Quali | TORQUE

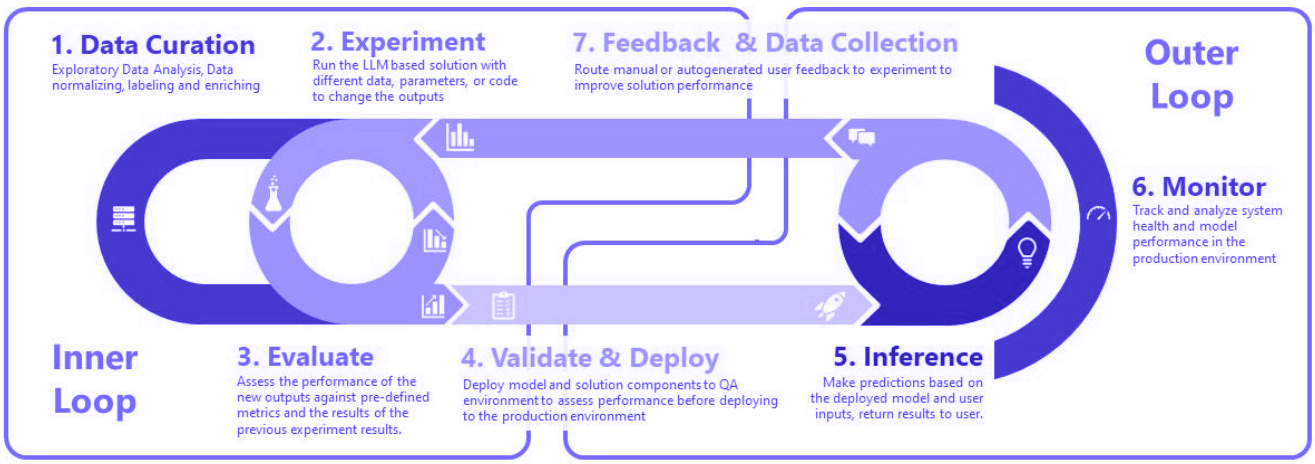
The AI Supercomputer: from your desk to the private dev cloud

Model fine-tuning as a Service with Torque and NVIDIA DGX™ Spark

Immutable infrastructure and containers changed everything about how we ship code. Developers got used to fast feedback loops, spinning up an environment in seconds, test their changes instantly, tear it down and iterate. That velocity became not only an expected baseline, but also made possible faster product feedback loops and business value delivery.

When it comes to resource intensive LLMOPs operations, the world seems to shift back to a pre-cloud era of scarcity. The requirements for expensive GPUs and NPUs along with fast unified RAM and IO are even more relevant to the development and experimentation phases of model development than production inference.

Faced with mounting exponential cloud costs, many companies are seeking a path for scaling LLMOps operations that would be both viable from a cost perspective and avoid holding back engineering efforts. Short of that coveted middle ground, the release process, is doomed to become choppy at best, held back and at times grinding to a halt, not because the code isn't ready, but because the infrastructure to validate it is locked up, idle, or hopelessly oversubscribed.



Source: https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/mlops-in-openai/

Enter the NVIDIA DGX Spark

The new NVIDIA DGX Spark is an appliance that aims to unclog that very bottleneck. At around 4K USD it is economical enough to allow engineers uninterrupted access to a resource capable of supporting experimentation in model fine tuning and inference with tens of billions of parameters. These numbers make all of the difference between having a scalable LLMOps process and one blocked by cost constraints. Alternative cloud solutions, such as AWS Bedrock, could cost over 100K for a model pipeline just based on provisioning, storage and resource costs alone.

Having such an appliance on your desk is definitely every engineer's dream, allowing to test inference behavior locally, and validate end-to-end pipelines without the latency or cost of cloud GPUs. But it's important not to oversell: these workstations also come with constraints. They are not a silver bullet, but rather a powerful dev level accelerator.

