



MID-PROJECT REPORT CSE 239 PROJECT: PERSONAL GPT

Mathis AUBERT

PhD Student in Computer Science.

maubert@ucsc.edu

Table of Contents

<i>Project title: Personal GPT</i>	3
Project statement.....	3
<i>Project Overview</i>	4
Functional Implementation.....	4
Tools & Frameworks.....	4
System Architecture.....	5
Performance	6
Metrics.....	6
Scalability.....	6
Comparison between Google Cloud Platform and Nautilus.....	7
<i>Initial GANTT</i>	8
<i>Risk analysis</i>	9
<i>Source code availability</i>	9
<i>Reference</i>	10

Project title: Personal GPT

Project statement

Goal: Train and deploy a machine learning model

Description: Deploy a Large Language Models (LLMs) leveraging cloud-managed services. In this project, you will be containerizing and deploying llama.cpp. The main goal of llama.cpp is to enable LLM inference with minimal setup and state-of-the-art performance on a wide variety of hardware - locally and in the cloud. Students will develop and deploy llama.cpp using GCP and Kubernetes for auto-scaling. The focus is on performance tuning and resource optimization during model inference. Students will need to test various models under different configurations (e.g., quantizations) that may result in different performance-cost tradeoffs.

Requirements:

- Containerize and build a llama.cpp (<https://github.com/ggml-org/llama.cpp>) pipeline with feature engineering and model deployment.
- Demonstrate how different models compare for various input queries.
- Profile training time, inference latency, and resource consumption of the models under various parameters.
- Use auto-scaling and GPU acceleration to achieve minimal average latency for user queries.
- Compare cost-performance for different instance types on GPC and on Nautilus.

Deliverables:

- Source code and model deployment pipeline.
- Performance profiling results.
- Visualization of cost-performance trade-offs and system behavior.

Project duration: November 2, 2025 - December 4, 2025

People: Mathis AUBERT

Project Overview

The project goal is to deploy and optimize a containerized llama.cpp pipeline on Google Cloud Platform (GCP) and Kubernetes, focusing on auto-scaling, GPU acceleration, and cost-performance trade-offs for real-time LLM inference.

The key objectives are:

- Containerize and deploy llama.cpp with feature engineering.
- Profile and compare models under different quantizations and hardware configurations.
- Implement auto-scaling and load balancing for minimal latency.
- Benchmark system stability, scalability, and cost-efficiency.
- Visualize performance metrics and system behavior under load.

Functional Implementation

Containerization: Dockerize llama.cpp with optimized build flags for GPU/CPU inference.

Deployment Pipeline: Helm charts for Kubernetes deployment on GCP (GKE) and Nautilus.

Model Selection: Test 2/3 models (e.g., Llama-2-7B, Mistral-7B) with varying quantizations (Q4, Q5, Q8).

Auto-scaling: Horizontal Pod Autoscaler and Cluster Autoscaler for dynamic workloads.

Load Balancing: Nginx Ingress Controller for traffic distribution.

Tools & Frameworks

Table 1: Tools and Frameworks used for this project

Category	Tools
Containerization	Docker [1]
Orchestration	Kubernetes [2]
Load balancer	Nginx [3]
Monitoring	Grafana [4], GCP Cloud Monitoring [5], Prometheus [9]
Benchmarking	K6 [6]
Visualization	Grafana Dashboards [4]
CI/ CD	GitHub Actions [7]
Version control	GitHub [8]

