MLOps IRL: Vertex AI and Kubeflow Pipelines

Code breakfast

Julian de Ruiter & Timo Uelen

DATA & AI CONSULTANCY AND TRAINING SERVICES

Driving Your Success With Data and AI

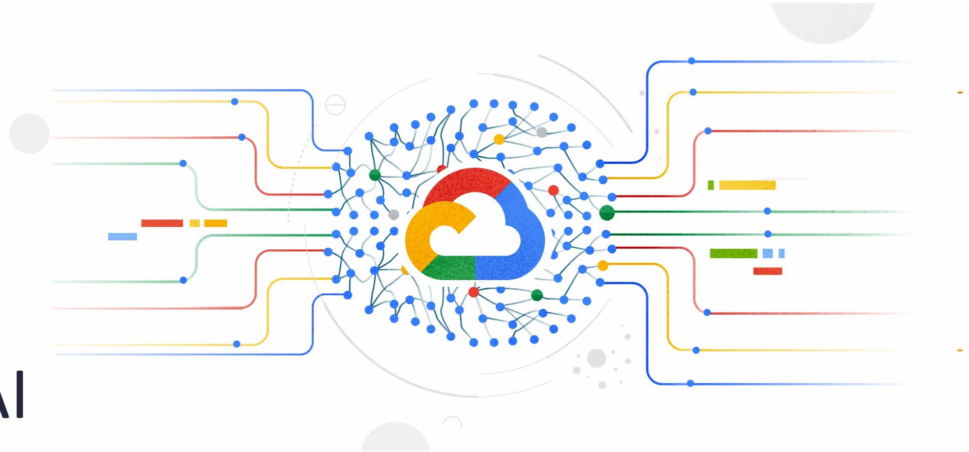
About us

- Julian de Ruiter
  - Background in computer and life sciences
  - Machine Learning Engineer at GoDataDriven since 2018
  - Co-author of Manning's 'Data Pipelines with Apache Airflow'
- Timo Uelen
  - Background in Information Sciences
  - Machine Learning Engineer at GoDataDriven since 2020

Julian

Timo

This morning



- 08:30 – 08:45 – Intro to MLOps + Vertex AI
- 08:45 – 09:00 – Getting started: Vertex Workbench
- 09:00 – 09:15 – Intro to Kubeflow pipelines
- 09:15 – 10:15 – Hackathon: building an ML pipeline
- 10:15 – 10:30 – Discussion & wrap-up