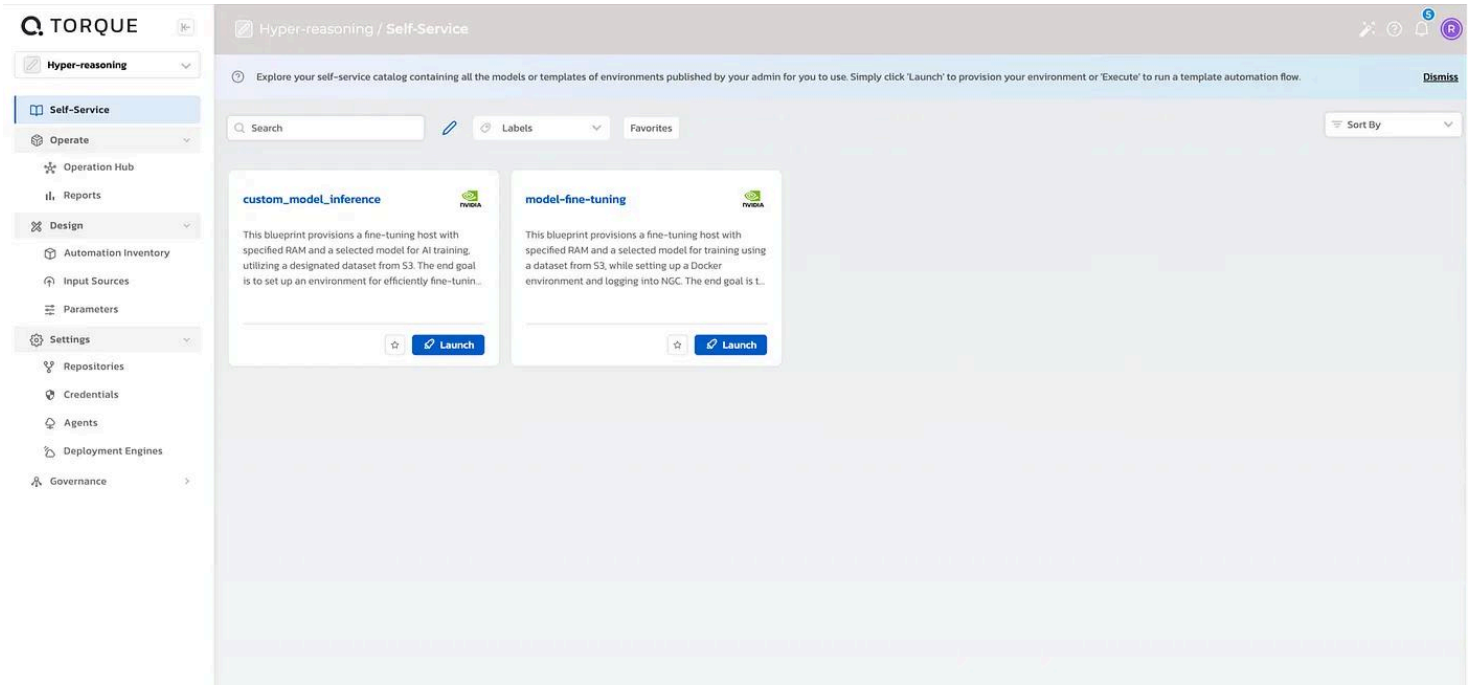
We've been testing out a sample unit of this new NVIDIA product for almost a month and can certainly testify its positive effect greasing our own LLMOps pipeline, however it was also immediately apparent that using this box to solve localized single-engineer problems misses out on its true potential.

The real question was: how do we make those workstations part of something bigger, a shared, scalable, managed fabric rather than isolated islands of super-computer-productivity? Working as a part of an engineering team, it doesn't make any sense to have this (literally) golden box sitting dark and unused overnight or on weekends, when our distributed team is starved for similar resources.

It soon became clear: the problem wasn't just resourcing. It was coordination, governance, and repeatability. We needed a system that allowed experimentation to scale without collapsing into chaos.

From Individual Developers to Teams: How to Carry Scalable LLM Development on a Budget

We love dogfooding at Quali, and our main product line Torque, seemed to be a great fit for the challenge of cloudifying the DGX Spark access. Transforming it from a localized resource and into a fabric, with self-service access and governance built in.



