

BATTLE-TESTING SWARM

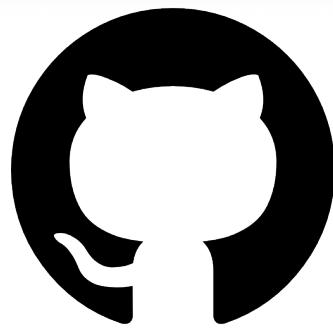
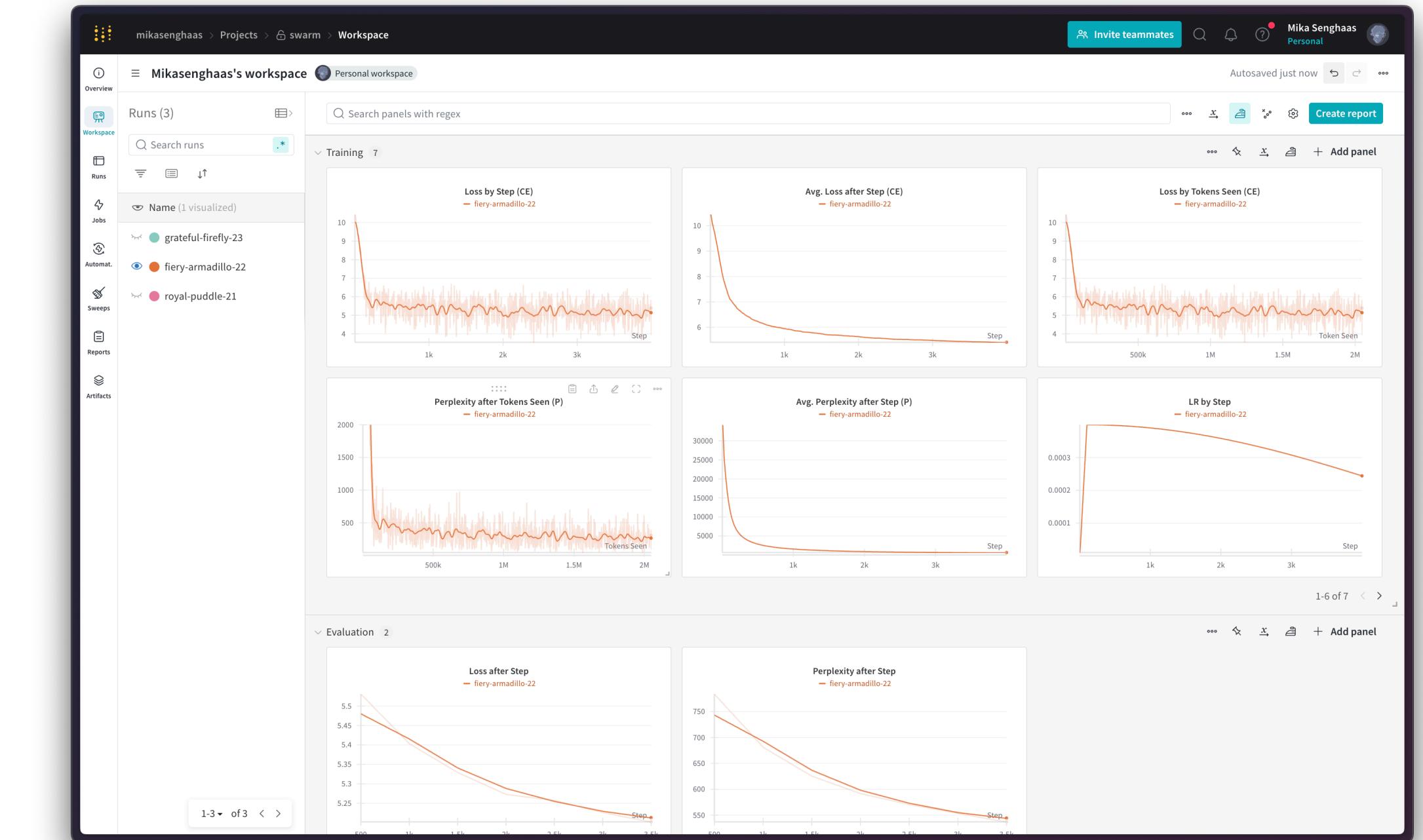
SETTING FOUNDATIONS