GO DATA DRIVEN

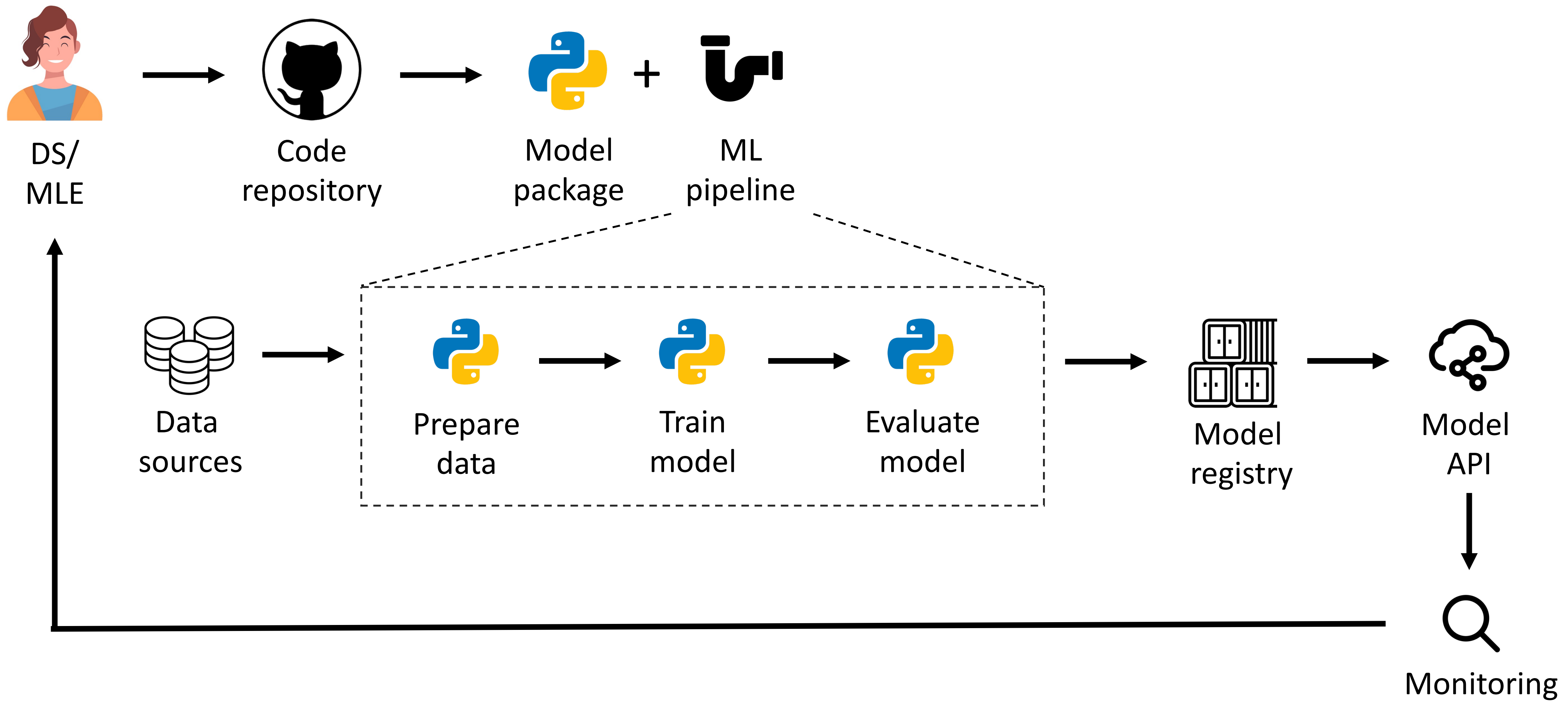
MLOps

What is MLOps?

What is MLOps?

- Practice that aims to apply lessons from DevOps to build reliable, reproducible and scalable machine learning solutions
- This includes
  - Continuous re-training with re-usable ML pipelines
  - Continuous delivery for new ML models
  - Short feedback loops (monitoring, user feedback)
  - End-to-end ownership of the ML product

What can this look like in practice?