As we began envisioning what this project would look like, it was very clear for us what success would look like:

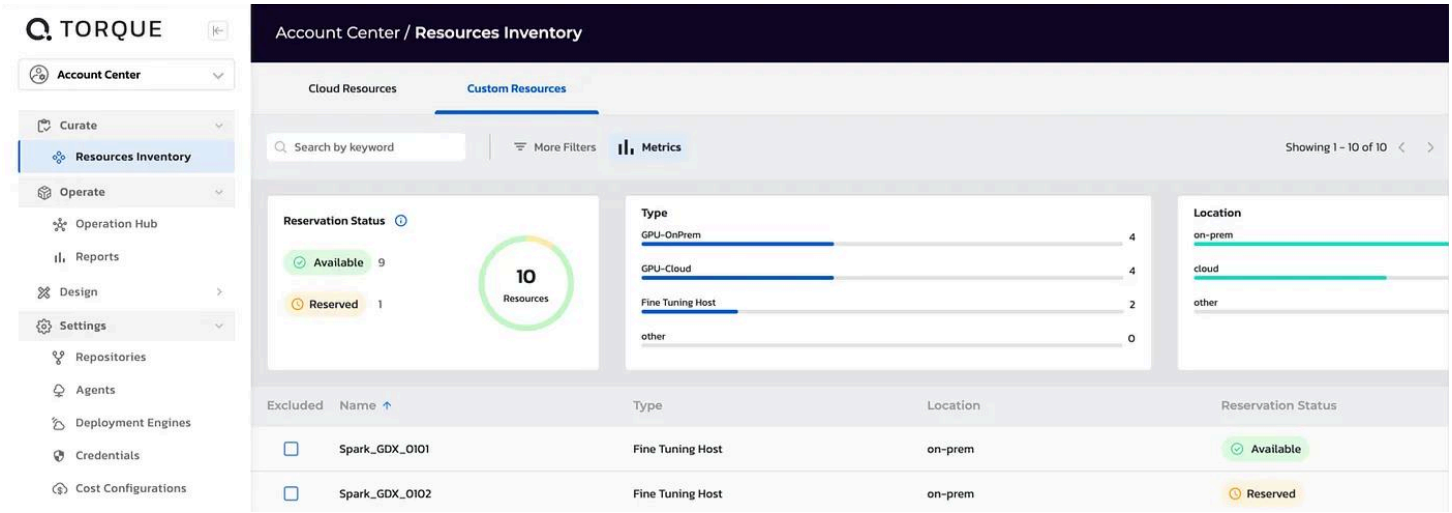
- **Developer-first self-service:** Engineers should be able to request fine-tuning environments without hopping into Slack or emailing ops teams.
- **Policy-guarded access:** We needed governance - quotas, scheduling limits, access controls, so nobody hogs the fleet.
- **Reproducibility and consistency:** The same blueprint used in local dev, CI, or staging should produce essentially identical environments.
- **Smart utilization:** Idle hardware (overnight, weekends) should be auto-harvested rather than wasted.
- **Cost elasticity:** When demand exceeds internal capacity, spill over seamlessly to external cloud resources.

In pursuing those goals, we realized the architecture we needed looked a lot like internal “infrastructure as a service,” but for AI and inclusive of discrete resources such as the Spark box.

The Super Computer as shared AI Infrastructure

As we were considering where Environment-as-a-Service concepts could apply in the LLM workflow, we realized the same principles that make cloud infra self-serve, inventory, templating, scheduling, governance, also apply to on-prem and workstation-level GPUs. Torque became our mechanism for that translation.

Thankfully, in addition to IAC assets such as Helm chart or Terraform files and native support for running configuration scripts, Torque also includes a resource inventory module. This feature allows the user to define a given set of discrete resources (anything from connection pools to DGX Spark) and then allow access control, sharing and scheduling.



