

# Engineering Tips

Theodore Ehrenborg



# Disclaimers

- ▶ This talk is unofficial and doesn't represent views of my employers
- ▶ Conflict of interest: Last spring I did some cybersecurity work for Vast
- ▶ If AI gets more capable, the best way to use it will change
- ▶ Not a tutorial. I'll skip basic but important information

# Theme: A research engineer is an engineer specializing in experiments

## Experiments:

1. Take longer than running unit tests
2. Unclear ahead-of-time if they'll succeed
3. Important to run a lot of them
4. Important to run them correctly

# Table of contents

Vast.ai

Claude Code

Inspect

Miscellaneous

# Table of Contents

Vast.ai

Claude Code

Inspect

Miscellaneous

# General tips, part 1

Probably everything I say applies to Runpod, but I'm much less familiar with Runpod

- ▶ Mark a machine as broken so you don't get it again. Always choosing the cheapest machine is OK, except if you keep getting a broken machine, make sure you're not always getting the same one

The screenshot shows the Runpod interface for a machine with 1x RTX 4090. At the top, it displays 'Verified' status, IP address 195.139.71.84, a Norwegian flag, uptime of 2m 16s, and a cost of \$0.381/hr. Below this, a table lists hardware specifications: Instance ID (31506650), Host (47518), Machine ID (8321), Vol (No Volumes), Max CUDA (13.0), 83.0 TFLOPS, VRAM (0.5/24.0 GB), 879.2 GB/s, DLPerf (879.2 DLP/\$/hr), Network (100 ports, 854.1 Mbps up, 832.4 Mbps down), CPU (AMD EPYC 7542 32-Core, 32.0/64 CPU, 0/257.8 GB), Disk (Corsair MP600 PRO XT, 6261.0 MB/s, 0.0/256.0 GB), and Motherboard (H12SSL-NT, PCIE 4.0/16x, 25.0 GB/s). A row of icons includes '>\_Connect', a monitor, a trash can, a power button, a refresh icon, a cloud icon, a recycle icon, a 'LOG' button, a flag icon, a terminal icon, and a magnifying glass. At the bottom, it shows 'GPU: 0% 43°C, CPU: 0.33%, Status: success, running theodore' and a 'Report this machine' button.

1x RTX 4090

Verified 195.139.71.84 2m 16s 12d No Savings \$0.381/hr

Instance ID: 31506650	Max CUDA: 13.0	DLPerf	Network	CPU	Disk	Motherboard
Host: 47518	83.0 TFLOPS	-	100 ports	AMD EPYC 7542 32-Core...	Corsair MP600 PRO XT	H12SSL-NT
Machine ID: 8321	VRAM 0.5/24.0 GB	879.2 DLP/\$/hr	854.1 Mbps ↑	32.0/64 CPU	6261.0 MB/s	PCIE 4.0/16x
Vol: No Volumes	879.2 GB/s		832.4 Mbps ↓	0 / 257.8 GB	0.0 / 256.0 GB	25.0 GB/s

>\_Connect [Icons: Monitor, Trash, Power, Refresh, Cloud, Recycle, LOG, Flag, Terminal, Search]

GPU: 0% 43°C, CPU: 0.33%, Status: success, running theodore Report this machine

## General tips, part 2

- ▶ NVIDIA driver version is machine dependent, not image-dependent
- ▶ The Vast support team is pretty good.
- ▶ If you are auto-buying credits when your balance is low, be aware that sometimes it takes 5–10 minutes until the credits arrive.
  - ▶ Set the trigger balance high enough so that you don't hit zero (and hence all machines stop) in the meantime.
  - ▶ Probably not a huge issue unless you're running multiple H200s

# Script everything with the Vast.ai CLI, part 1

- ▶ This will notify you if there are any machines running:

```
running_count=$(uvx vastai show instances 2>/dev/null | grep -c "running")
if [ "$running_count" -gt 0 ]; then
    notify-send -t 0 "Vast.ai Alert" "$running_count instance(s) currently running"
fi
```

- ▶ Can be wrapped into a systemd job that runs every 2 hours (example)