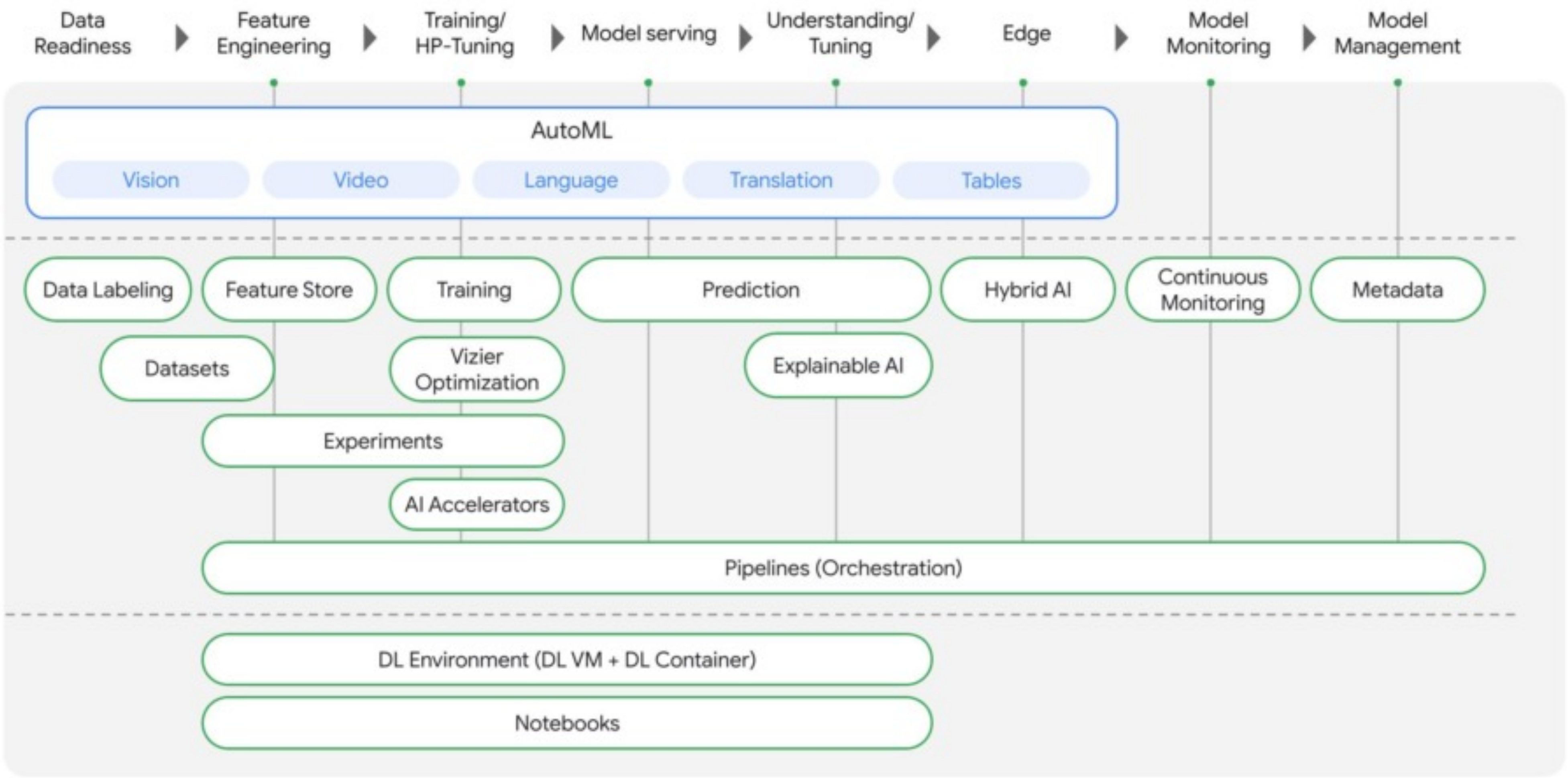
Vertex AI

What is Vertex AI?

