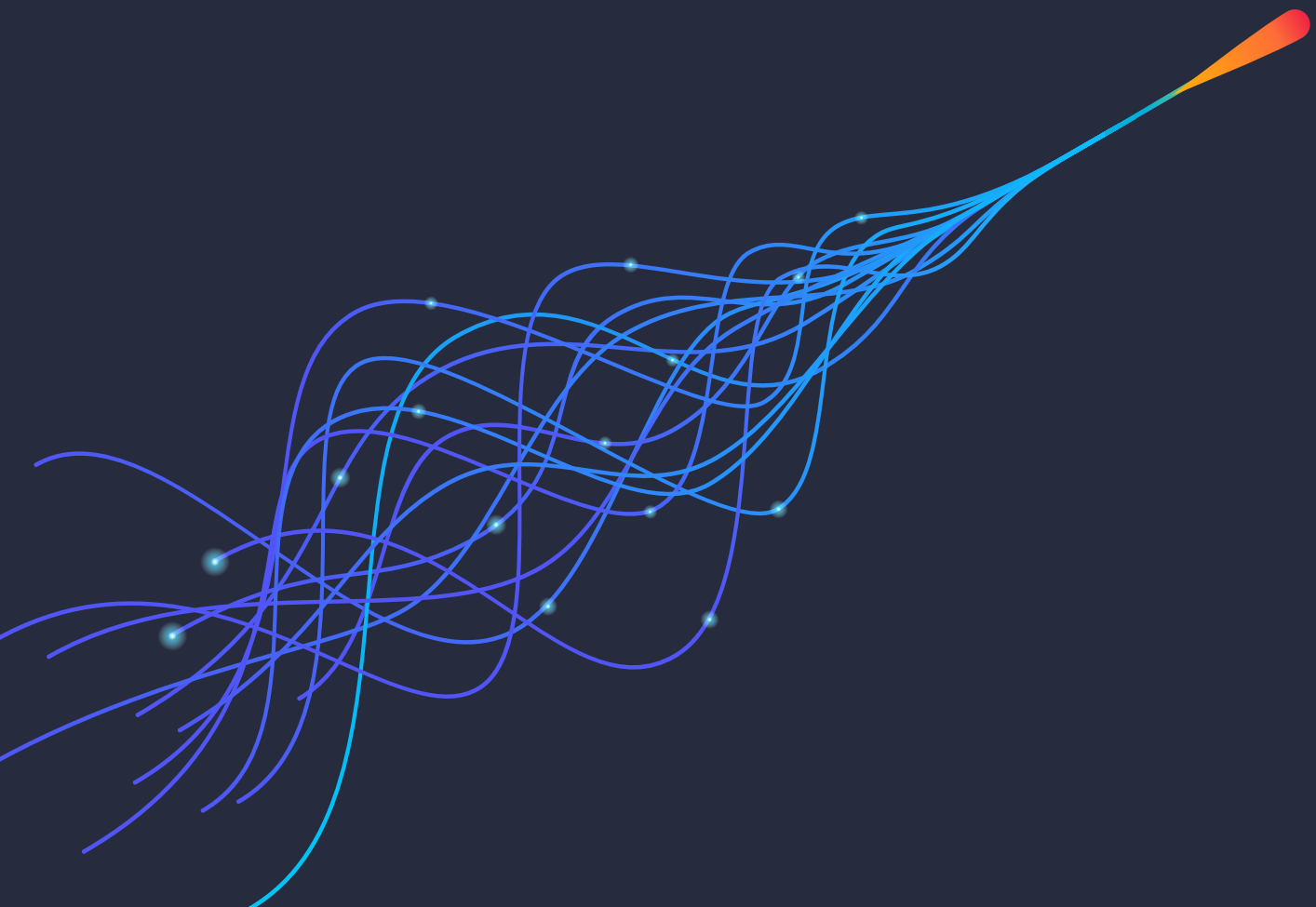
comet

Vertex



Introduction

Vertex is Google's end to end Machine Learning platform for training and deploying models.

Comet is an excellent complement to Vertex, enhancing the developer experience by allowing users to easily track experiments, collaborate with team members, and visualize results in an intuitive and easy-to-understand way while using the frameworks and tools that they are most comfortable with. Additionally, the platform provides a wide range of customization options, including the ability to create custom visualizations and dashboards, so that users can tailor their experience to meet their specific needs.

