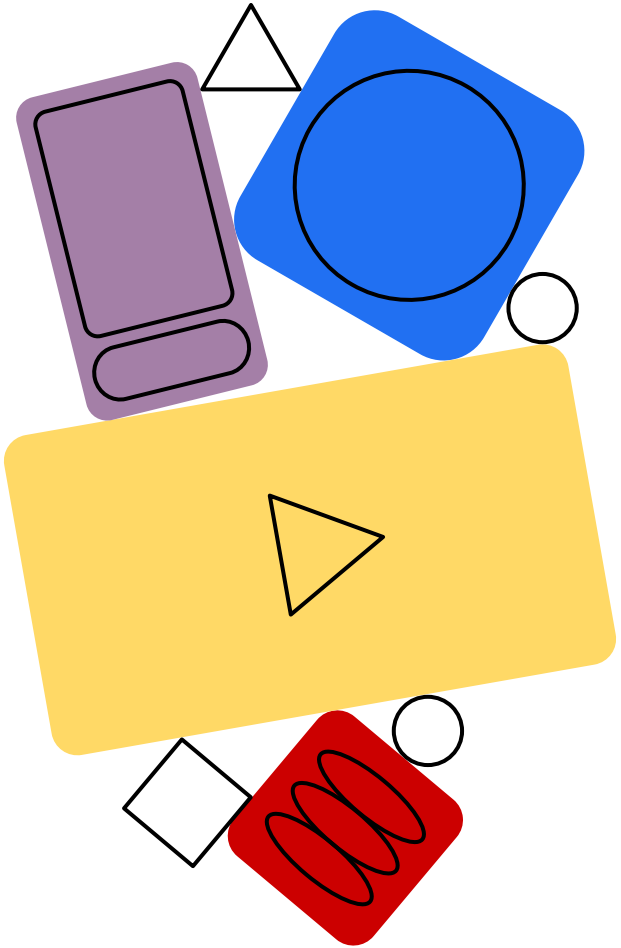
A Study of Optimization Techniques for Scalable Image Classification Inference on the Cloud



Timothy Chang
Tyler Chang
Athitheya Gobinathan
Pablo Ordorica-Wiener

Presentation Guidelines

- **Project Overview** – Clearly describe the project's purpose, goals, and significance.
- **Course Relevance** – Include a dedicated slide explaining how the project relates to at least two of the three course core topics (Cloud, DNN, Performance Analysis). Also, outline task distribution among team members.
- **Related Work** – Discuss prior research or existing solutions that informed your project.
- **Design & Implementation** – Explain how your system works, including architecture, methods, and technologies used.
- **Live Demo** *(strongly encouraged)* – Showcase your system in action.
- **Evaluation** – Present performance metrics, results, and key findings.
- **Discussion** – Reflect on challenges faced, lessons learned, and any unexpected outcomes.
- **Conclusion & Future Work** – Summarize key takeaways and propose potential next steps.

Project Overview

Goal

- Evaluate how different post-training optimization strategies impact end-user experience
- Gain practical experience with self-managed AI inference services,
    - building on our previous work deploying models with Kubeflow on GCP.

What we did

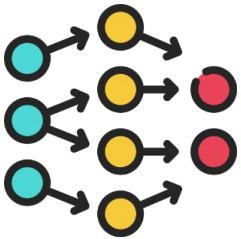
- We developed four web endpoints on Google Cloud Vertex AI, each serving the same image classification model but with distinct post-training optimizations and one baseline without optimization.
- To facilitate interaction and benchmarking, we built a web app that sends requests to these endpoints.

Significance

- Insights into the trade-offs between model performance and user experience,
- Inform on challenges for deploying optimized AI models in real-world cloud environments.

Course Relevance

- Implemented our ML Platform on the Google Cloud Platform



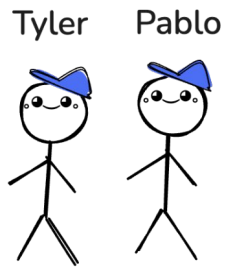
- Performed post-training optimizations on neural network models

- Analyzed the impact of the optimizations end-user experience

Related Work

- ML Platform Related
  - Homework 3 - Simple DL Workflow in Kubeflow
  - Cole Bailey: Cooking up a ML Platform - Growing pains and lessons learned | PyData YouTube
  - NVIDIA Triton Inference Server and its use in Netflix's Model Scoring Service | Outerbounds YouTube
- Neural Network Optimizations
  - Class Presentation - NN Optimization Frameworks: Torch2Compile, TensorRT, Faster Transformer
- AI Inference
  - How to Put AI Models into Production: A Guide to Accelerated Inference | NVIDIA Book
  - 2024 State of AI Inference Infrastructure Survey Report by BentoML

Task Assignments



Platform Team

Clearly, more blue boxes means more work.

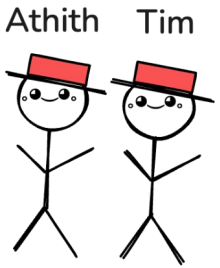
*

ML Dev

ML Deploy

Web App

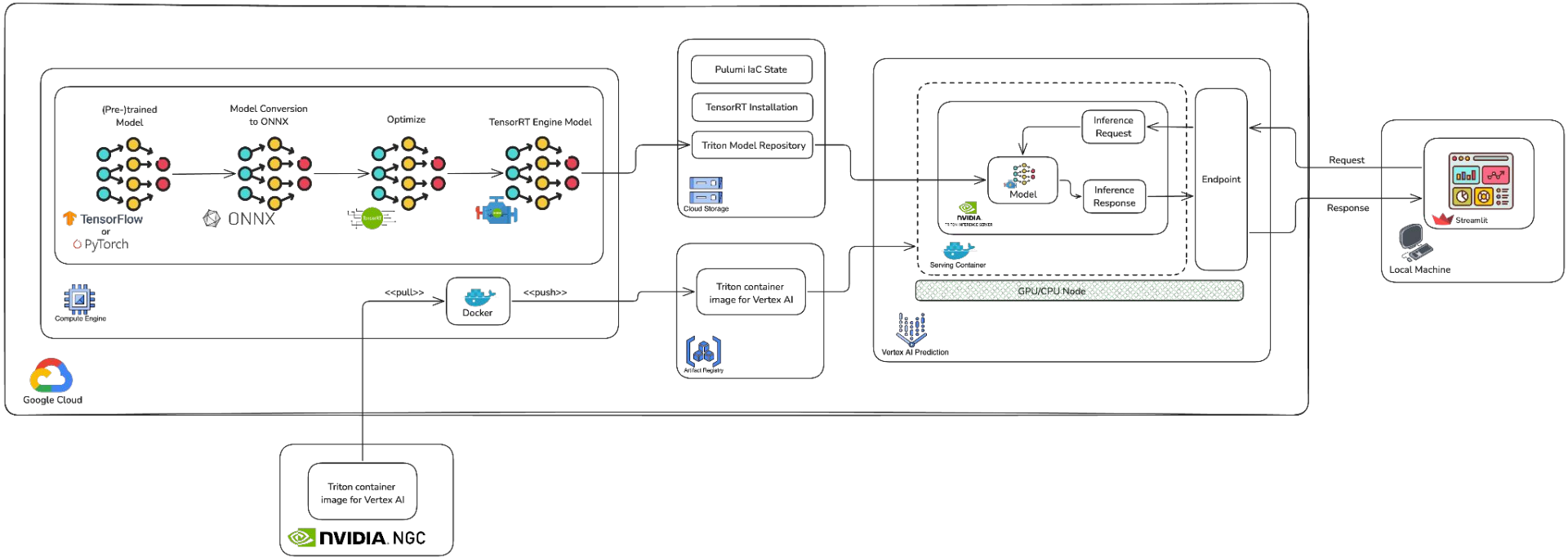
ML Platform



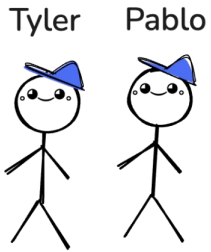
ML Team

*This slide was made by a member of the platform team.

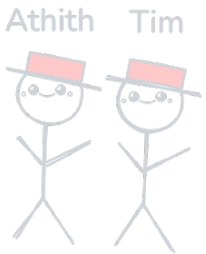
Design & Implementation



Design & Implementation



Platform Team

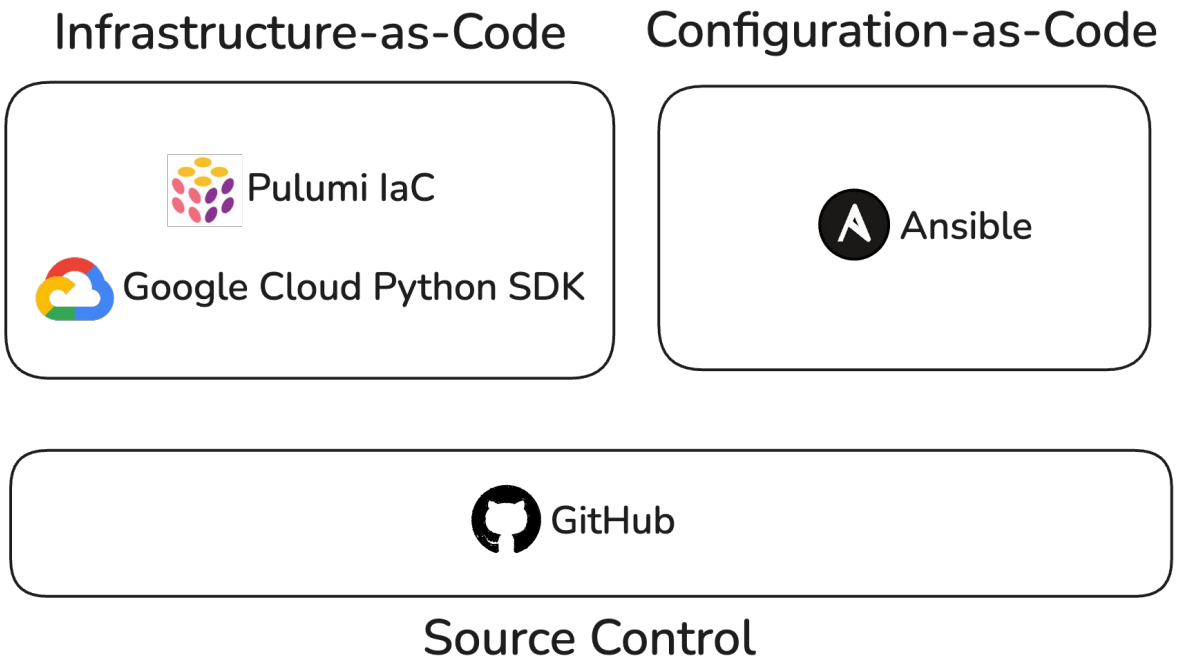


ML Team

ML Dev | ML Deploy | Web App

ML Platform

Design & Implementation



Quick Aside: Imperative IaC

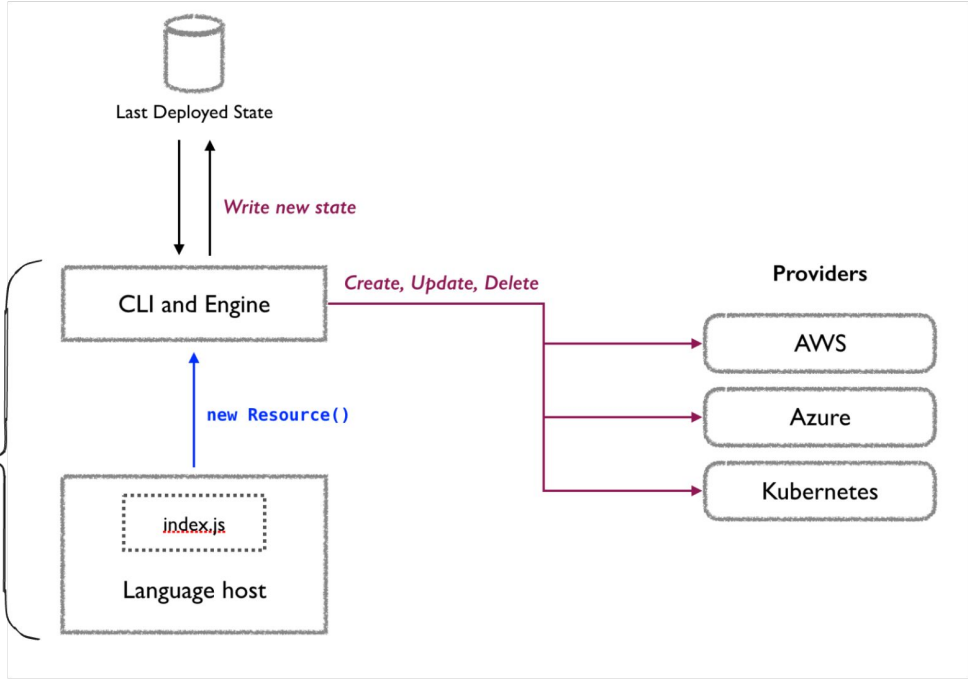
1. Define the exact steps needed to achieve a desired infrastructure state