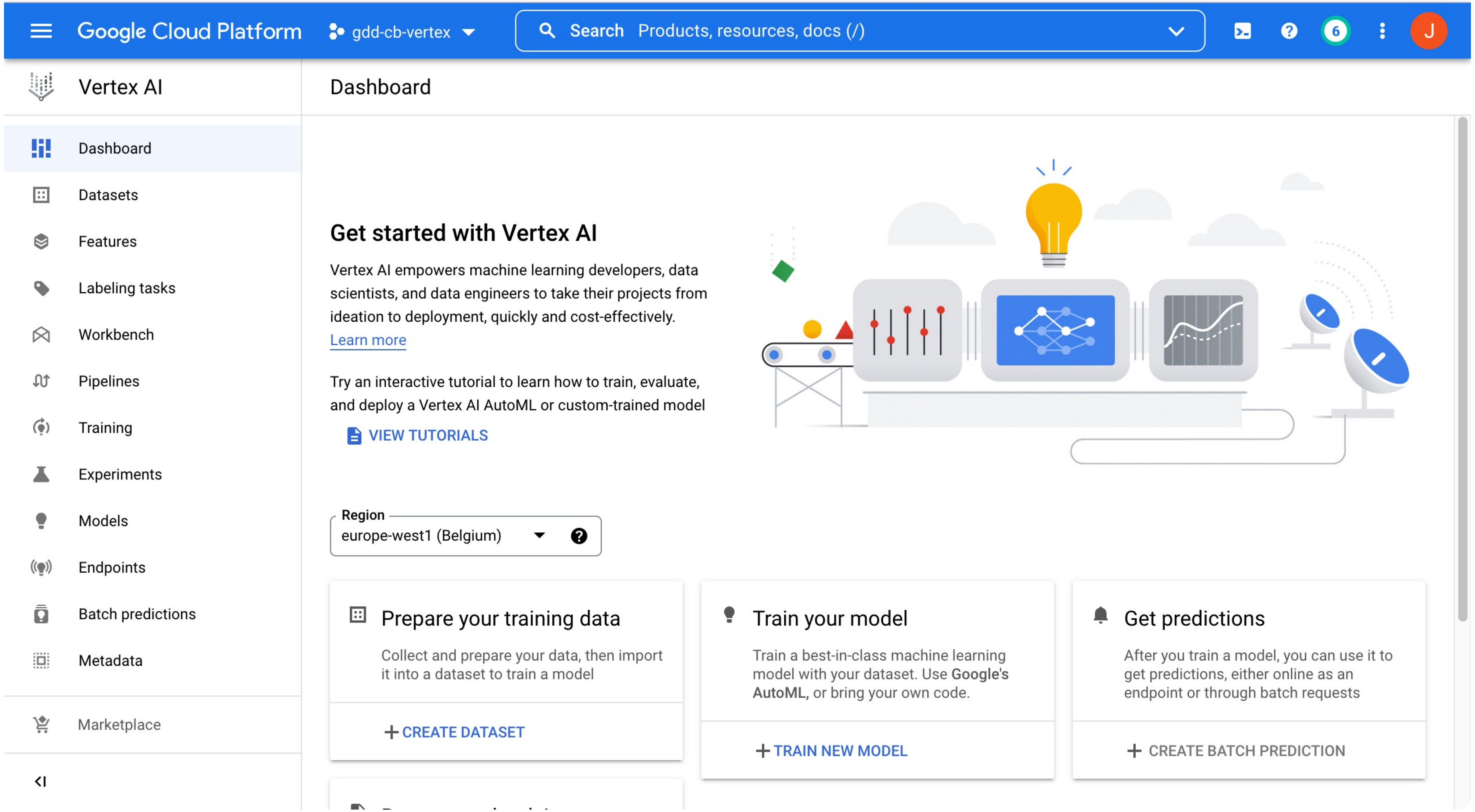
- Google's unified platform for ML (formerly *AI Platform*)

- Key features
  - Entire ML workflow (e.g. training, tracking and deploying models) in one unified UI
  - Integrates with open-source frameworks (e.g. Tensorflow) via custom containers
  - Easy integration with other GCP services such as Dataproc, Dataflow, BigQuery, etc.
  - Access to AutoML, pre-trained APIs for video, vision, NLP, etc.