By using Comet, users can streamline their workflows while benefiting from Vertex's powerful infrastructure orchestration and model deployment capabilities.

How does Comet work with Vertex?

Comet's SDK is integrated with Vertex AI Pipelines allowing you to easily log, visualize, and analyze your pipelines.

Vertex AI Pipelines helps you to automate, monitor, and govern your ML systems by orchestrating your ML workflow in a serverless manner. Comet automatically logs your pipeline metadata and artifacts with minimal changes to your existing pipeline code.

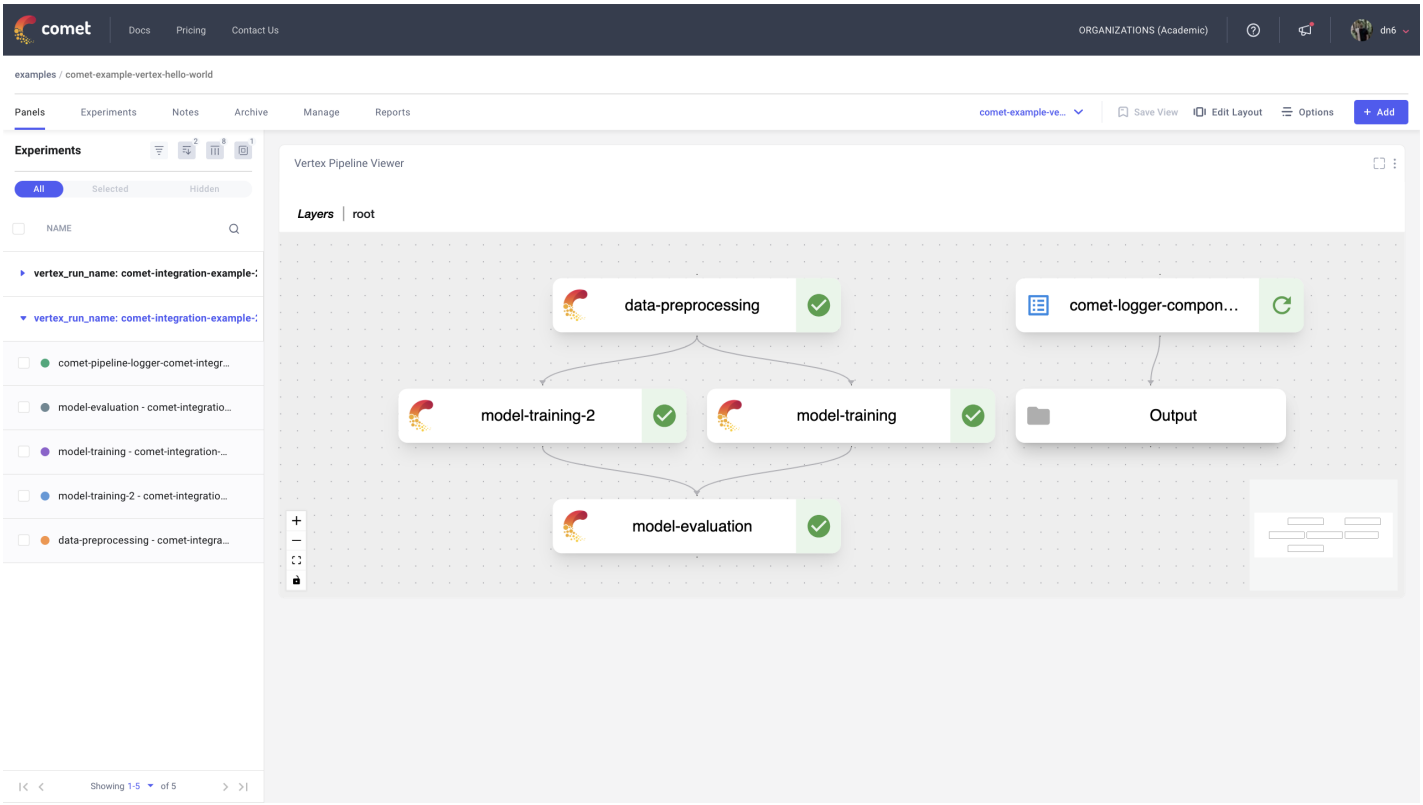
Let's take a look at a simple example of how to get started with Comet and Vertex

```
import comet_ml.integration.vertex
import kfp.dsl as dsl

@dsl.pipeline(name="ML training pipeline")
def ml_training_pipeline():
    # Add the Comet logger component
    comet_ml.integration.vertex.comet_logger_component(
        api_key="<>", workspace="<>", project_name="<>"
    )
```

You can inject your credentials into the Comet Logger component via environment variables in your Vertex run. The Comet Logger creates an Experiment object when the Vertex pipeline is compiled into JSON.