Quick Aside: Declarative IaC

1. Declare desired state
2. State = record of infrastructure managed by IaC tool
3. Backend where state is stored
4. Stack = group of infra resources managed as a unit

Pulumi IaC

Terraform



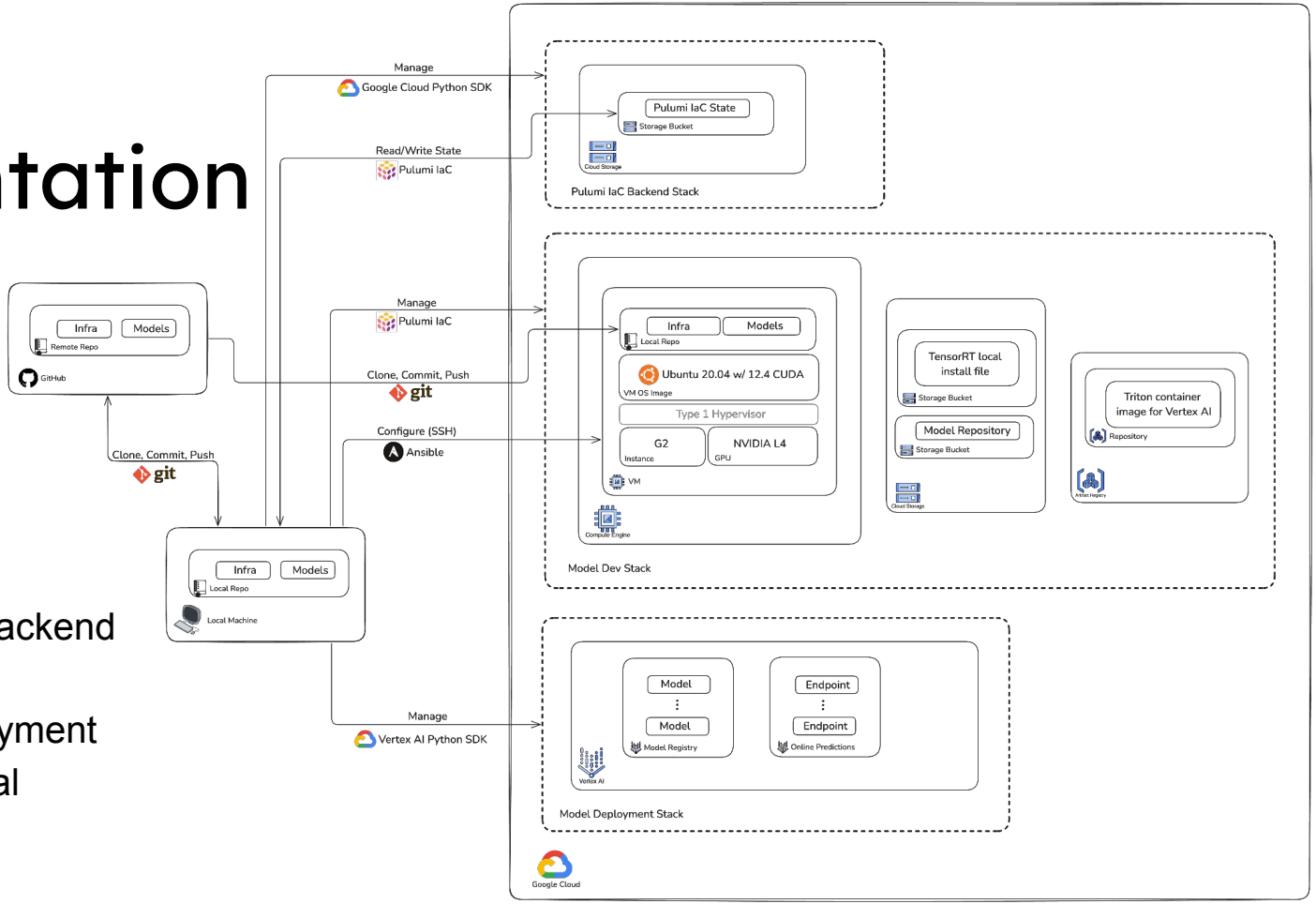
Chapter 5. Building Infrastructure Stacks as Code | O'Reilly Book by Kief Morris

How Pulumi Works | Docs Pulumi IaC Concepts

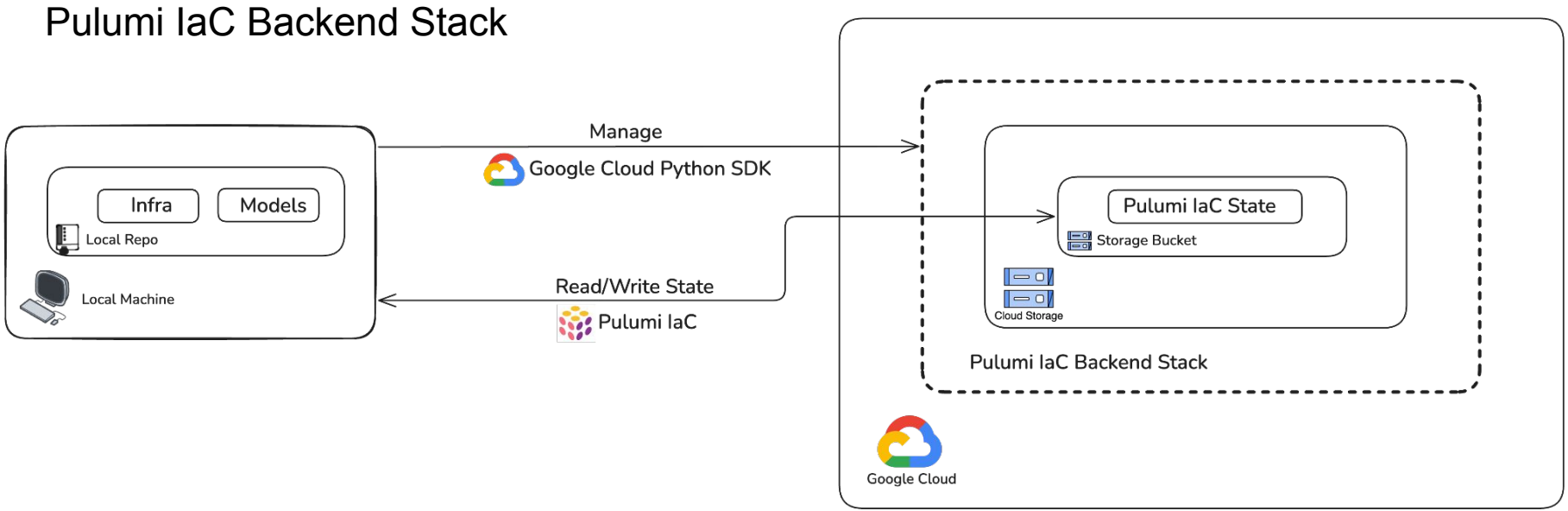
Design & Implementation

ML Platform

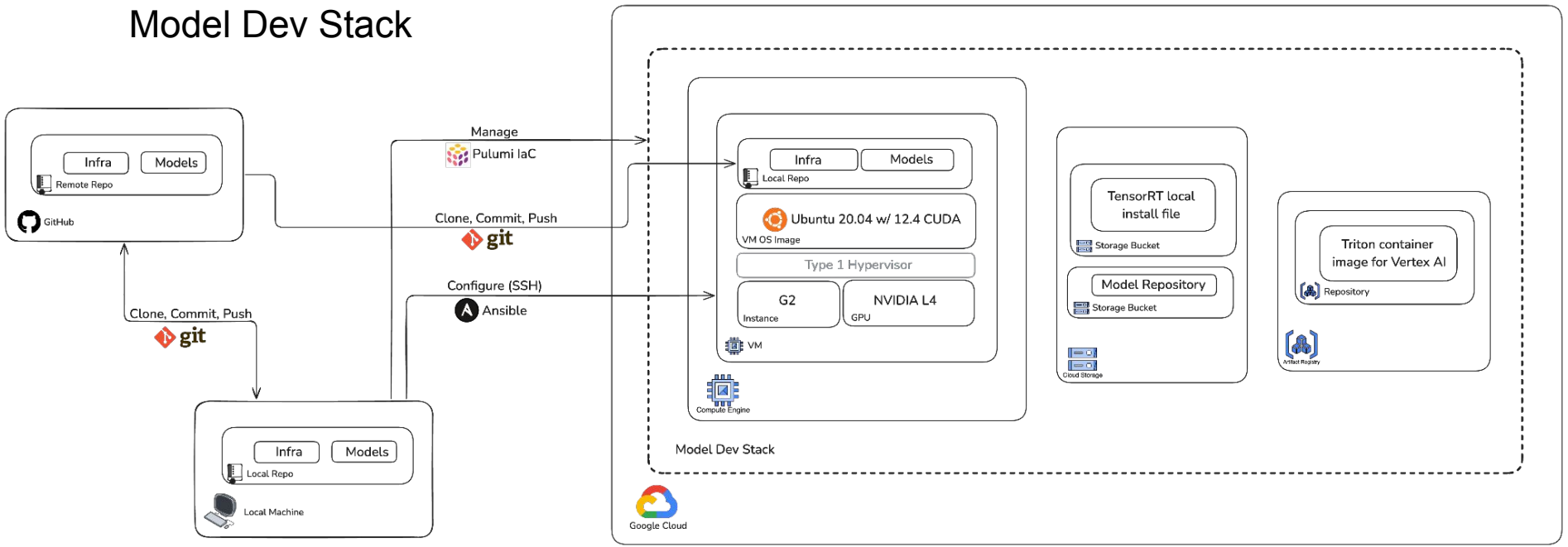
- 3 stacks
  - Pulumi Iac Backend
  - Model Dev
  - Model Deployment
- deployed from local