MIKA
SENGHAAS

EPFL

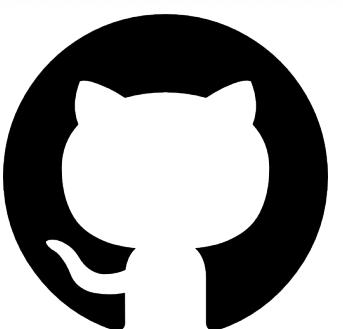
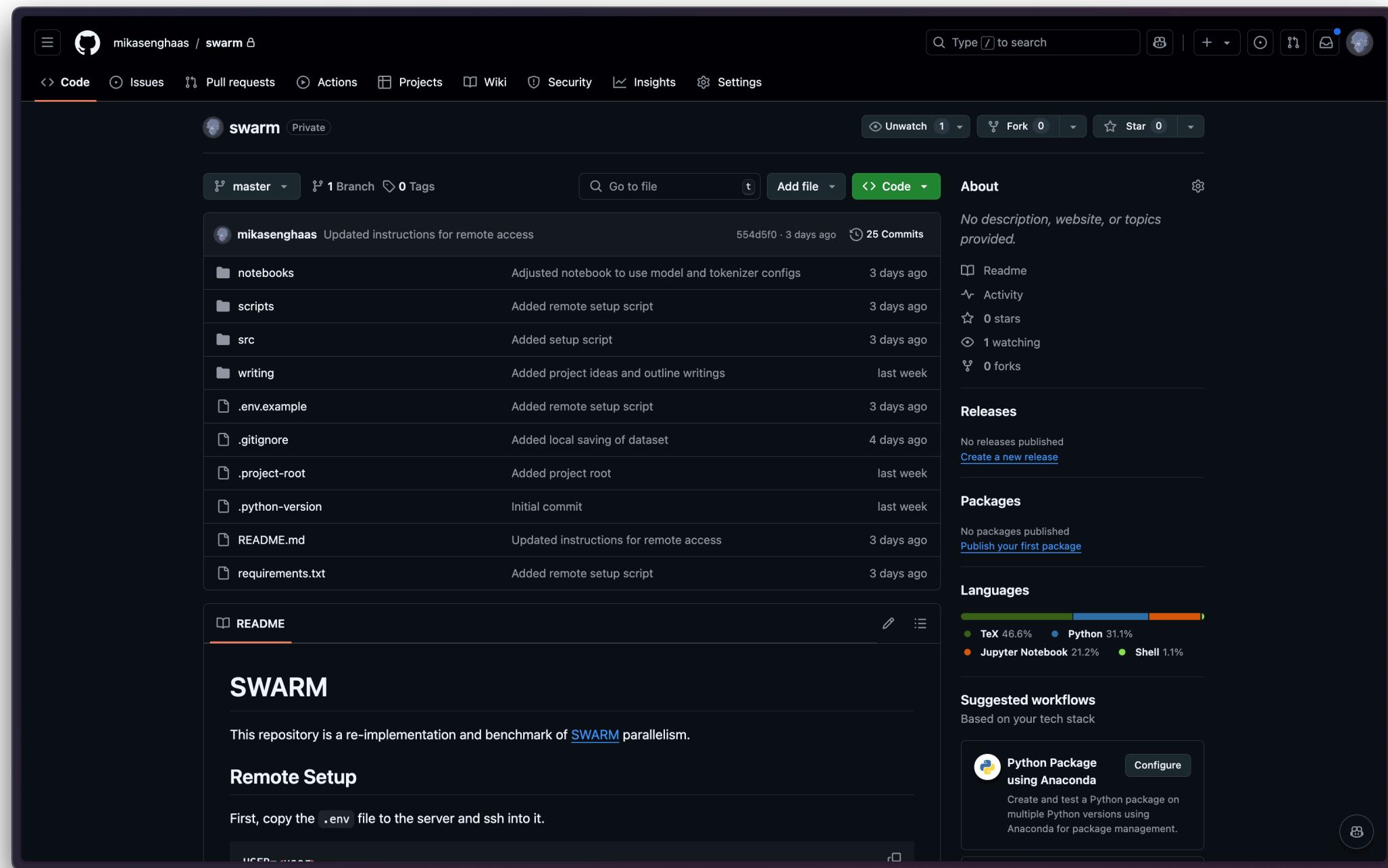
Implementing Single GPU

A screenshot of a GitHub repository page for 'swarm'. The repository is private and contains 25 commits across 1 branch. The README section includes a 'SWARM' heading, a note about being a re-implementation and benchmark of SWARM parallelism, and a 'Remote Setup' section with instructions to copy the .env file to a server and ssh into it. A 'Python Package using Anaconda' button is also present.