Vertex AI – Components



Vertex AI – Demo

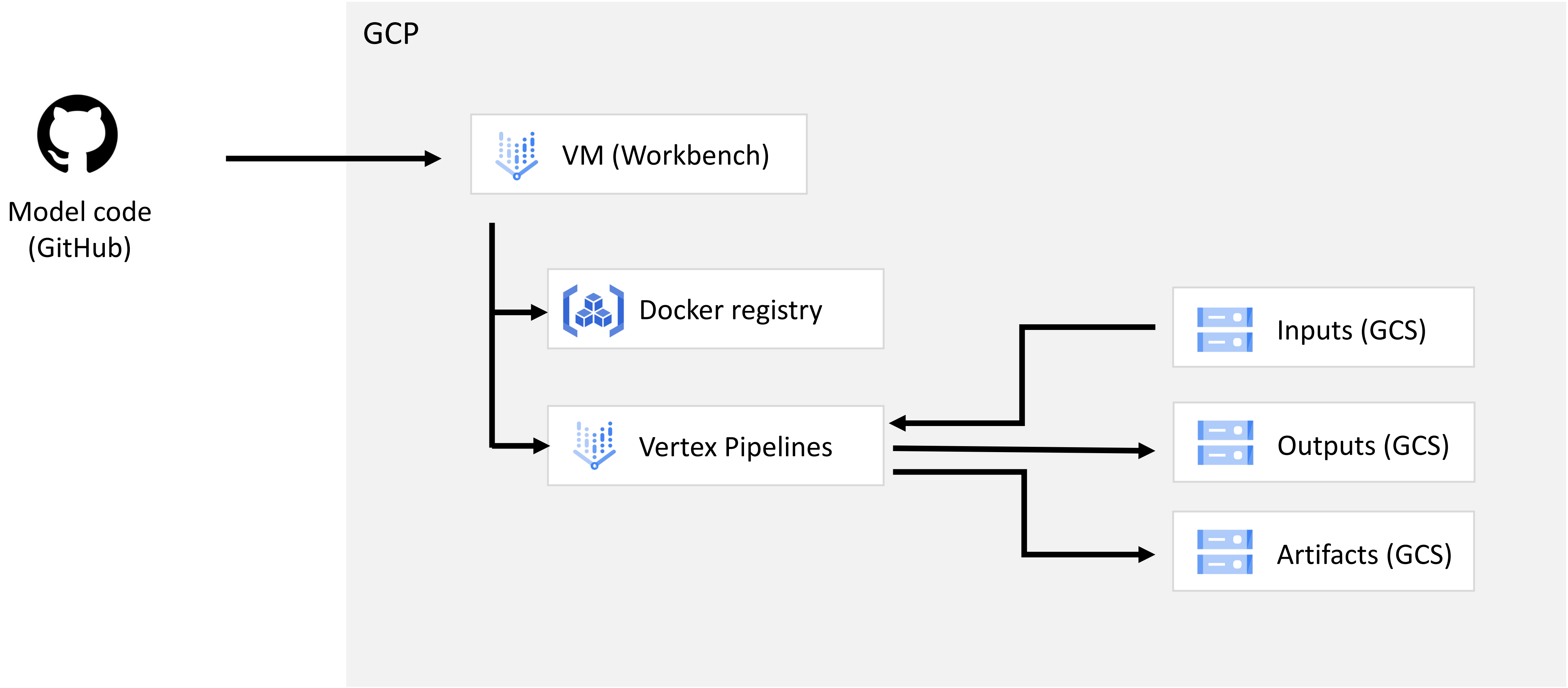


Getting started

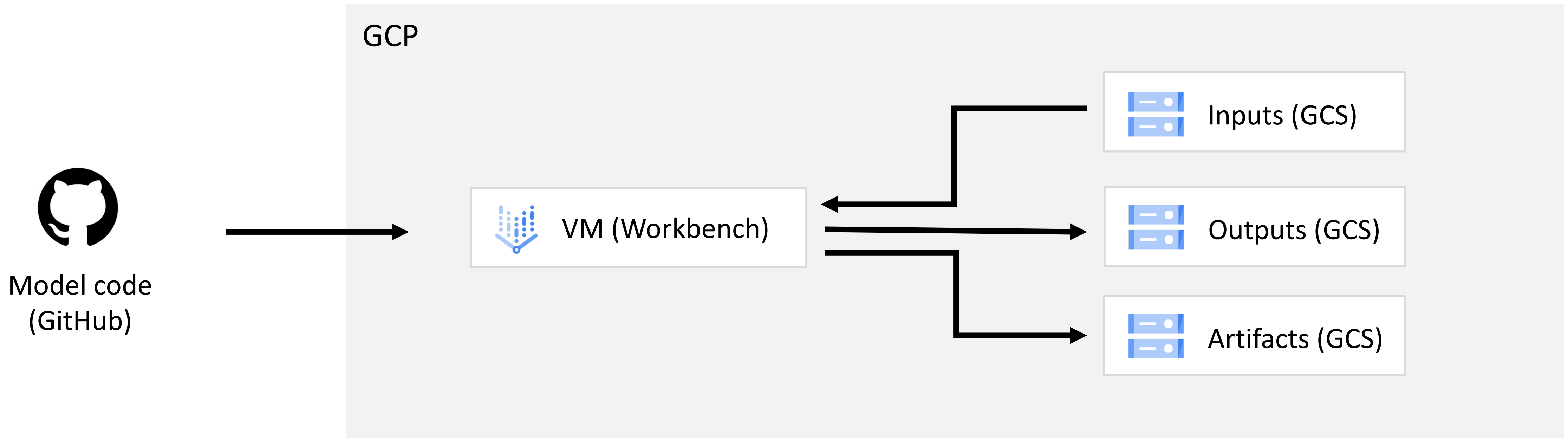
Use case

- You’re working at Fancy Fashion, a sustainable fashion start-up with an app that helps people sell and share second-hand clothing
- A key part of the app is an ML model that automatically analyses uploaded images to automatically assign labels to fashion article
- As a PoC, we’re working on a model that takes these images and classifies them into preset categories (e.g. bag, sneaker, etc.)