Design & Implementation



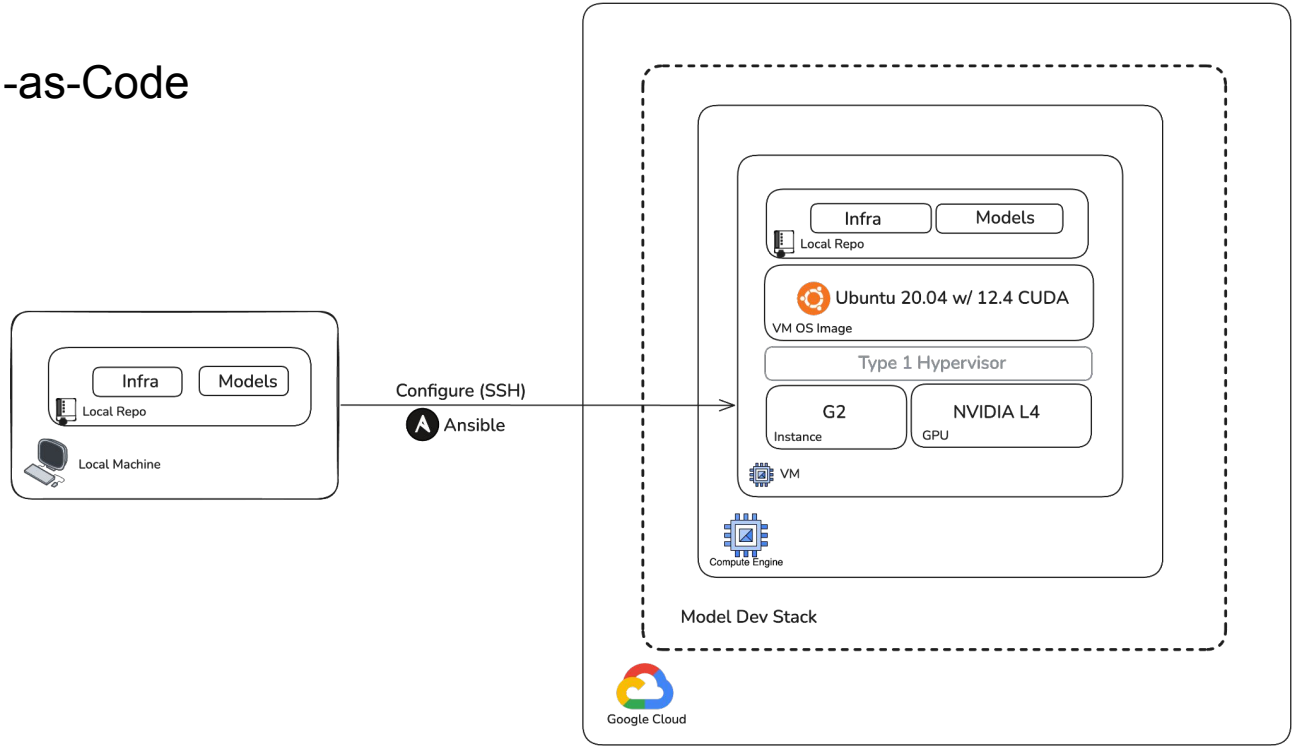
Design & Implementation



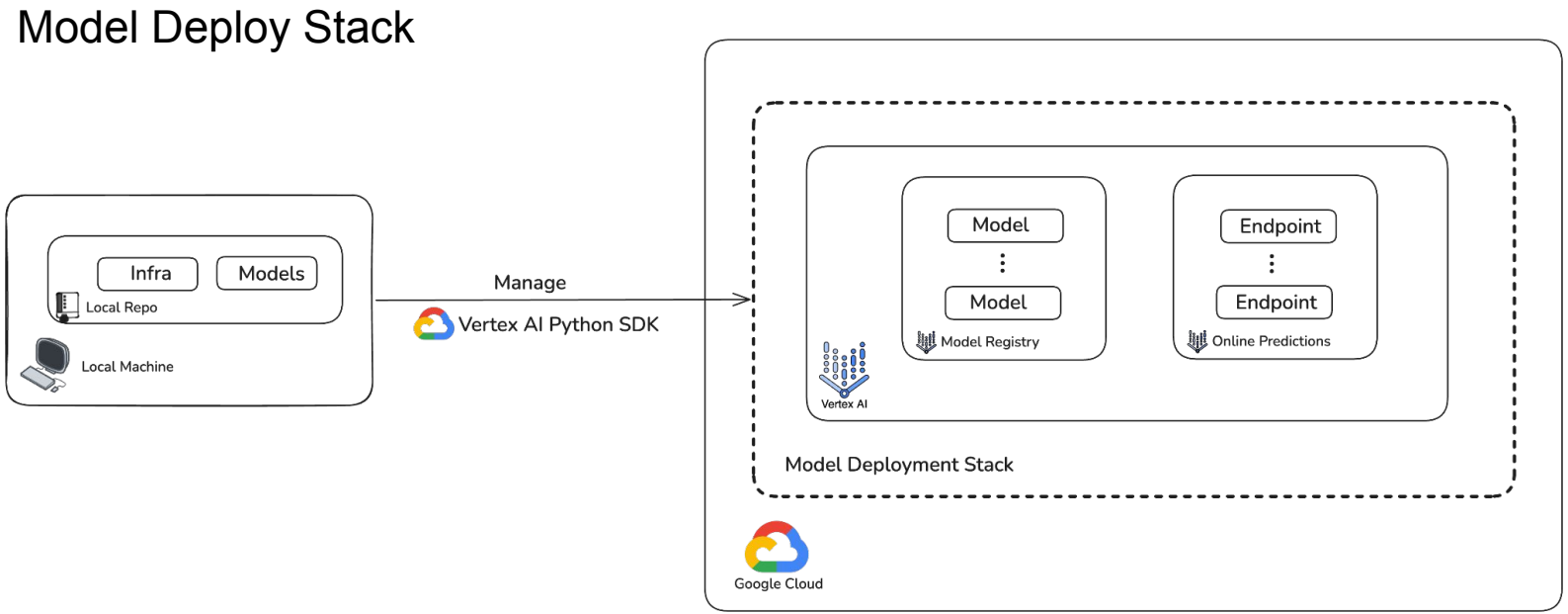
Design & Implementation

GPU VM Configuration-as-Code

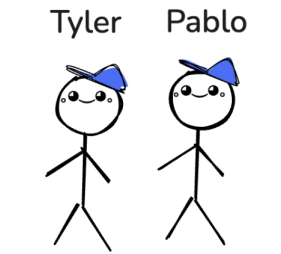
- Install
  - uv - Python dependency manager
  - TensorRT
- Configure Docker

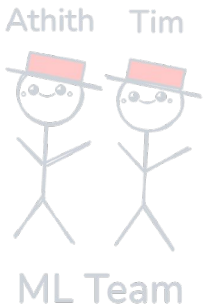


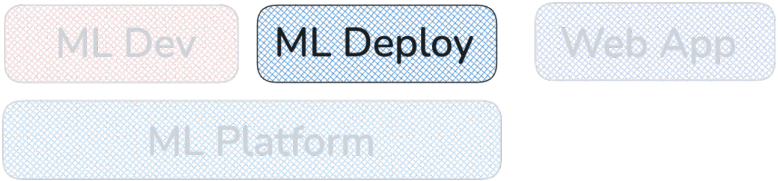
Design & Implementation



Design & Implementation







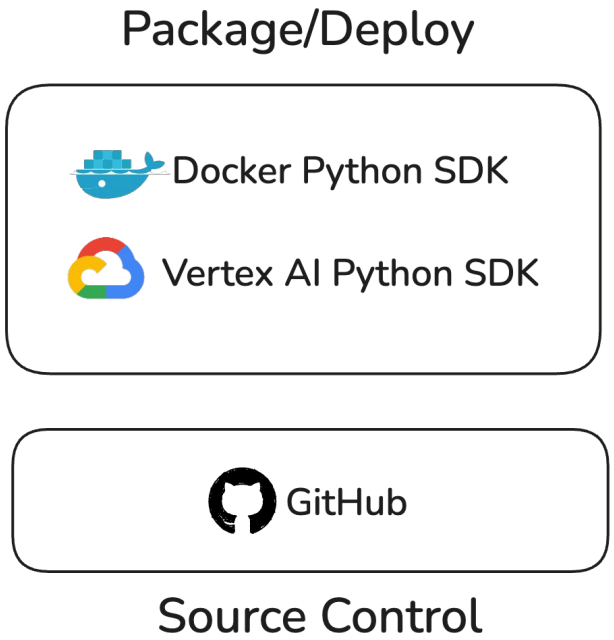
Design & Implementation

- Fully managed infrastructure specifically designed for ML workloads
- Connects with the model repository, GPU/CPU nodes, and Triton serving container
- Automatically provisions the specialized computing resources needed for inference while abstracting away the underlying infrastructure complexity

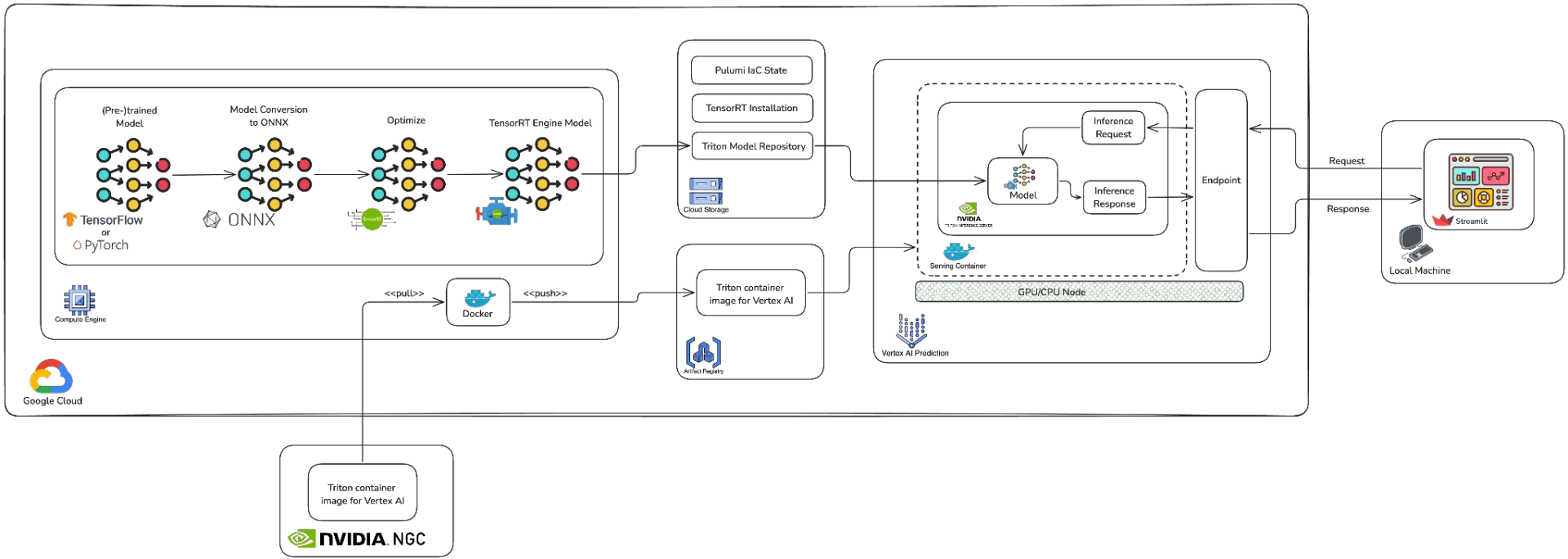
Design & Implementation

- Specialized inference engine handling the actual model serving and inference requests with optimizations specifically for NVIDIA hardware
- Enables deployment of models from different frameworks (TensorFlow, PyTorch, ONNX) through a unified serving interface, streamlining operations regardless of the original training framework
- Enable efficient GPU utilization during inference, automatically scaling resources based on model demands and optimizing performance for the specific hardware configuration

