Hyperscalers Benchmarking Toolkit

# Project Overview

The Hyperscalers Benchmarking toolkit compares the Llama 3 offerings from Azure OpenAI, Google Vertex AI, and AWS Bedrock. It automates identical prompts, captures execution metrics, and aggregates the results so you can make data-informed platform decisions.

# Key Features

- Single command orchestrator `run\_all\_benchmarks.py` runs the provider benchmarks back-to-back and tracks total elapsed time.

- Provider scripts issue native API calls, capture latency and token usage, and log helpful DEBUG output for troubleshooting.

- Results flow into provider-specific CSV files as well as aggregated summary and transposed views for quick comparisons.

- Metrics cover response time, token counts, character and word counts, plus coarse cost estimates using per-1K token pricing.

# Repository Structure

- `run\_all\_benchmarks.py` orchestrates sequential runs and builds consolidated CSV summaries.

- `azure\_llama\_demo.py`, `gcp\_llama\_demo.py`, and `aws\_llama\_demo.py` handle provider-specific authentication, invoke Llama 3, and persist run metrics.

- CSV outputs include per-run detail plus an averages row so the orchestrator can extract comparable statistics.

# Setup Requirements

- Python 3.8 or newer with dependencies: boto3, google-genai, vertexai, azure-ai-inference, and tiktoken.

- Cloud credentials exported as environment variables for each provider. The README documents exact variables and regions.

- Access to the respective Llama 3 deployments in Azure, Google Cloud, and AWS Bedrock including invoke permissions.

# Typical Workflow

- Configure environment variables with provider credentials and model identifiers.

- Install Python dependencies in your virtual environment.

- Run `python run\_all\_benchmarks.py` to kick off all providers with the shared question.

- Inspect the provider CSV files plus `benchmark\_summary.csv` and `benchmark\_summary\_transposed.csv` for aggregate insights.

# Extending the Toolkit

- Update the shared question or wire in command-line arguments to evaluate additional scenarios.

- Add new providers or model variants by cloning an existing script and adjusting authentication and pricing constants.

- Enhance cost accuracy by refreshing the per-1K token pricing constants or ingesting billing exports.

# Operational Notes

- Each run records UTC timestamps, making it easy to correlate with provider logs or throttling events.

- Token counts fall back to a tiktoken estimate when providers omit usage data, ensuring statistics remain populated.

- DEBUG statements in the provider scripts surface raw responses to help diagnose schema or permission issues.

# Next Steps

- Review the README for the latest setup guidance and adjust prompt length, temperature, or repetition count to fit your benchmarking goals.

- Consider layering visualizations or statistical tests on top of the CSV outputs to highlight significant differences between providers.