Creating the Model Fine-Tuning blueprint

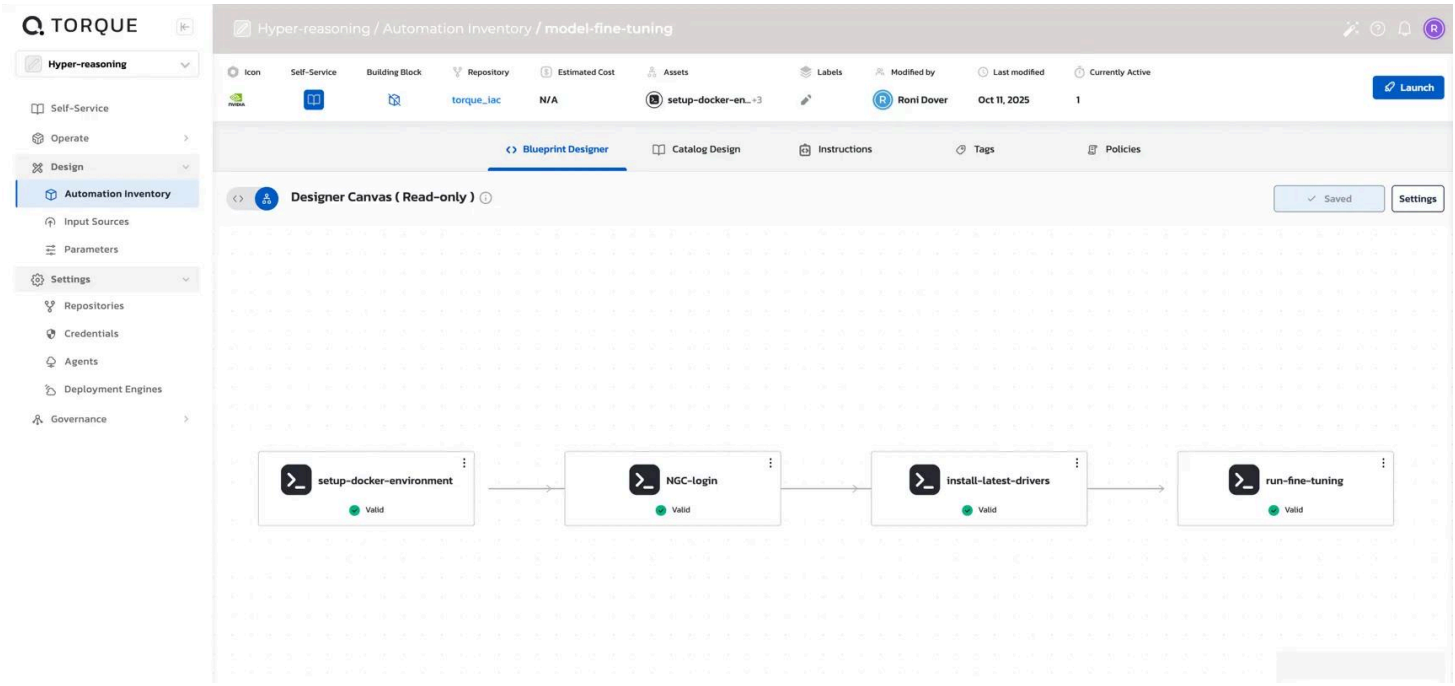
We wanted to create a flow whereby the developer selects the base model for fine-tuning and provides a link to the data set, then simply launches the blueprint, either manually or as a part of the git flow. Behind the scenes, we wanted the system to allocate a DGX Spark or queue the request up if one is not immediately available.

With Torque, all blueprints are defined as simple YAML files, sourced within the GH repositories or kept on the Torque server. We can very quickly set one up for this use case.

```yaml
spec_version: 2
inputs:
  ram_size:
    type: string
    allowed-values:
    - 64
    - 128
    description: |
      64GiB: Suitable for lightweight applications or testing
      128GiB: Common for moderate workloads or production databases

  base_model:
    type: string
    allowed-values:
    - NVIDIA-Nemotron-Nano-9B-v2
    - Llama 3.1 Nemotron Nano VL 8B v1
    - gpt-oss-20b
    description: |
      NVIDIA-Nemotron-Nano-9B-v2:  high-efficiency LLM with a hybrid Transformer-Mamba design, excelling in
      reasoning and agentic tasks. It can generate reasoning traces before providing a final response.
      Llama 3.1 Nemotron Nano VL 8B v1: A vision-language model, part of the Llama Nemotron family,
      optimized for real-time applications and suitable for deployment on PCs and edge devices.
      gpt-oss-20b: Smaller Mixture of Experts (MoE) text-only LLM for efficient AI reasoning and math

  training_file_s3_arn:
    type: string
    default: arn:aws:s3:::acme_de_latest_ds/ds_09_25_25.jsonl
    description: |
      Data set for fine tuning the model

resources:
  fine_tuning_host:
    selector:
      type: Fine Tuning Host
      quantity: 1
      attributes:
        - RAM: '{{ .inputs.ram_size }}'
```

The next stage is making sure our environment is set up properly. This is where DevOps meets ML: installing the correct CUDA drivers, PyTorch versions, NVIDIA NeMo software release, and dependencies for the tokenizer or optimizer.