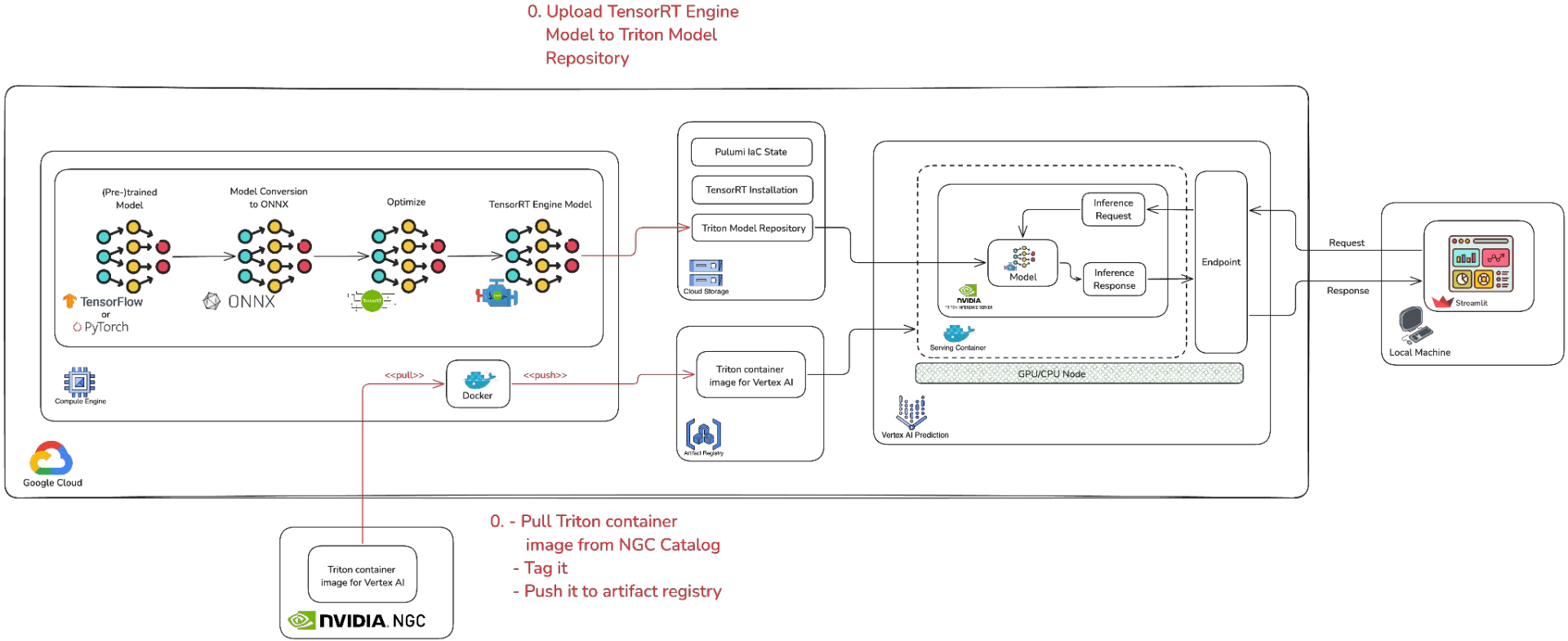
Design & Implementation



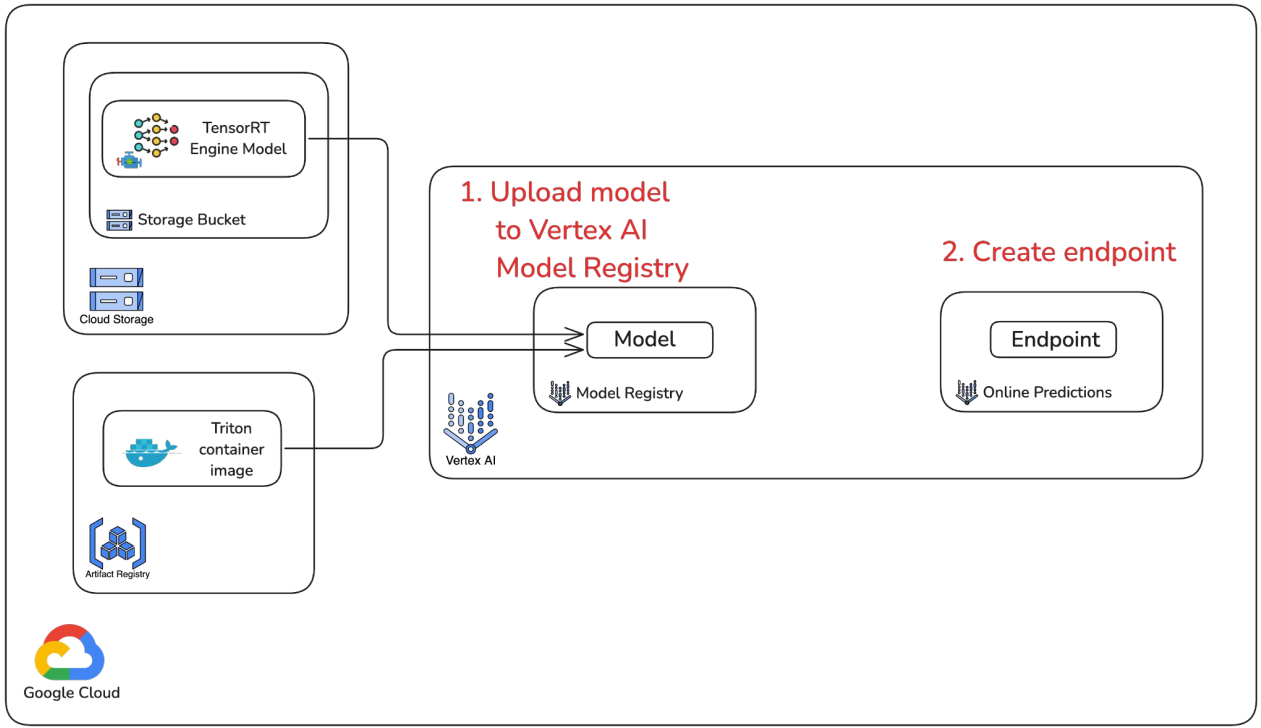
Design & Implementation



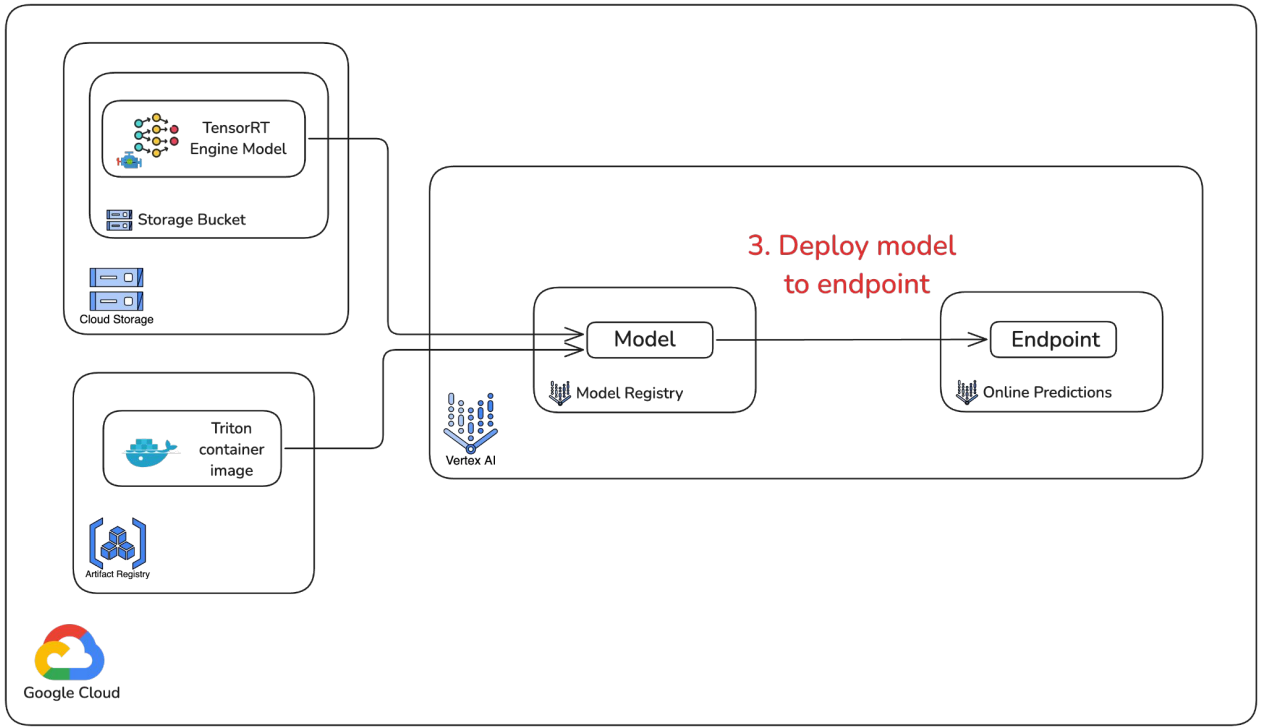
Design & Implementation



Design & Implementation

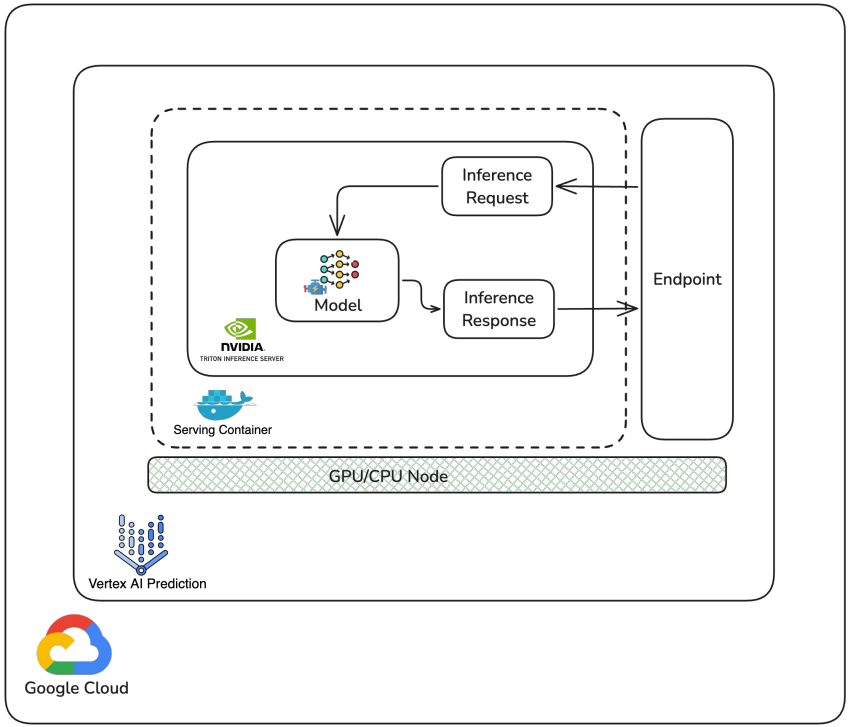


Design & Implementation

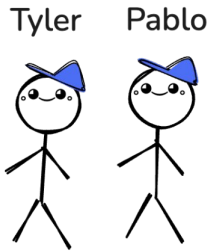


Design & Implementation

Result of a single deployment



Design & Implementation



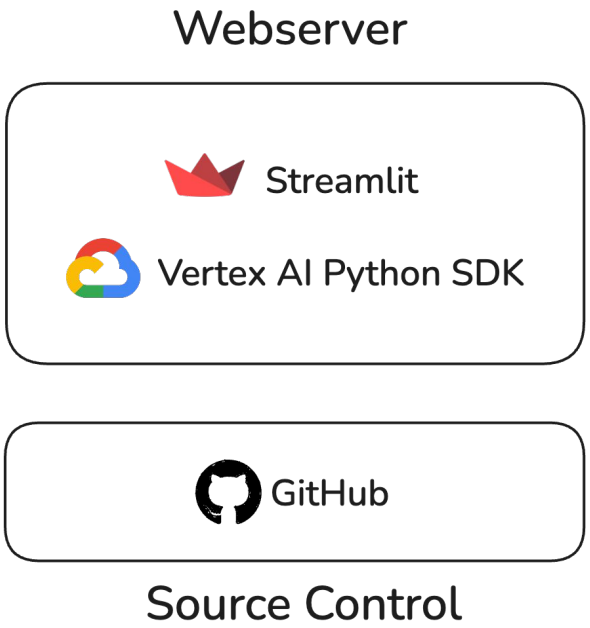
Platform Team

ML Team

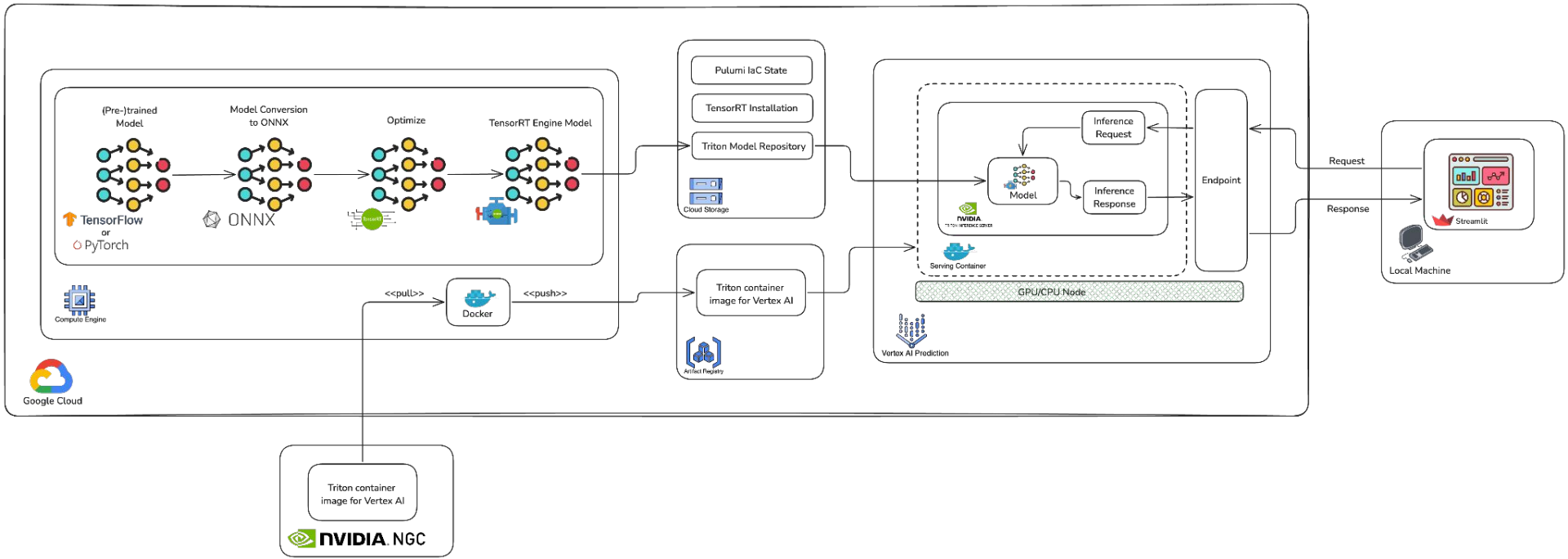
ML Dev | ML Deploy | Web App