Todays target: building a training pipeline

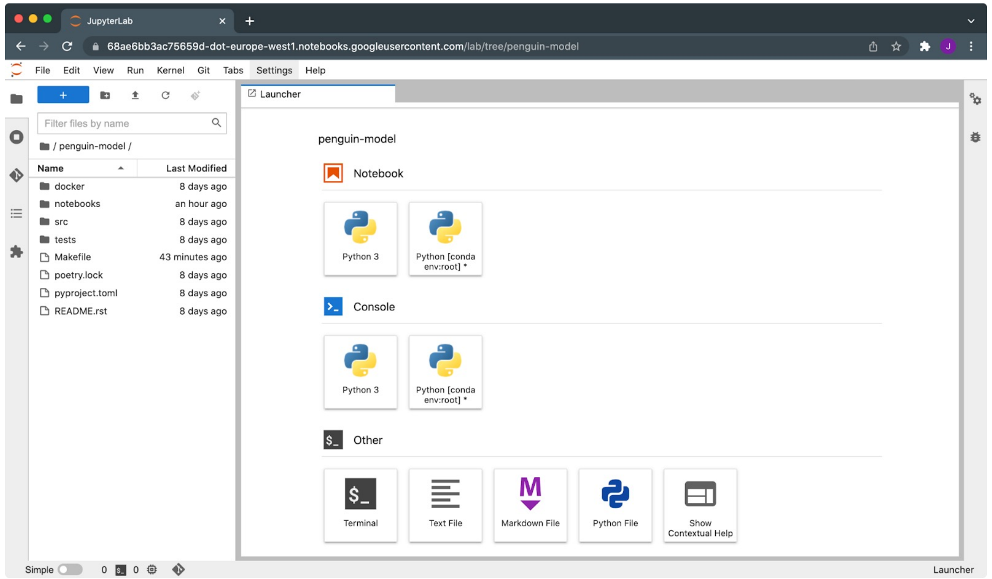


First: running the model interactively



What is Vertex Workbench?

- Provides managed VMs with JupyterLab and many common ML frameworks
- Software can be customized with custom VM images or bootstrap script
- Supports different machine sizes + GPUs
- Access to other resources managed transparently using service account
- Accessible over HTTPS (for Jupyter) and SSH (for IDE integration)