## Script everything with the Vast.ai CLI, part 2

- ▶ Often you'll stop machines overnight when you're not running experiments.
- ▶ In the morning those machines may be unavailable, so it's useful to make setting up a new machine painless.
  - ▶ The Vast.ai CLI allows you to copy data off a stopped machine, but it's unreliable. So make sure all useful data is copied off the machine before you stop it.
- ▶ Demo of `https://github.com/TheodoreEhrenborg/vast-utils/blob/main/src/vast-utils/setup\_vast.py`

## Script everything with the Vast.ai CLI, part 3

Sometimes you have a lot of jobs to run, and one server isn't enough, e.g. a hyperparameter search. One approach is to treat servers as ephemeral:

1. spawn a pool of instances
2. install your code
3. run experiments
4. upload results to s3/W&B
5. destroy instances when the job queue is empty

Demo of [https://github.com/TheodoreEhrenborg/vast-utils/blob/main/src/vast-utils/ersatz\\_slurm.py](https://github.com/TheodoreEhrenborg/vast-utils/blob/main/src/vast-utils/ersatz_slurm.py)

# Table of Contents

Vast.ai

Claude Code

Inspect

Miscellaneous

# Background jobs

- ▶ Use case: Running a script over and over, editing the code until all bugs are eliminated
- ▶ Use case: Claude watches an overnight job and restarts it when there are errors you haven't anticipated
- ▶ Note Claude isn't 100% reliable, e.g. if you ask it to run many evals on many checkpoints, it will sometimes get the checkpoint paths wrong

# Give it free rein over a sandbox

- ▶ For local work,  
<https://code.claude.com/docs/en/sandboxing>
- ▶ For remote work (e.g. if it needs a large GPU), make an ephemeral server and give it to Claude
  - ▶ `"allow": ["Bash(ssh vast00:*)",]`
  - ▶ Note the Claude instance is local (so you have its logs), but it does everything over ssh
  - ▶ Often you'll want Claude to use tmux, see example prompt (link)

# Managing Claude

- ▶ Claude forgets everything every few hours
  - ▶ You'll be making the strategic decisions for the project on a scale of weeks
  - ▶ It's helpful to know everything Claude has done and how confident you are that Claude wrote each piece correctly
- ▶ Don't ask Claude to do impossible things or things you can't check
  - ▶ Claude is especially useful for throwaway scripts, plots, streamlit, etc

# Claude's docs are too verbose

- ▶ Telling it to write concisely like a human may help
- ▶ But in general it's worth writing public-facing docs by hand
  - ▶ Claude doesn't know what the important parts are
  - ▶ Some people don't like reading AI-generated text

## Useful permissions in `.claude/settings.json`

- ▶ `"env": {"BASH_DEFAULT_TIMEOUT_MS": "3600000" },`
- ▶ Allow git show, commit, push
- ▶ Allow ruff check, format
- ▶ Set a Notification hook for when Claude is done:  
<https://code.claude.com/docs/en/hooks>



# Table of Contents

Vast.ai

Claude Code

**Inspect**

Miscellaneous

- ▶ Read the logs
  - ▶ e.g. [https://github.com/UKGovernmentBEIS/inspect\\_ai/issues/2985](https://github.com/UKGovernmentBEIS/inspect_ai/issues/2985)
- ▶ Share via netlify or on self-hosted Apache server

- ▶ `https://github.com/Beneficial-AI-Foundation/dalek-lean-ai`

# Table of Contents

Vast.ai

Claude Code

Inspect

Miscellaneous

- ▶ W&B is less good for big files, but it does OK for 10–100 metrics
- ▶ For multi-person projects, W&B is nice for sharing, but for one-person projects, copying around TensorBoard logs works

## Get notified when job ends

- ▶ Get PID with `ps aux | grep python`
- ▶ `tail --pid 1234 -f /dev/null; pingme Done`
- ▶ Where `pingme` is a shell script that could be
  - ▶ `notify-send -t 0 "$1"`
  - ▶ `curl -d "$1" ntfy.sh/unique-id`

- ▶ <https://www.kolaayonrinde.com/blog/2025/01/01/shazeer-typing.html>
- ▶ Useful if you're debugging tensor operations by hand

- ▶ Take manual notes on the things of interest
  - ▶ Whereas s3 is where you dump everything so you can make graphs later
- ▶ Give each experiment an auto-generated timestamped unique ID, e.g. a template like  
`20250327-175110-sophisticated-pumpkin`