ML Platform

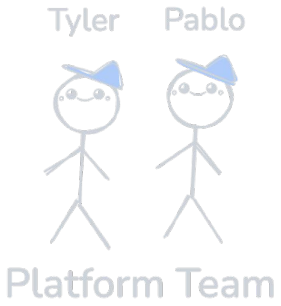
Design & Implementation

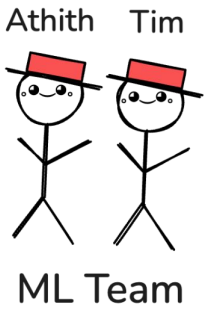


Design & Implementation



Design & Implementation





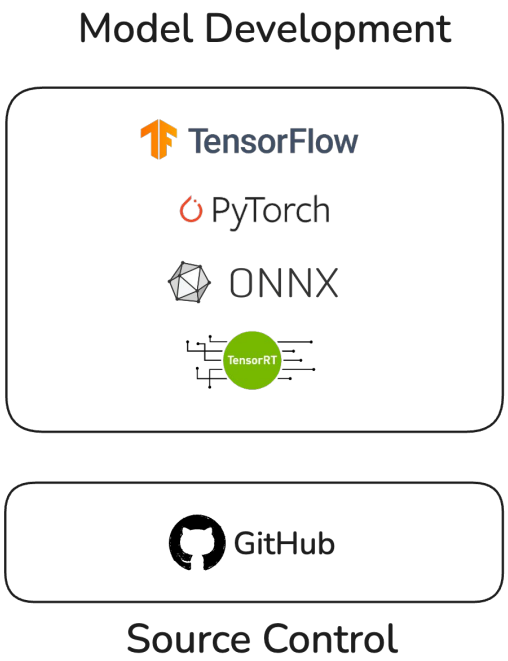
ML Dev

ML Deploy

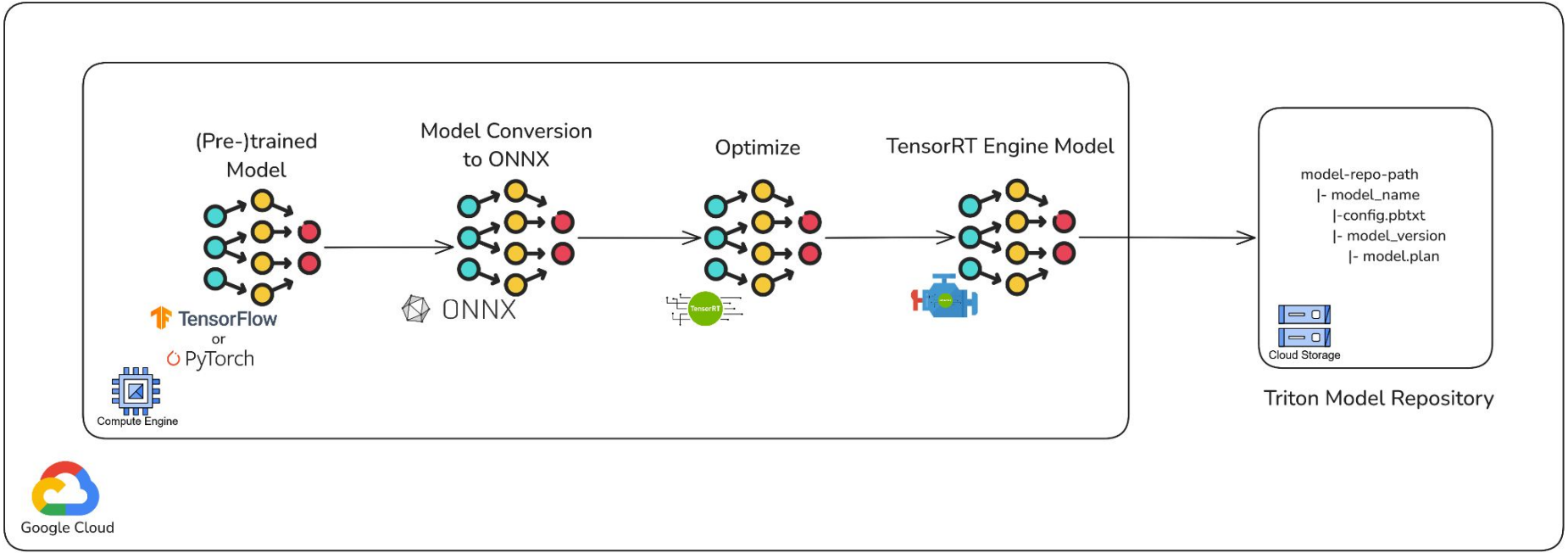
Web App

ML Platform

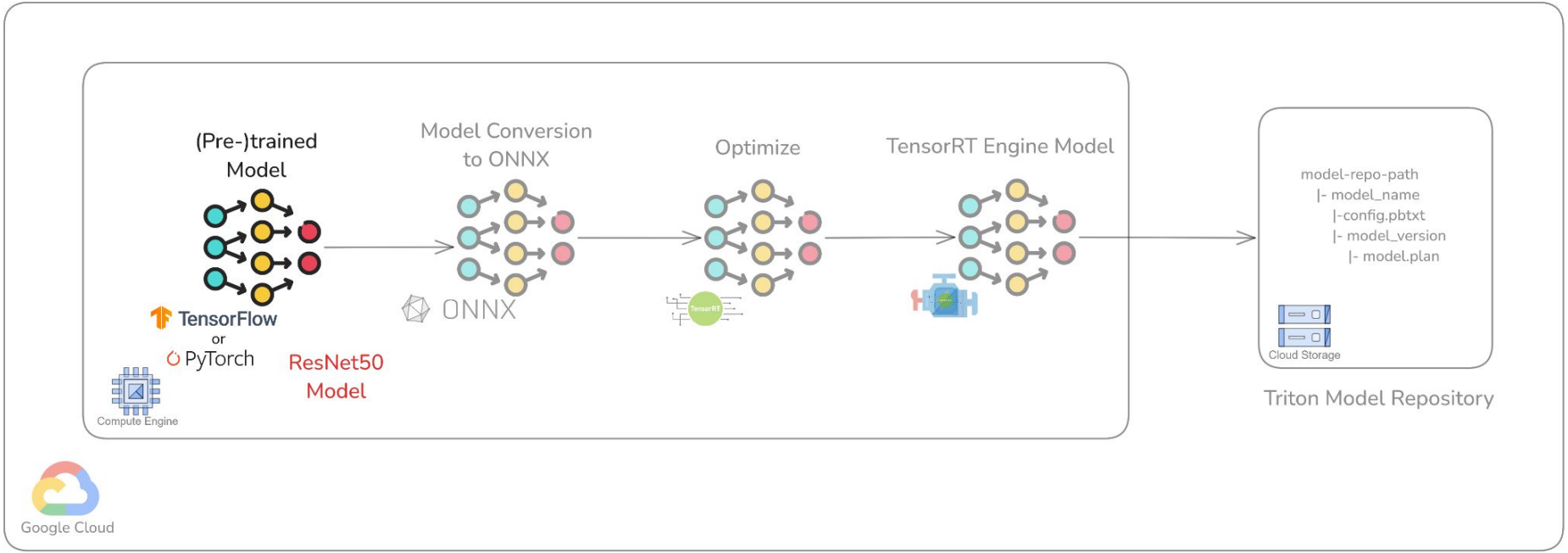
Design & Implementation



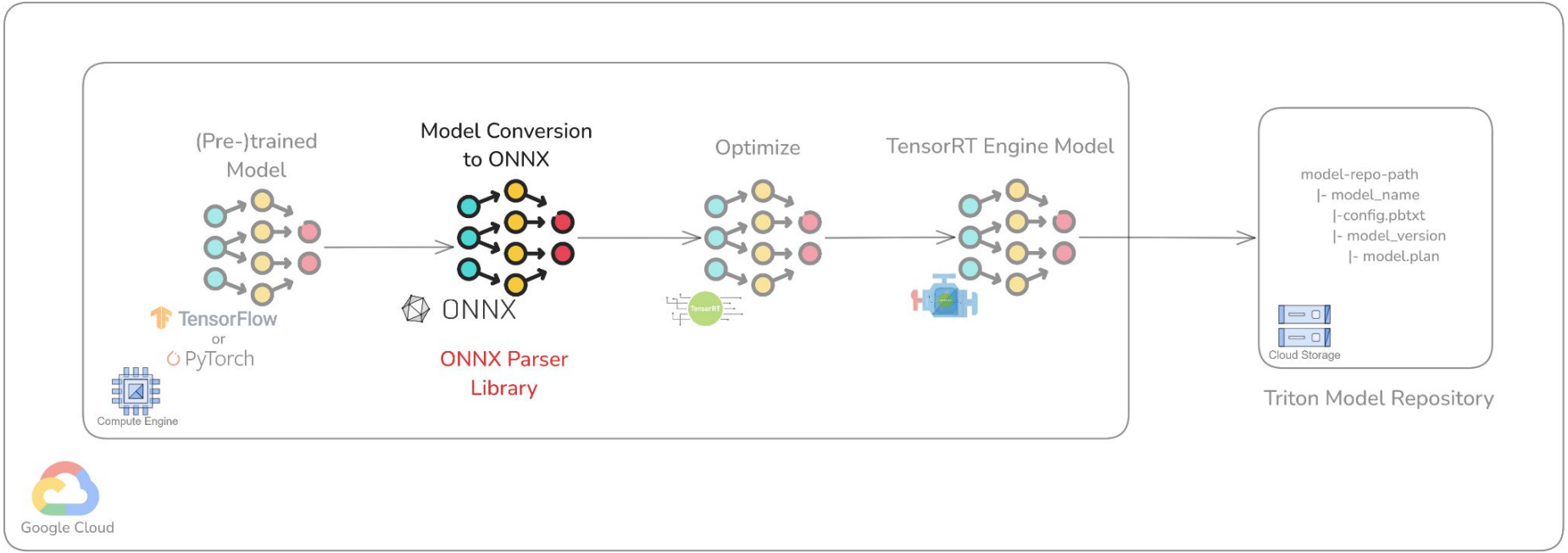
Design & Implementation



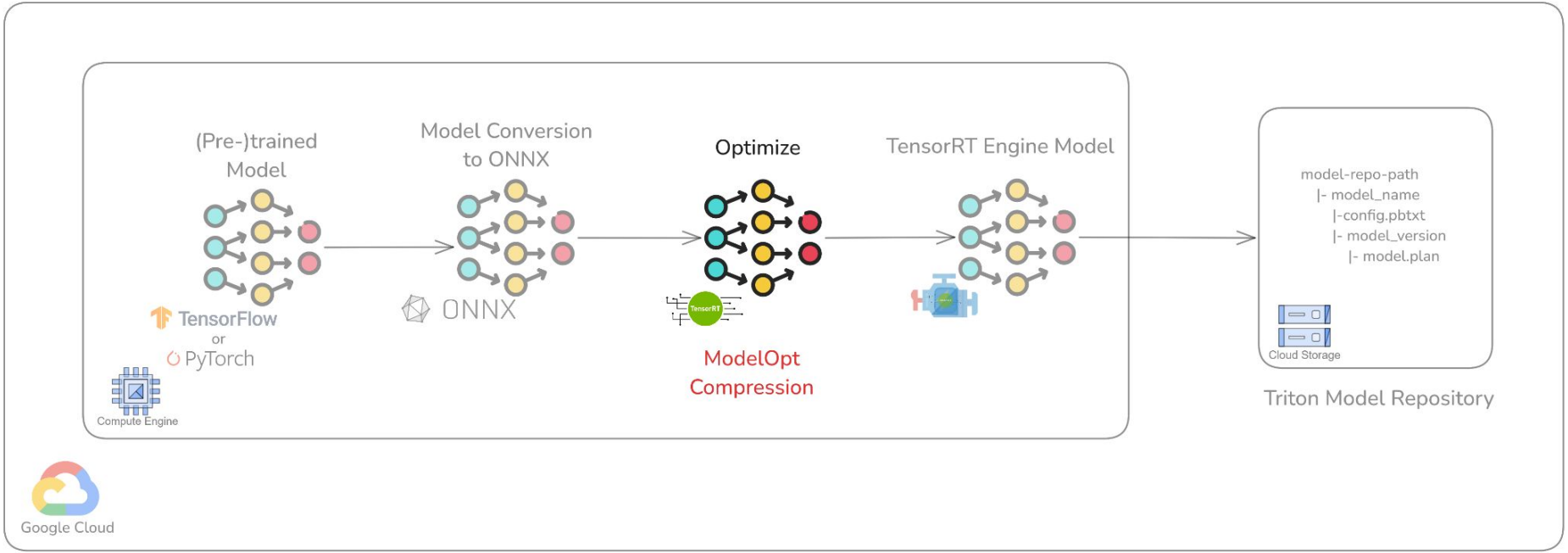
Design & Implementation



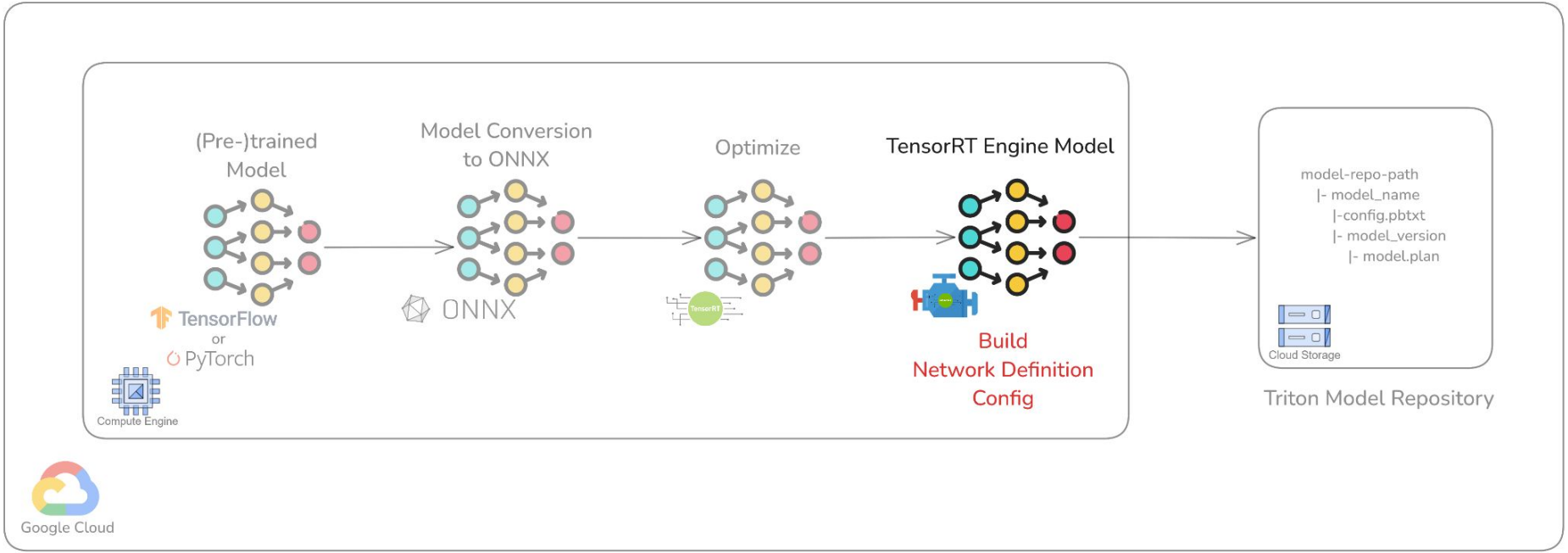