Weights & Biases

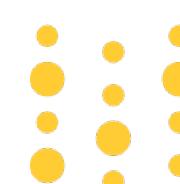
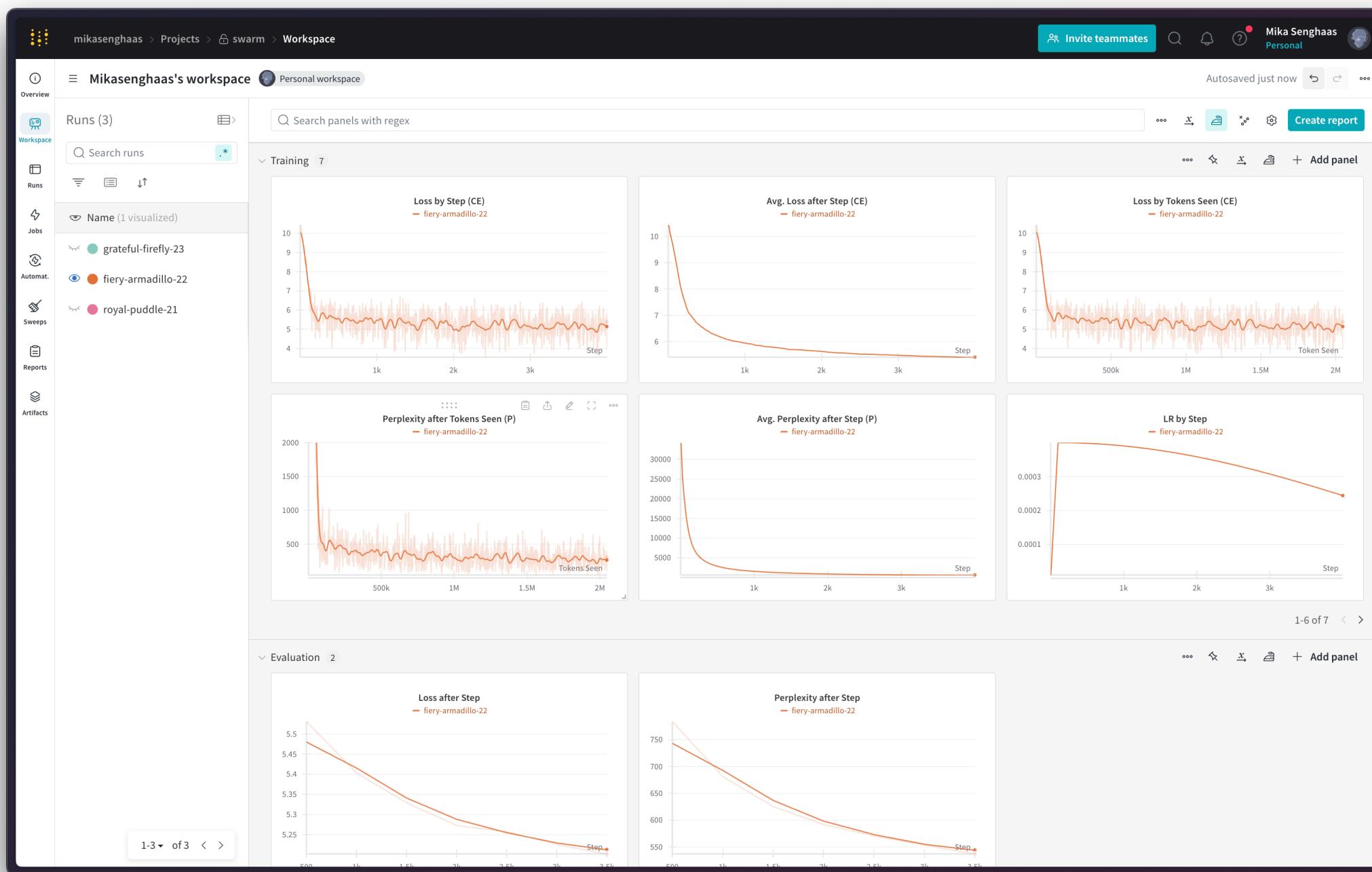
Implementing Single GPU



```
[ I ] ~/e/c/p/swarm master > git ls-files | grep '\.py' | xargs wc -l
    1 .python-version
    0 src/__init__.py
  98 src/config.py
100 src/logger.py
161 src/metrics.py
185 src/train/base.py
175 src/utils.py
720 total
```

