EPHEMERAL

Compute options

ar064 · 11 August 2026

This is the operational map for pinglab compute. It separates the persistent control plane from the machines that perform numerical work, and records the authentication boundary for each provider. Prices, balances, queues, and GPU stock change. The commands below query those values when a decision is made instead of preserving a stale snapshot here.

Never place a password, TOTP seed, private SSH key, RunPod API key, or Modal token in this repository. Authentication commands either use an encrypted local key or open the provider's own interactive login flow.

1. Local Mac

When to use. Use the Mac for editing, Demolab preview, plotting, analysis of collected results, dry-run dispatch plans, and plumbing-scale tests. It is the shortest feedback loop. It is not a CUDA training machine.

Provider overview. This is the local development workstation. It has no scheduler and no marginal compute charge. The working checkout is `/Users/eoin/pinglab`.

How to use. Local access requires the macOS user session rather than a separate infrastructure login.

```
cd /Users/eoin/pinglab
uv sync --dev

# Run an experiment locally.
uv run python experiments/expNNN.py

# Preview the Demolab collection.
demolab dev
```

Keep cloud commands in dry-run mode until spending has been explicitly authorised.

2. Hetzner control plane

When to use. Use Hetzner for the persistent Codex session, orchestration, monitoring, result collection, and the public development preview. Do not use its small CPU and memory allocation for substantial numerical experiments.

Provider overview. Hetzner supplies the always-on Linux host named `pinglab-codex`. Caddy terminates HTTPS for `pl-hetzner.eoinmurray.info` and proxies to the Demolab development server on port `3000`. The server is a control plane, not a GPU worker.

How to use. The Mac alias `hetzner` authenticates as `eoin` with the dedicated Ed25519 identity configured in `~/.ssh/config`. The private key remains on the Mac. The Hetzner host has its own dedicated key for Olorin.

```
ssh hetzner

# Inspect the persistent session and preview service.
tmux list-sessions
tmux attach -t shd-autoresearch
systemctl --user status demolab-dev.service

# Reach Olorin from the control plane.
ssh olorin 'hostname; whoami'
```

3. Olorin

When to use. Use Olorin for large single-GPU or multi-GPU work when a GPU is visibly idle and the Division F fair-use policy permits the allocation. Its large VRAM makes it the first choice for workloads that do not fit on consumer GPUs.

Provider overview. Olorin is a shared Division F machine with four NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs, each with approximately 96 GB of VRAM. It currently has no Slurm scheduler. Availability is therefore manual and may change between inspection and launch.

How to use. The Mac alias `olorin` connects through `gate.eng.cam.ac.uk` using `~/.ssh/olorin_codex`. Hetzner has a separate dedicated key. Passwords are not stored. Check utilization immediately before every launch, select only an idle GPU, and keep heavy files under `/scratch/em586`.

```
ssh olorin
nvidia-smi

# Inspect utilization without opening an interactive shell.
ssh olorin \
  'nvidia-smi --query-gpu=index,utilization.gpu,memory.used,memory.total \
  --format=csv,noheader'

# On Olorin, after confirming that GPU N is idle:
cd /scratch/em586/pinglab
CUDA_VISIBLE_DEVICES=N uv run python experiments/expNNN.py
```

Never launch on all four GPUs merely because the machine accepts the command. Shared infrastructure without a scheduler requires more restraint, not less.

4. CSD3 Wilkes3, Service Level 2

When to use. Use SL2 for planned production runs where a queue of hours is acceptable. It is the default scheduled backend for long, reproducible A100 jobs and should consume the purchased allocation before commercial cloud credits.

Provider overview. Wilkes3 is Cambridge's Slurm-managed A100 cluster. The SL2 GPU account is `OLEARY-SL2-GPU`; the Ampere partition uses QOS `gpu1`. Account balance and scheduler estimates are live administrative state and must be queried at submission time.

How to use. Connect as Cambridge user `em586`. SSH requires the Raven/UIS password plus the six-digit token labelled `CSD3: SSH Login`. The `csd3` alias uses `ControlMaster` so subsequent commands reuse the authenticated connection. An SSH key can replace the password factor, but mandatory TOTP remains unless Research Computing Services arranges an automation-specific solution.

```
# Establish or reuse the multiplexed connection.
ssh csd3

# Inspect purchased hours and account associations.
ssh csd3 mybalance
ssh csd3 'sacctmgr show assoc user="$USER" format=Account,Partition,QOS'

# Submit one A100 job on SL2.
sbatch \
  --account=OLEARY-SL2-GPU \
  --partition=ampere \
  --qos=gpu1 \
  --gres=gpu:1 \
  job.slurm

# Ask Slurm for an estimate without submitting.
sbatch --test-only \
  --account=OLEARY-SL2-GPU --qos=gpu1 job.slurm
```

Command-line `sbatch` options override matching `#SBATCH` lines, so one job script can target either service level.