Design & Implementation



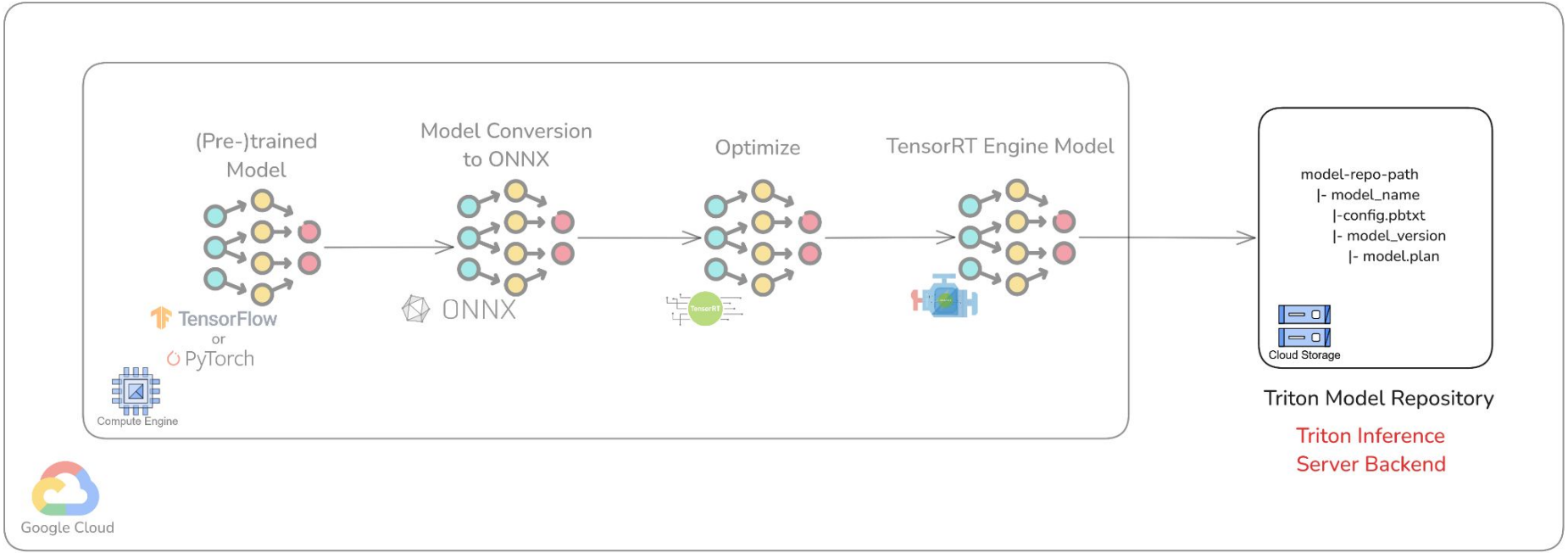
Design & Implementation



Design & Implementation

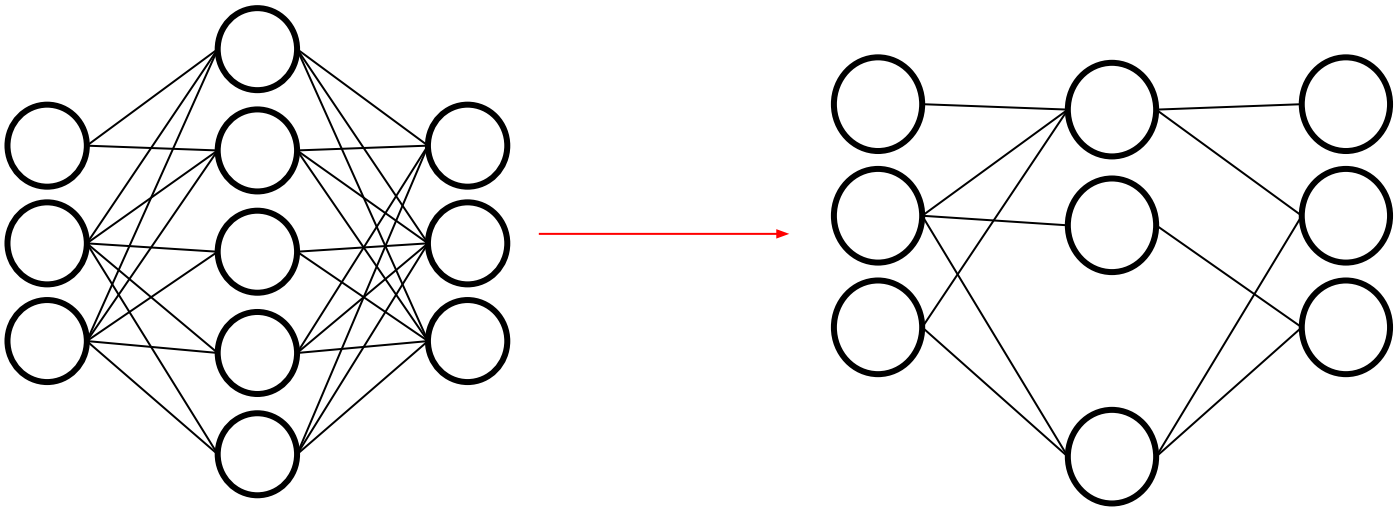


Design & Implementation



Design & Implementation

Pruning in TensorRT: Finding the optimal sub-network



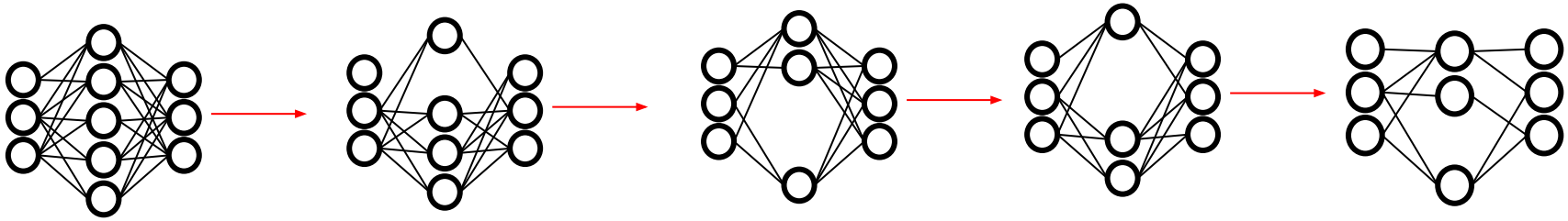
Design & Implementation

Set constraints: Give me a subnet with 70% of the FLOPs.

Sensitivity analysis: Understand how pruning different elements of the model affects performance

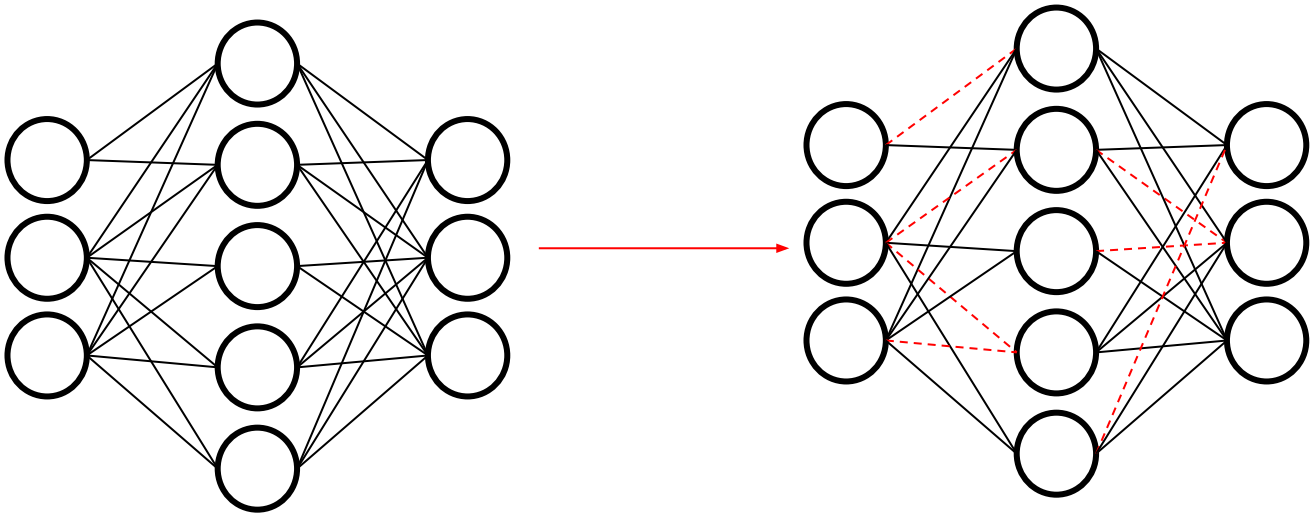