In theory, this process should be reproducible, but in practice, it's brittle. A local run on CUDA 12.2 may behave slightly differently than a shared GPU node on 12.1. While containerization helps, NVIDIA provides NeMo Docker images on NVIDIA NGC™ with most dependencies pre-installed, but even then, mismatches in data mounts, authentication tokens, or Python patch versions can derail a run.

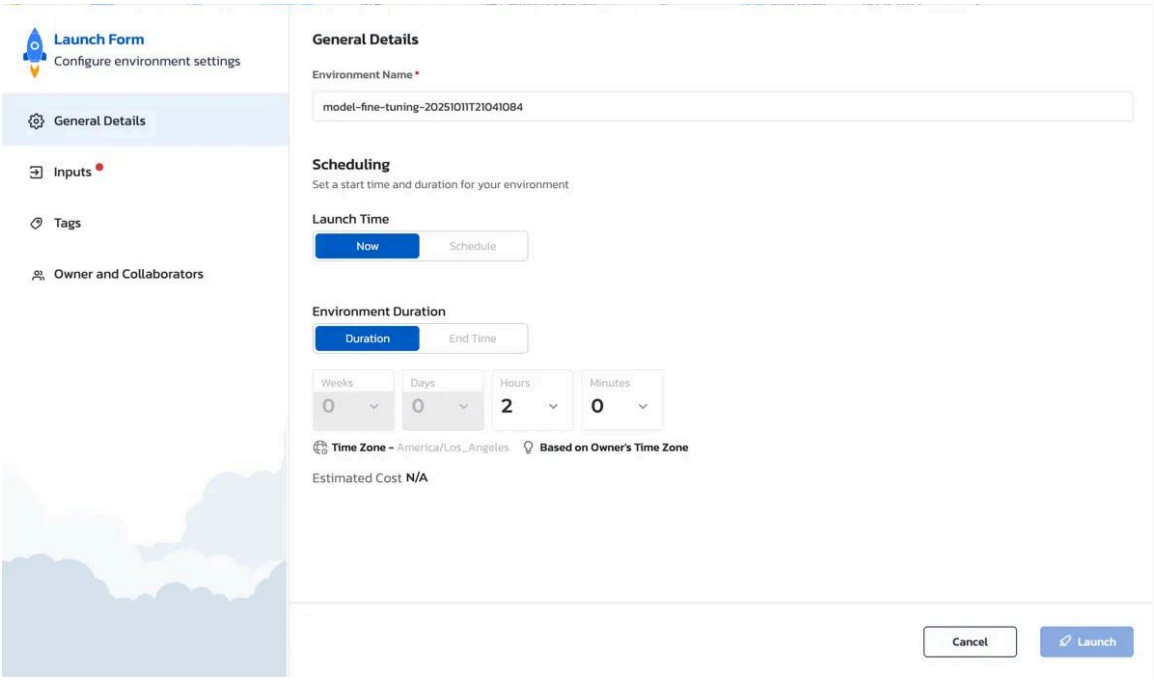


Once the fine-tuning run completes, the output is evaluated against test sets or downstream benchmarks. In the NeMo framework, you might run evaluate.py with the same YAML config to generate accuracy, perplexity, or BLEU scores, and log the metrics into your experiment manager.

If the model meets its goals, it's exported as a .nemo file, a self-contained package including weights, tokenizer, and configuration. From there, the file is uploaded to an inference endpoint or served through NVIDIA Triton Inference Server or TensorRT-LLM.

Overcoming the Hidden Tax of Discontinuous GPU Access

As a last step, we can set additional rules about the DGX Spark usage. Constrain the number of hours/days for which it can be reserved by engineers and enforce policies and quotas per project.



DGX Spark aims to bring supercomputer-class AI capabilities to the desk of every data scientist. Quali Torque transforms those individual workstations into a shared, on-demand cloud of fine-tuning resources.

Together, they solve a fundamental challenge in AI development: bridging the gap between powerful desktop prototyping and realistic, collaborative testing environments and fast development cycles. By providing on-demand access to production-class hardware, organizations maximize ROI on expensive AI infrastructure. Data scientists wake up each morning to a clean workstation, while their off-hours capacity powers the team's rapid progress.

In summary, NVIDIA DGX Spark gives engineers a reliable, desk-level accelerator for model fine-tuning and inference. Paired with orchestration platforms like Torque, it bridges the gap between local experimentation and scalable cloud deployment, without bottlenecking progress or budgets.