Through the Experiment object, the Vertex pipeline code, metrics, hyperparameters, are logged automatically to Comet. Additionally the Vertex Run Name, Run ID, Task Type, Pipeline Type, and Pipeline State are saved to Comet. Once the data is logged to Comet, you can explore the pipeline using Comet's Custom Panel for Vertex AI.



The individual tasks in the pipeline are logged as separate experiments in Comet, and can be grouped based on the pipeline metadata such as Run Name or Run ID.

Visit our examples repository for a full example of using Comet with Vertex.

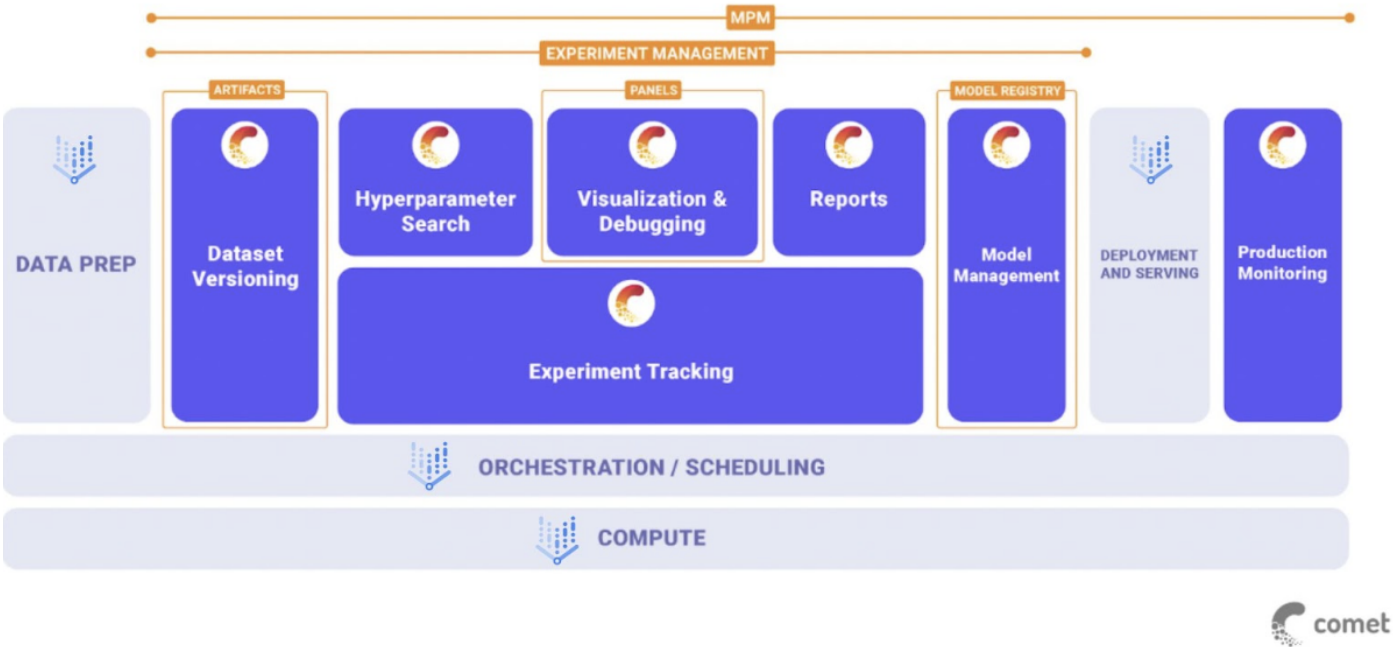
Why use Comet with Vertex?

Simplicity

Comet is a comprehensive and user-friendly machine learning platform that emphasizes a lightweight, API-focused approach. Its features are accessible through a single SDK, which is available in multiple programming languages, as well as through a REST API.