Search: Binary search over the search space of architectural hyperparameters (e.g., number of channels, layers, etc.)

Return the highest performing (best scoring) subnet that meets the constraints

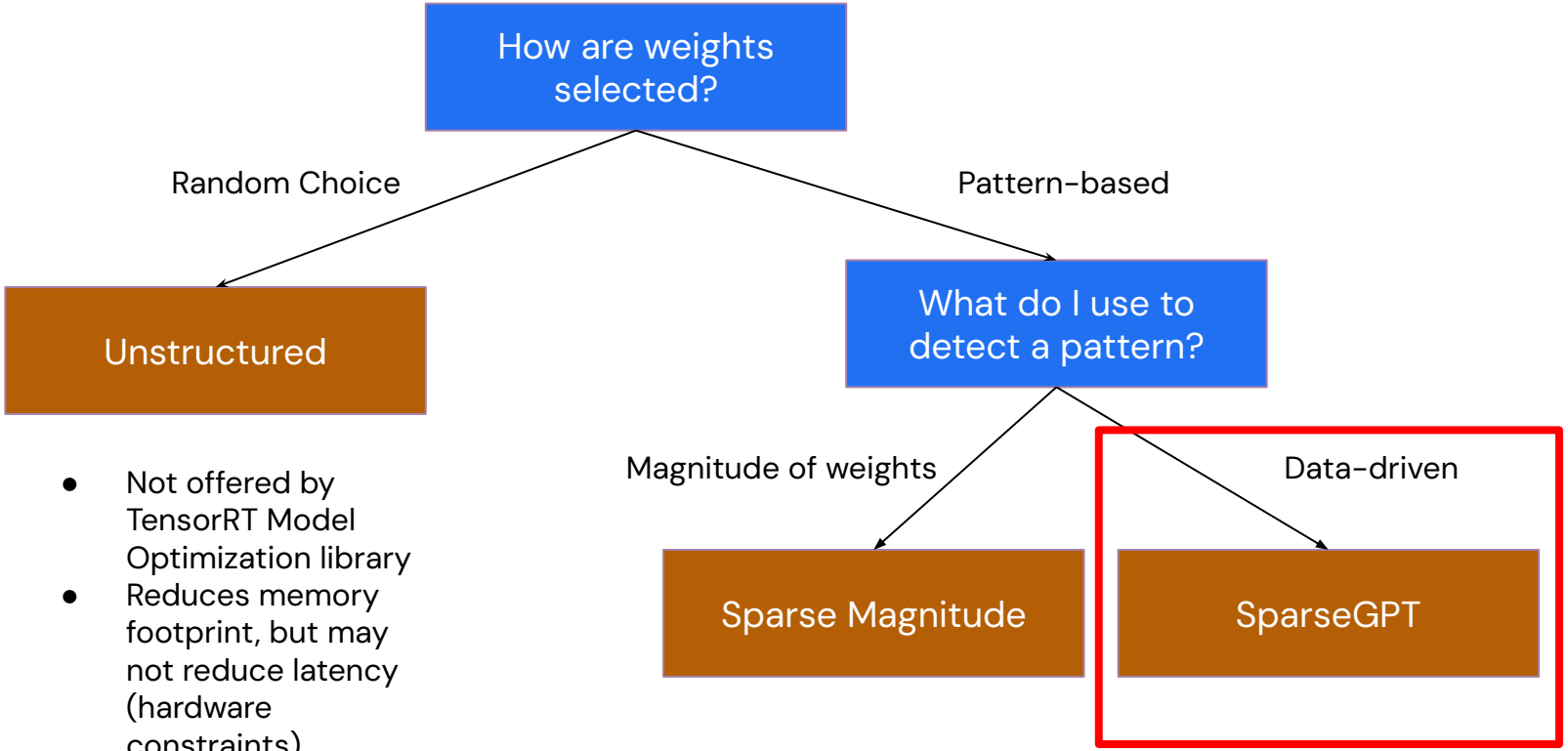


Design & Implementation

Sparsification in TensorRT: Zeroing individual weights



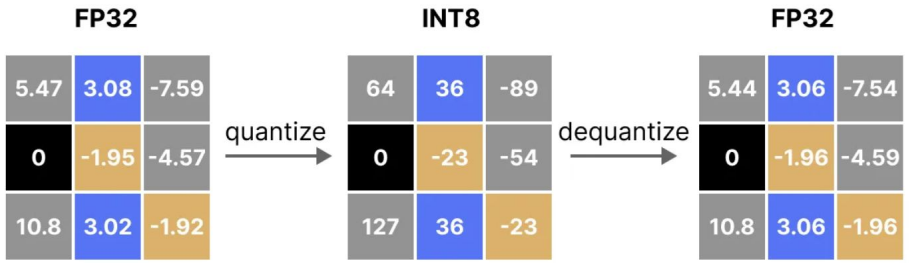
Design & Implementation



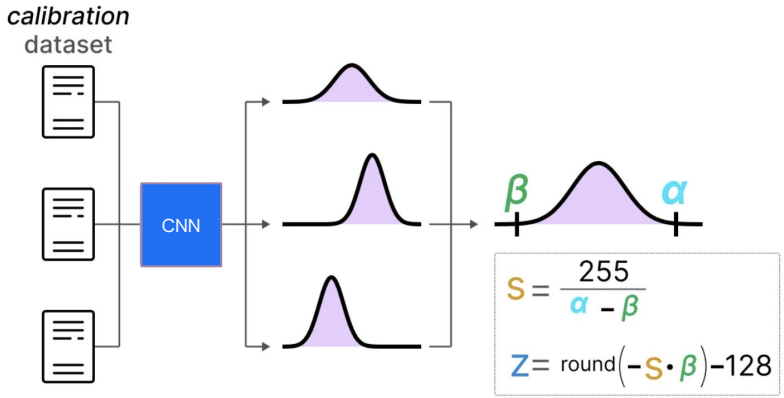
- Not offered by TensorRT Model Optimization library
- Reduces memory footprint, but may not reduce latency (hardware constraints)