- Minimal → 720 LOC / Pure Torch
- HF Data/Model (Dummy: Wikitext 2, Llama2 14M)
- Console/W+B Logging
- Advanced config (Pydantic)
- Cosine LR scheduler
- Mixed precision training (on CUDA)
- Gradient Accumulation
- Custom Metrics (+ Logging)
- Easy remote access → Integrates w/ PL Compute

Implementing Single GPU



Weights & Biases

Metrics

- Step
- CE Loss
- Perplexity
- #Tokens (#Examples)
- LR
- Time

Also evaluable

+Throughput (T/s)

Computed

+ W&B Perf. Metrics (GPU Mem./Utilisation/Temp. ...)

Next Steps

① Verify 1-GPU Baseline

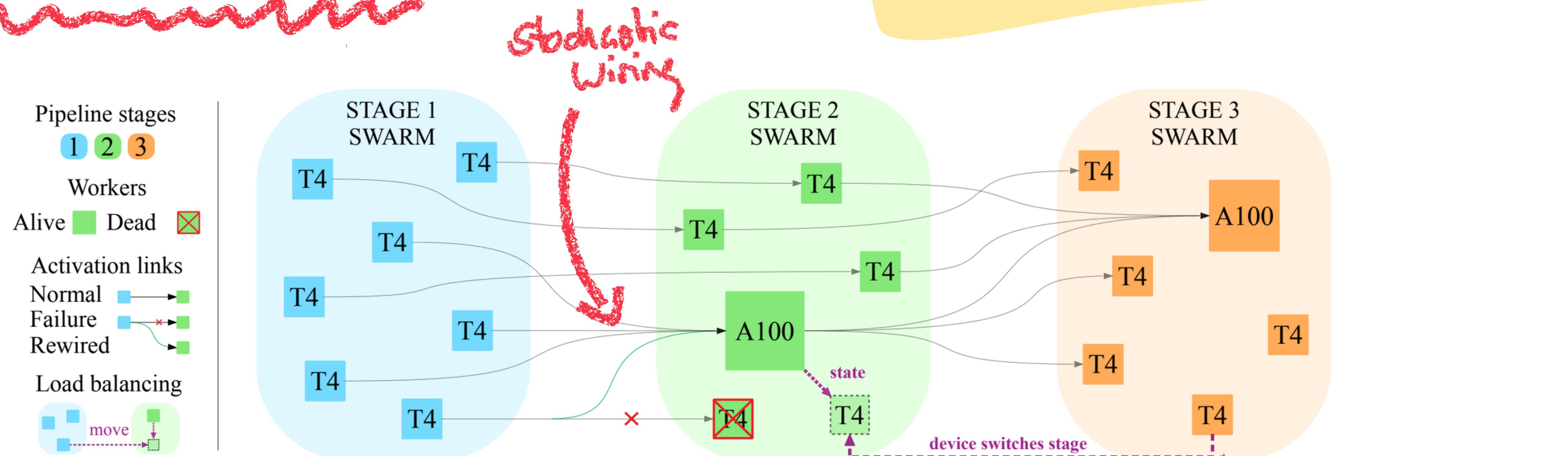
NanoGPT setup
→ (GPT-2 (140m) on
Fine webEdv on
AOC)

② Implement traditional PP → Minimal (20xx)
PP Gist

③ Implement SWARM → Yandex SWARM +
Pi Zero Band

References

SWARM



DYNAMIC + REDUNDANT PP

Decentralised Setting

① Heterogeneous Hardware

② Heterogeneous Network

③ Reliability

④ Scale

↳ Infini-k Model
Data Scale

↳ On-Off
Ramping