Comet serves as a knowledge base for ML projects, offering practitioners a single tool to log, visualize, and share any type of data related to their experiments. This enables them to spend less time piecing together disparate technologies and infrastructure for tracking ML runs, and more time developing models.

In contrast, bundled platforms are typically composed of different services from the platform provider. For example, running a Vertex Pipeline requires setting up Google Cloud Storage buckets, knowledge of the Kubeflow Pipelines SDK, and the Vertex SDK. Users tend to be forced into using multiple services to accomplish the singular task of training a model.

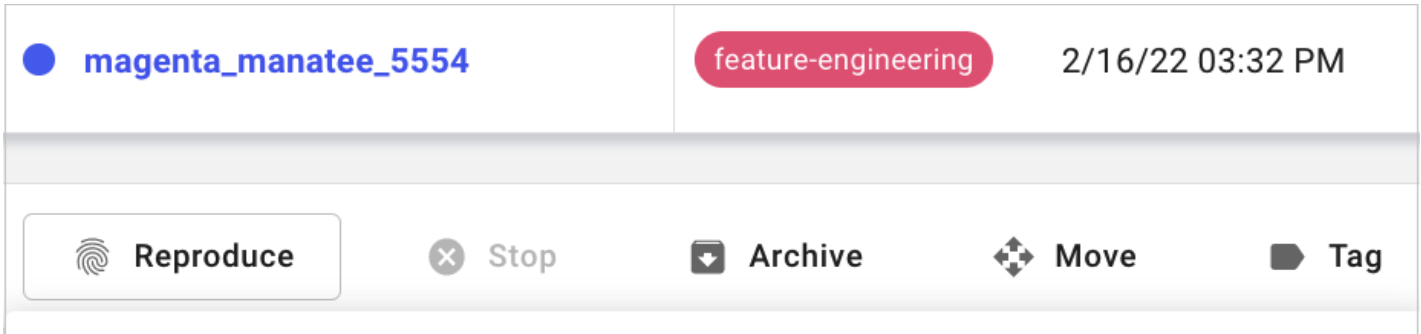


Reproducibility

Machine learning (ML) systems consist of many components, including source code, datasets, software dependencies, and hyperparameters. Any changes made to these components can affect the model's performance, so they need to be rigorously tracked.

However, Vertex does not support logging ancillary details such as software dependencies, operating system information, or git information. As a result, it becomes much more difficult to reproduce a specific training run to verify results. In domains with stringent compliance requirements, such as financial services and healthcare, full reproducibility is of utmost importance.

Comet is tightly integrated with git and automatically logs system details. This allows users to gain reproducibility out of the box with a single click



Environment Information

172.31.27.35 ip-172-31-27-35 Dhruv Nair

Git Information

commit 0a3c7a0e8d5568f53d81a305acef9068b093cb76

branch feature-engineering

patch View Download

Reproduce

```
$ git checkout 0a3c7a0e8d5568f53d81a305acef9068b093cb76
$ curl https://www.comet.com/api/rest/v1/git/get-patch?
  experimentKey=507c3b5ad42b462096ead8637b07a177 -
  H"Authorization: <Your Api Key>" > patch.zip
$ unzip patch.zip
$ git apply git_diff.patch
```