Exercise: Using Vertex Workbench

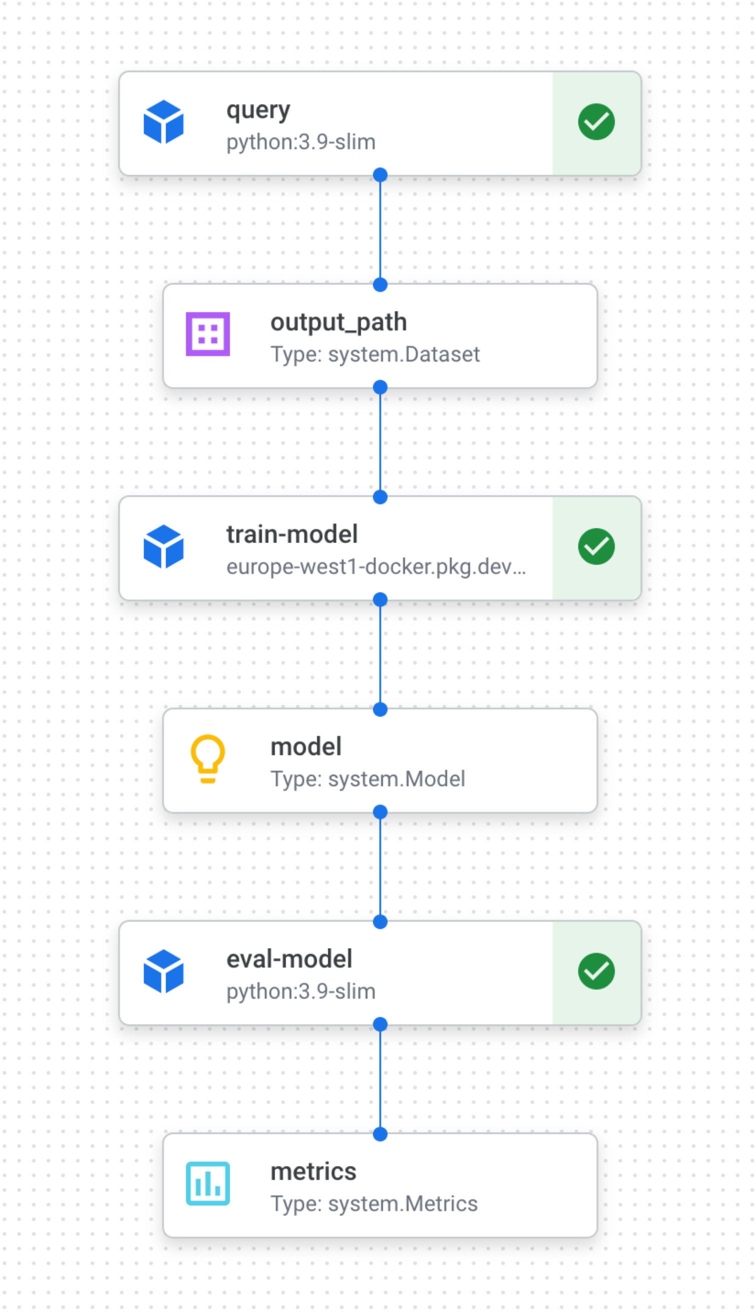
- Open the Google Cloud Console in your browser (https://console.cloud.google.com)
- Select the ‘gdd-cb-vertex’ project (top-left)
- Navigate to the ’Vertex AI’ section and open the Workbench tab
- Start your VM and open Jupyter Lab
- Clone our repository for the hackathon using git:
    - git@github.com:jrderuiter/code-breakfast-vertex-ai.git
- Run through the first exercise (training a model locally)

GO DATA DRIVEN

Kubeflow pipelines

What is Kubeflow Pipelines?

- Open-source pipeline SDK for building workflows as DAGs of containerized tasks
- Provides tracking of produced artifacts in a coupled metadata store
- Can be run in Vertex AI using Vertex Pipelines, metadata stored in the ML metadata store



Defining tasks as components

```
@component(
    base_image="python:3.9-slim",
    packages_to_install=["google-cloud-bigquery", "pandas", "pyarrow"],
    output_component_file="_artifacts/query.yaml",
)
def fetch_bigquery(
    query: str, output_path: OutputPath("Dataset"), project_id: Optional[str] = None
) -> None:
    """Runs a query on BigQuery."""

    from google.cloud import bigquery

    client = bigquery.Client(project=project_id)
    job = client.query(query)

    df = job.to_dataframe()
    df.to_parquet(output_path)
```

Combining tasks into a pipeline

```
@kfp.dsl.pipeline(name="penguin")
def pipeline():
    fetch_task = fetch_bigquery(
        "SELECT * FROM bigquery-public-data.ml_datasets.penguins",
        project_id=GCP_PROJECT_ID,
    )

    train_task = (
        train_model(fetch_task.outputs["output_path"])
        # Docs: https://www.kubeflow.org/docs/distributions/gke/pipelines/enable-gpu-and-tpu/
        .set_gpu_limit(1).add_node_selector_constraint(
            "cloud.google.com/gke-accelerator", "nvidia-tesla-k80"
        )
    )

    eval_model(train_task.outputs["model"])
```

Compiling and running the pipeline

```
compiler.Compiler().compile(
    pipeline_func=pipeline,
    package_path="_artifacts/pipeline.json",
)
```