System Architecture

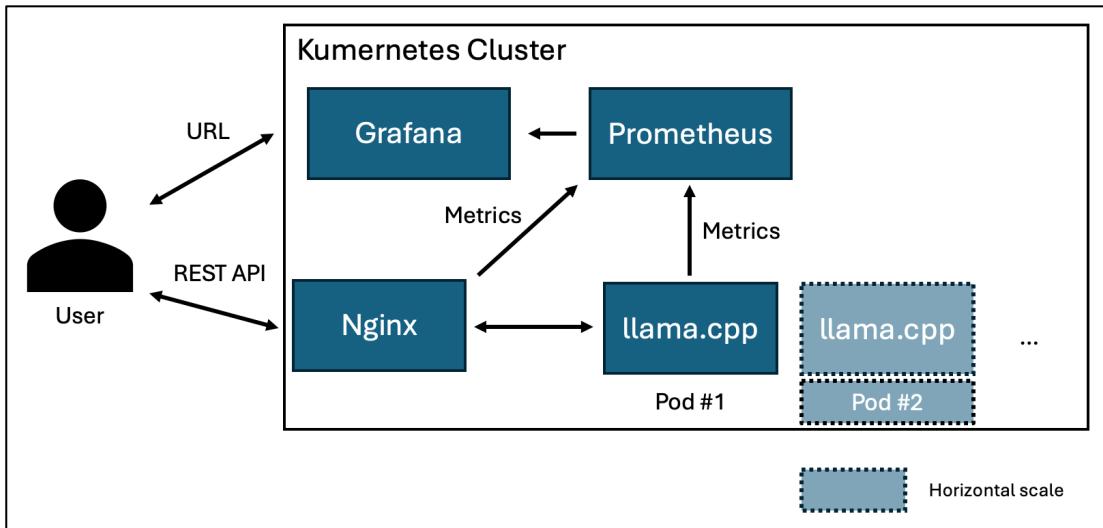


Figure 1: Personal GPT - System Architecture

Key Components:

- **Nginx Controller**: Routes user requests to pods.
- **Kubernetes Pods**: Run containerized `llama.cpp` with different quantizations.
- **Monitoring**: Prometheus collects metrics; Grafana visualizes.
- **Auto-scaler**: HPA adjusts pods based on CPU/GPU load.
- **Infrastructure**: Deployed on both GCP (GKE) and Nautilus for comparison.

Performance

Metrics

Metric	Definition	Tools	Target
Latency	P95/P99/Average response time per query	K6, Prometheus	<500ms avg, <1s P99
Accuracy	% of correct answers (MT-Bench)	MT-Bench scripts [10]	>85% for Q8, >80% for Q4
Cost	Total \$ spent per 1K queries (GCP vs. Nautilus)	GCP Billing API, Nautilus logs	<\$0.10/1000 queries
Execution Time	Total time to process a dataset		
Scalability	% of requests meeting <1s response time under load	Grafana, HPA logs	90% at 50 users
Resource Usage	CPU/GPU/memory utilization	cAdvisor, NVIDIA Nsight	<70% avg under peak load

At the end of the project, a complete report will be done with evaluation of these metrics and the comparison between the implementation on Google Cloud Platform and Nautilus.