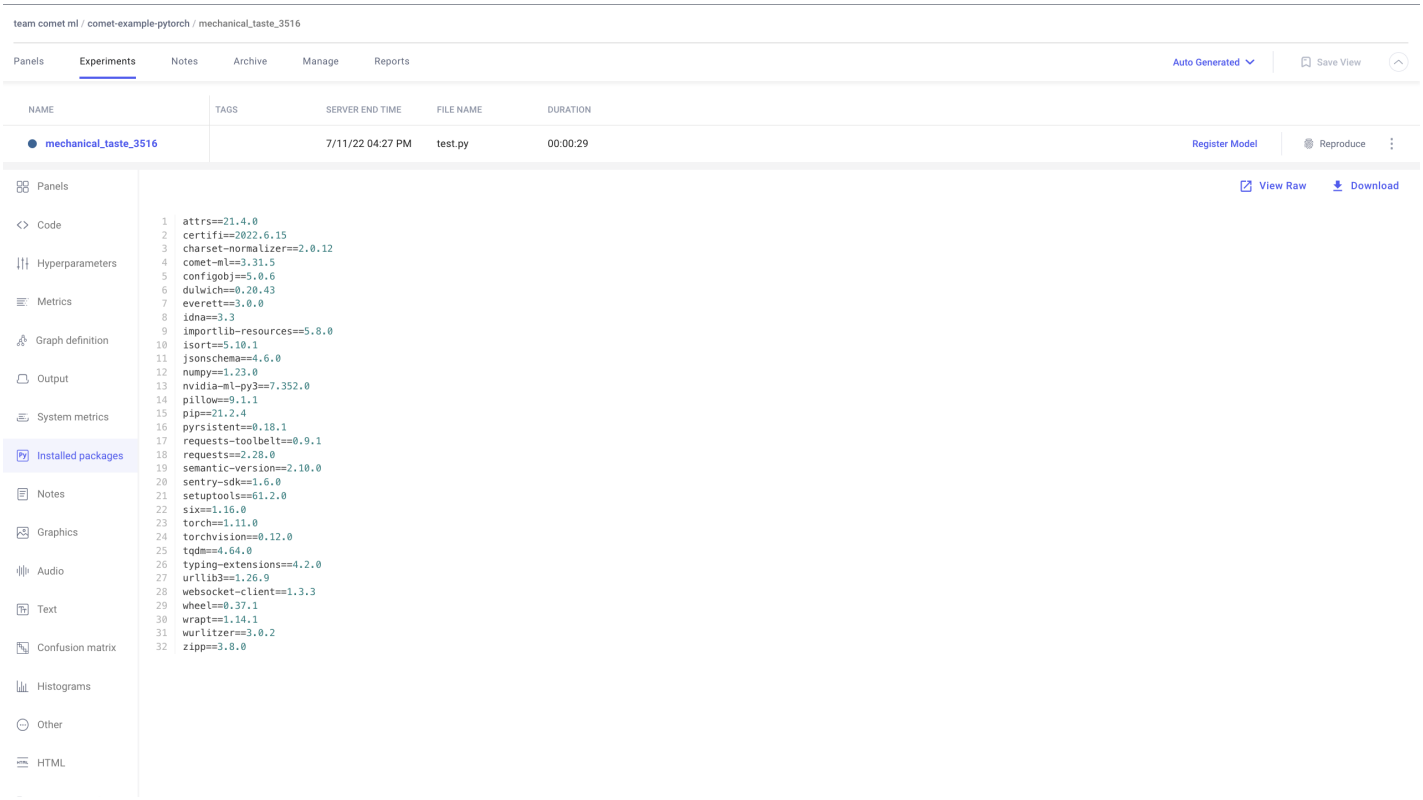
Run

```
$ /home/ubuntu/anaconda3/envs/hn-predictor/bin/python
  /home/ubuntu/dhruv/hn-
  predictor/hn_predictor/src/data/features.py --
  input_artifact_name hn-dataset --output_artifact_name hn-
  features --target_name score --pipeline_model_name
  distilbert-base-uncased --pipeline_batch_size 8 --
```

```
sampling_fraction 0.001 --message Extract timestamp features

and text features --random_state 42 --text_feature_name title
```