Design & Implementation

Post-training Quantization:
Quantized to INT8

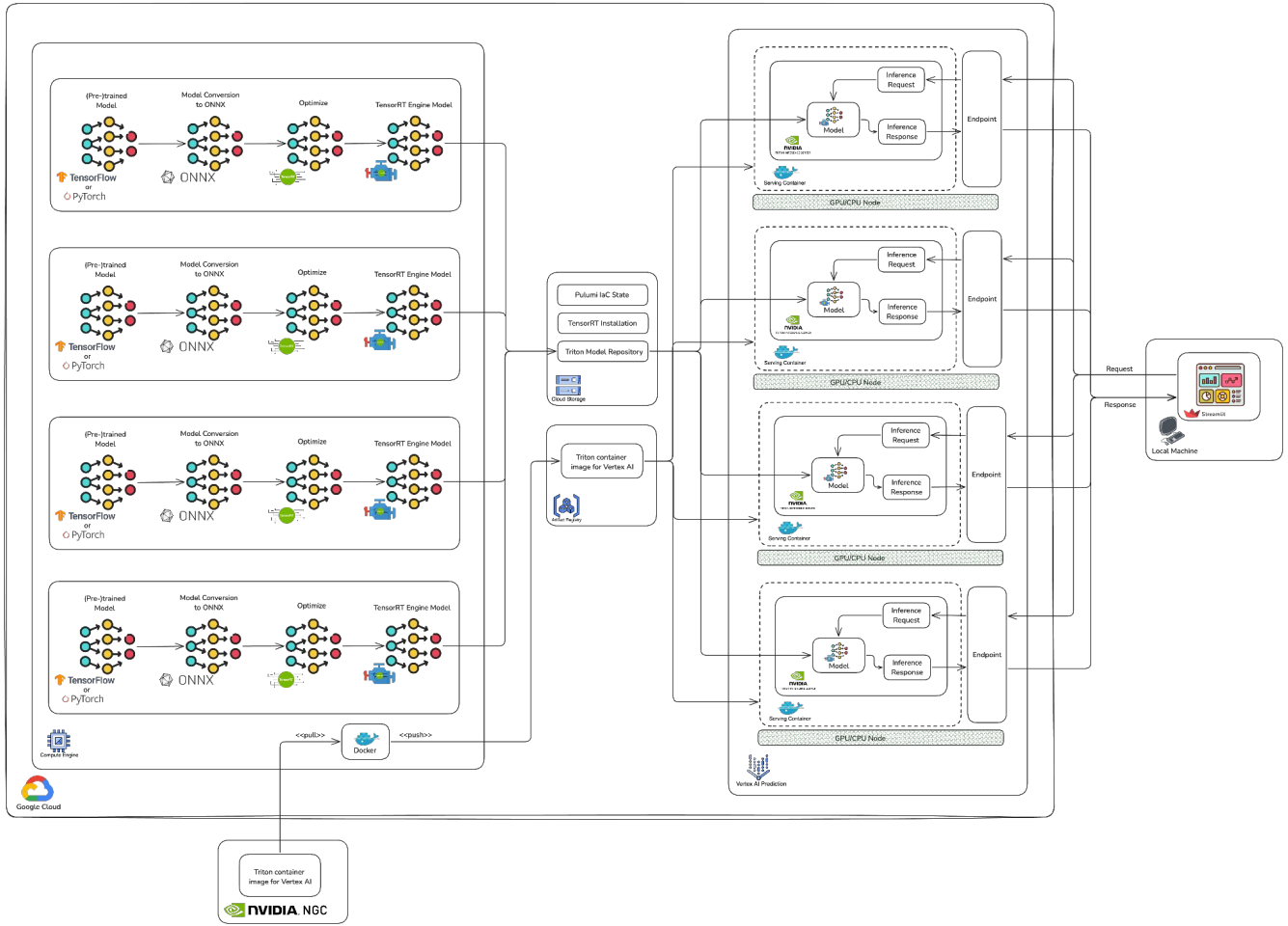


Static Quantization:
Calibration with a subset of ImageNet (1000 images)



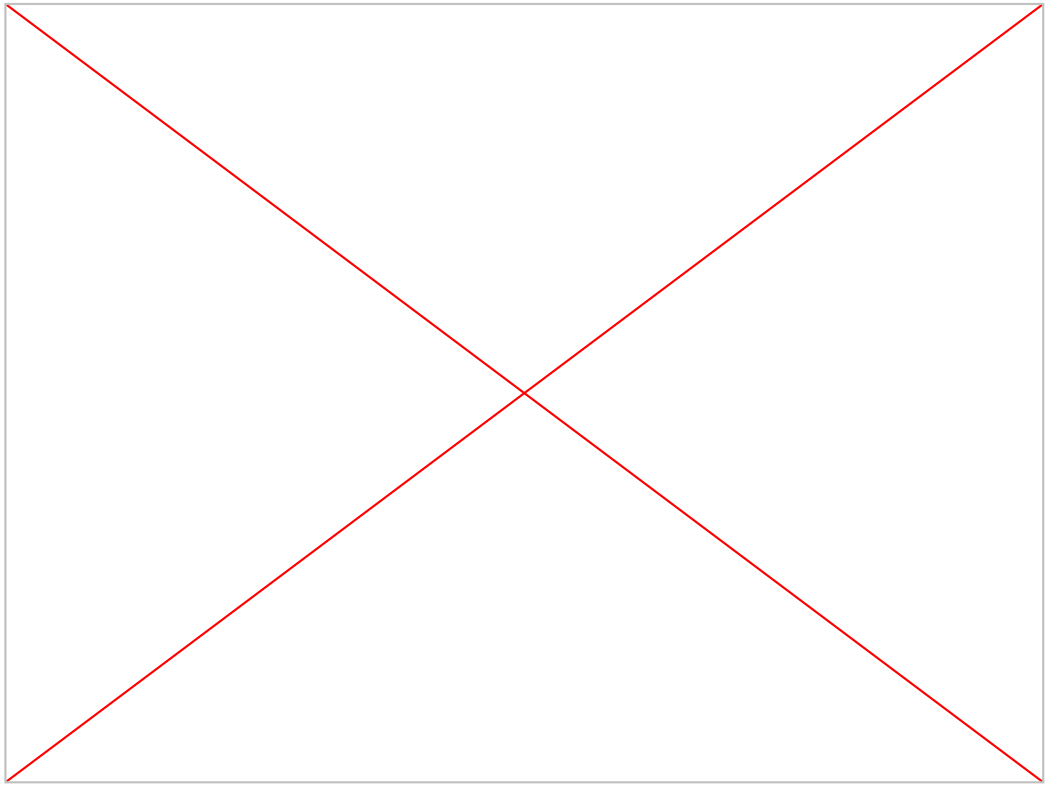
Design & Implementation

Whole Picture



Demo

Demo recording in case the demo gods are not on our side

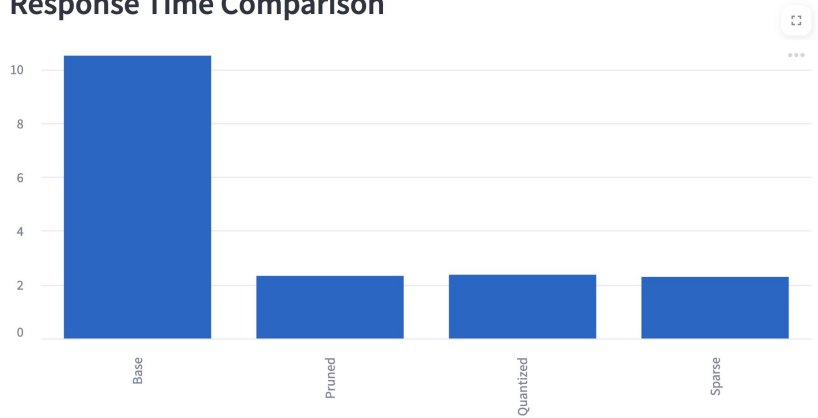


Evaluation

Model Comparison Summary

	Model	Top Prediction	Confidence	Response Time (s)	Location	Batch Size
0	Pruned	jackfruit, jak, jack	3.1431	2.326	us-central1	1
1	Sparse	sulphur-crested cockatoo, Kakatoe galerita, Cacatua galerita	6.3566	2.288	us-east4	1
2	Quantized	tailed frog, bell toad, ribbed toad, tailed toad, Ascaphus trui	6.1126	2.368	us-west1	1
3	Base	tree frog, tree-frog	15.6493	10.503	us-east1	32

Response Time Comparison



Discussion

Challenges

- TensorRT (installation, working w/ diff versions)
- Vertex AI deployment time (~30 min - 40 min)
- Manual deployments

Unexpected outcomes

- Frogs look like jackfruit… (if you squint enough 😑)

Conclusion & Future Work

Key Takeaways

- Managed ML platforms i.e. Vertex AI, Sagemaker, Azure ML make somethings easy, but come with their own complications and challenges.

Future Work

Model Development

- Optimization with distillation
- Comparing static and dynamic quantization latency
- Comparing magnitude and data-driven sparsity

Conclusion & Future Work

Future Work

ML Platform

- Deploy models to the same endpoint. Configure Triton Server to handle multiple models.
- Create test environment. Automate deployment with deployment pipeline.
- Reduce permission scope & use service account credentials.

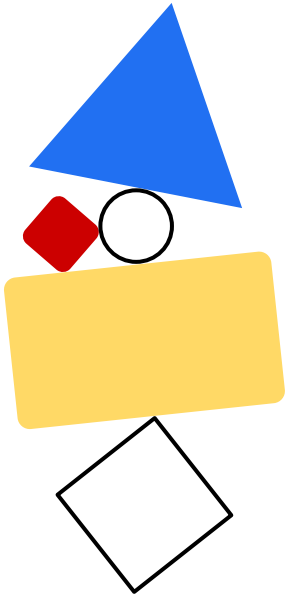
Benchmarking

- Collect more results from testing with more images.
- Load test the endpoints.

References

- Homework 3 - Simple DL Workflow in Kubeflow
- Cole Bailey: Cooking up a ML Platform - Growing pains and lessons learned | PyData YouTube
- NVIDIA Triton Inference Server and its use in Netflix's Model Scoring Service | Outerbounds YouTube
- Class Presentation - NN Optimization Frameworks: Torch2Compile, TensorRT, Faster Transformer
- How to Put AI Models into Production: A Guide to Accelerated Inference | NVIDIA Book
- 2024 State of AI Inference Infrastructure Survey Report by BentoML
- Chapter 5. Building Infrastructure Stacks as Code | O'Reilly Book by Kief Morris
- How Pulumi Works | Docs Pulumi IaC Concepts
- Visual Guide to Quantization

Questions? Comments?



Thank you for your time and attention.

Timothy Chang
Tyler Chang
Athitheya Gobinathan
Pablo Ordorica-Wiener