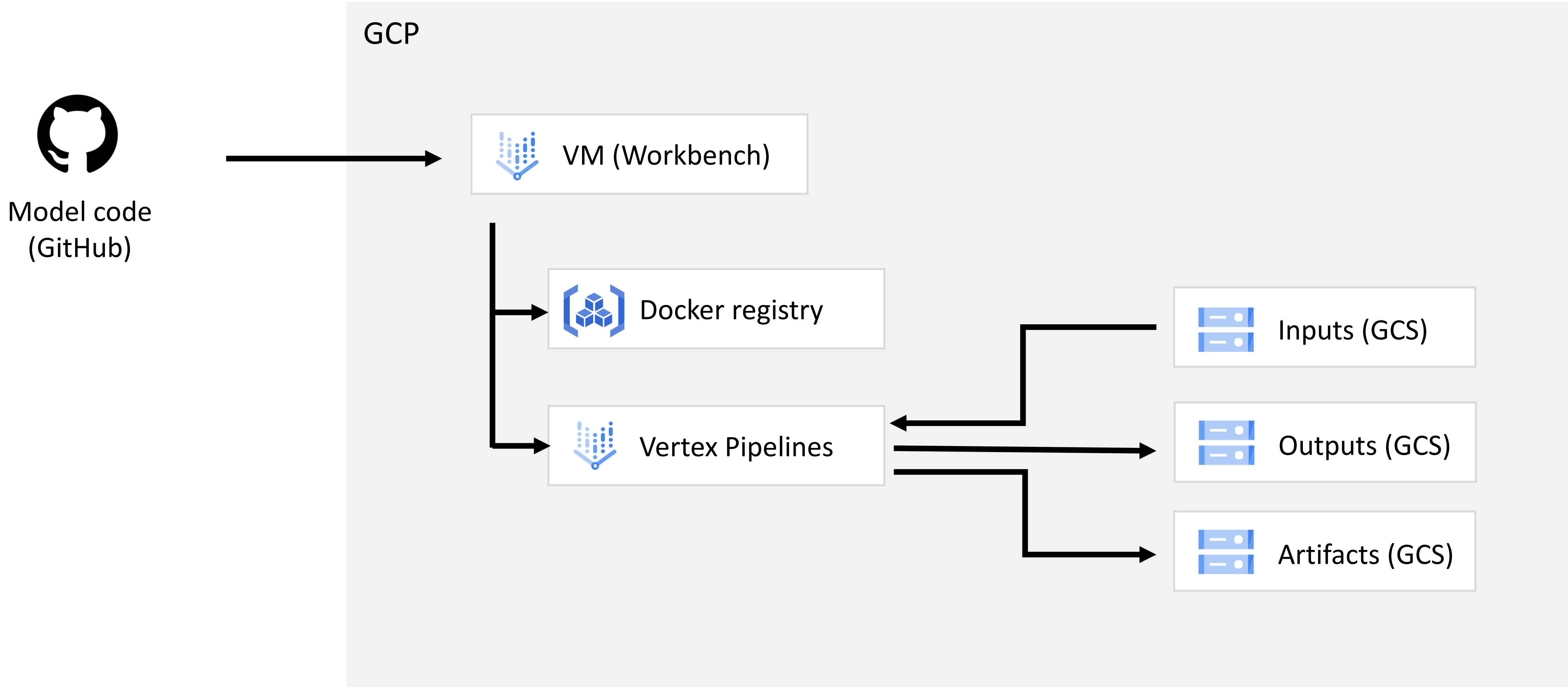
```
from google.cloud.aiplatform.pipeline_jobs import PipelineJob

job = PipelineJob(
    display_name=f"fancy-fashion-{USER_NAME}",
    enable_caching=False,
    template_path="_artifacts/pipeline.json",
    parameter_values={
        "train_path": "gs://gdd-cb-vertex-fashion-inputs/train"
    },
    pipeline_root=f"gs://gdd-cb-vertex-fashion-artifacts/pipelines",
    location=GCP_REGION,
)

job.run(
    service_account=f"vmd-fashion@gdd-cb-vertex.iam.gserviceaccount.com"
)
```

Hackathon: Building a Kubeflow pipeline

- Open the second notebook (2-run-pipeline.ipynb)
- Build and push a Docker image for the package
- Run the provided pipeline and view the results
- Extend the pipeline to include evaluation + prediction steps

GO DATA DRIVEN

Hackathon: Building an ML pipeline



Questions?