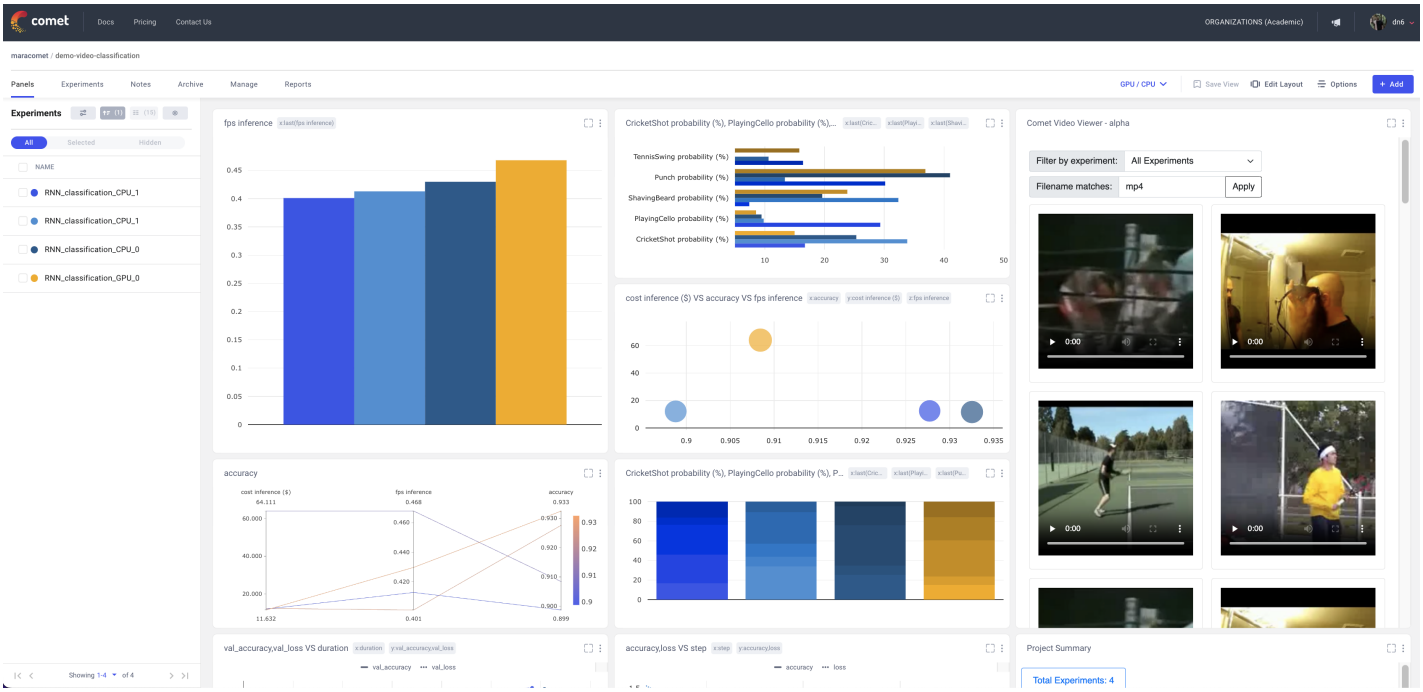
Comet also logs system information such as Operating System information and installed packages. Allowing you to see if changes in upstream dependencies are causing issues with your models.



Customizability

Machine Learning workflows vary dramatically between organizations and even between teams within the same organization. Comet is designed to be highly adaptable to the needs of individual practitioners, allowing them to customize their analysis in any way they see fit.

Comet's platform provides a wide range of customization options, including the ability to create custom visualizations and dashboards, so that users can tailor their experience to meet their specific needs. You can visualize your experiment data using either Comet's built-in panels or by building your own visualization tools with Comet Custom Panels.



Additionally, Comet supports saving and reusing visualization views in your projects, which allow you to create custom dashboards for individual stakeholders.

While Vertex does support logging experiment metadata, the platform's visualization capabilities are fragmented and can lead to a lot of friction when trying to gain a unified view of your experiment data. For instance, while Vertex does log metadata such as metrics and parameters, these are visualized as scalars in a separate part of the product, while time series and image data are visualized using Tensorboard, which does not support saving views or customizing the dashboard. This can make sharing insights between team members a more tedious process.

Avoid Vendor Lock In

When committing to an end-to-end platform, it often means that your workflow is closely tied to the product offerings of the platform. This can limit your flexibility in terms of the technologies you can use to develop your model. Do you want to run your training with a different GPU provider or migrate away from using Kubernetes to your own infrastructure? These tasks would not be possible when using a bundled platform, and would require a costly migration to another provider.

However, Comet stands out in that it is infrastructure agnostic. This means that code which utilizes Comet can run in any cloud provider's environment, as well as on-premises. This flexibility enables users to customize their infrastructure according to their specific needs, without worrying about vendor lock-in.

Conclusion

Vertex and Comet offer a complementary sets of tools for building a powerful machine learning platform. Vertex excels in resource management, computing, and deployment, while Comet leads in experiment management, artifact management, and production monitoring. Comet also provides a flexible user interface for analysis and reporting. By using both Comet and Vertex together, teams can access a comprehensive toolset that enables them to effectively and efficiently manage, monitor, and deploy their ML models.