5. CSD3 Wilkes3, Service Level 3

When to use. Use SL3 for non-urgent work that can wait several days, for overflow, or for jobs submitted well before their results are needed. It is a poor interactive backend.

Provider overview. SL3 uses the same Wilkes3 A100 hardware and Ampere partition but a lower-priority service level. The GPU account is `OLEARY-SL3-GPU` and its QOS is `gpu2`.

How to use. Authentication is identical to SL2. Toggle the account and QOS at submission time:

```
sbatch \
  --account=OLEARY-SL3-GPU \
  --partition=ampere \
  --qos=gpu2 \
  --gres=gpu:1 \
  job.slurm

sbatch --test-only \
  --account=OLEARY-SL3-GPU --qos=gpu2 job.slurm

squeue -u "$USER"
```

The names are easy to invert: SL2 maps to `gpu1`, while SL3 maps to `gpu2`.

6. RunPod

When to use. Use RunPod for urgent burst capacity when Olorin is occupied and Wilkes3 is queued. Prefer a 4090 when the workload fits in 24 GB and stock exists; use a 5090 for additional VRAM or faster turnaround. Always run the smoke test before a fleet launch because allocation does not guarantee that the container will become usable promptly.

Provider overview. RunPod provides per-second GPU pods. Pinglab targets Secure Cloud in `EU-RO-1`, attaches the shared network volume, and uses `ghcr.io/eoinmurray/pinglab:cu128`, the same image used by experiment dispatch. GPU stock and regional prices are volatile. Pods must be reaped after failures because a rented but unusable pod can still bill.

How to use. `runpodctl doctor` stores the account API key in the user's RunPod configuration. Do not commit the key. The account also holds an SSH public key, optional S3 credentials for volume collection, and optional container-registry authentication for GHCR pulls.

```
# One-time interactive authentication.
runpodctl doctor

# Read-only inventory and account checks.
runpodctl gpu list
runpodctl datacenter list
runpodctl pod list -o json
runpodctl user -o json

# Free plan, then an explicitly paid capacity check.
uv run python experiments/helpers/runpod_smoke.py
uv run python experiments/helpers/runpod_smoke.py --live

# Experiment dispatch remains a dry-run without --live.
uv run python experiments/exp022.py --runpod --gpu 5090
uv run python experiments/exp022.py --runpod --gpu 5090 --live
```

```
# Kill switch for an exp022 fleet.
uv run python experiments/exp022.py --runpod --reap
```

Every pod-creating command spends money. Dry-run first, obtain explicit approval, launch, monitor, collect, and verify that `runpodctl pod list` is empty.

7. Modal

When to use. Use Modal when reliable serverless startup, automatic scaling, and reduced infrastructure management justify a higher GPU-hour price. It is useful as an escape hatch when RunPod image or SSH transport is unreliable. Only runners with an implemented Modal backend can use it.

Provider overview. Modal runs containerized functions and bills GPU, CPU, and memory by execution time. It provides managed scheduling rather than a persistent SSH host. Pinglab's Modal integration is narrower than its RunPod integration, so backend support must be confirmed in the selected runner.

How to use. `modal setup` opens Modal's authentication flow and stores a local token. Never commit token values. As with RunPod, pinglab requires `--live` before a runner dispatches paid work.

```
# One-time interactive authentication.
uv run modal setup

# Confirm the runner's options before dispatch.
uv run python experiments/exp073.py --help

# Free plan, followed by an explicitly authorised live dispatch.
uv run python experiments/exp073.py --modal
uv run python experiments/exp073.py --modal --live
```

Modal is not a drop-in flag for every experiment. If a runner does not expose `--modal`, adding a backend is implementation work rather than a command-line choice.

8. C3 Cloud

When to use. Consider C3 Cloud when UK-hosted commercial GPU capacity or institutional procurement matters, or when RunPod and Modal are unsuitable. Treat it as a fallback until pinglab has a tested deployment path.

Provider overview. C3 Cloud offers on-demand NVIDIA GPU machines, including A100, H100, and L40-class hardware. Provisioning is VM-oriented and can include a cold-start delay. Pinglab currently has no C3 dispatcher, network-volume contract, automated teardown, or capacity smoke test.

How to use. Authenticate through the C3 dashboard, provision a machine with an SSH public key, record the assigned hostname, and connect using the corresponding private key. Do not upload or share the private key. The exact command is supplied by the provisioned instance:

```
ssh -i ~/.ssh/<c3-key> <user>@<assigned-host>

# On the instance, verify the accelerator before installing or launching
work.
nvidia-smi
```

Before a real experiment, C3 still needs a documented image or bootstrap procedure, artifact collection, a spending ceiling, and verified teardown. Until those exist, it is available infrastructure rather than an operational pinglab backend.

Decision order

Use the smallest adequate option. Develop locally; orchestrate from Hetzner; use an idle Olorin GPU for opportunistic free work; submit planned production to Wilkes3 SL2; use RunPod for urgent overflow; choose Modal when managed reliability is worth the premium; leave SL3 for work that can wait; and treat C3 as an integration candidate.