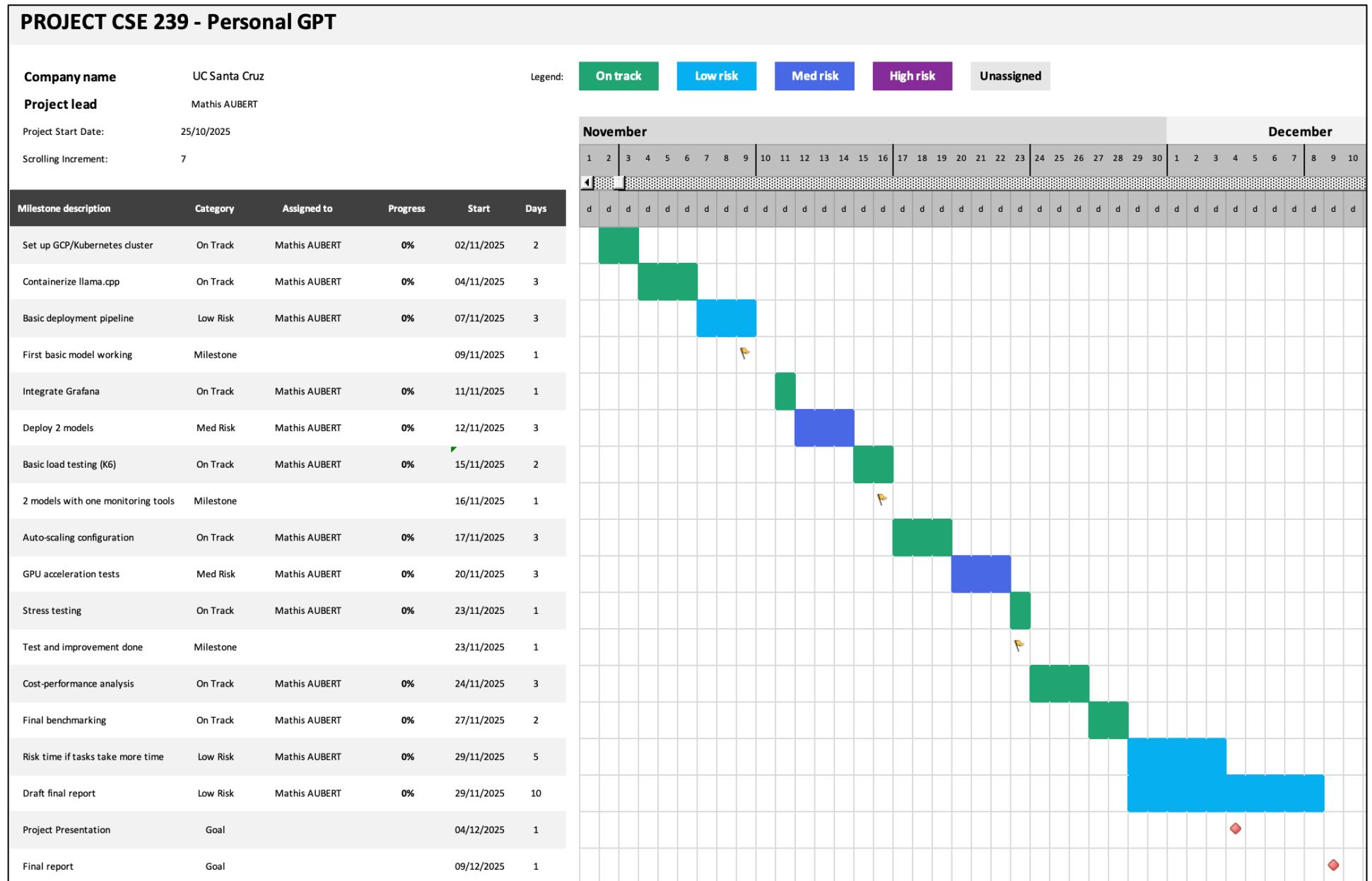
Scalability

Metric	Threshold	Source	Action Triggered
CPU Utilization	>70% avg over 1 minutes	Kubernetes Metrics Server	Scale up pods (add 1–2 replicas)
GPU Utilization	>60% avg over 1 minutes	NVIDIA DCGM + Prometheus	Scale up GPU-enabled pods
Memory Usage	>80% of pod memory limit	cAdvisor	Scale up or restart pod if OOM
Request Latency	P99 > 1s for 2 minutes	Prometheus (Nginx metrics)	Scale up pods or adjust load balancer weights
Requests per Second	>50 RPS	Prometheus (K6 metrics)	Scale up pods preemptively
Queue Length	>10 pending requests	Prometheus (Custom metric)	Scale up pods immediately

Comparison between Google Cloud Platform and Nautilus

Feature	Google Cloud Platform	Nautilus
Hardware	Different VMs choice, GPUs (if accessible)	Shared cluster, mixed GPU types
Cost model	Pay-as-you-go (50\$ limit for this project)	Free
Scaling Speed	Fast (seconds)	Slower (minutes)
Monitoring	GCP cloud monitoring + Prometheus	Prometheus
Storage	Persistent Disks (SSD)	Shared Network File System

Initial GANTT



Risk analysis

I'm worried about the potential cost of GPU constrainters (if possible, to acquire on GCP). So, I will work most of the time on my personal machine and I will try to find the best configuration before doing the final deployment on GCP.

Source code availability

For the entire lifetime of this project, my source code will be available on GitHub, via this link:

<https://github.com/Maths-A/personal-gpt>

The repository is private, don't hesitate to ask the access.

Reference

- [1] <https://www.docker.com>
- [2] <https://kubernetes.io/docs/home/>
- [3] <https://nginx.org/en/docs/>
- [4] <https://grafana.com/docs/grafana/latest/>
- [5] <https://cloud.google.com/docs/get-started/>
- [6] <https://grafana.com/docs/k6/latest/>
- [7] <https://github.com/features/actions>
- [8] <https://github.com>
- [9] <https://prometheus.io>
- [10] https://huggingface.co/datasets/lmsys/mt_bench_human_judgments