Team Details:

Group: 108

- **Akash Verma - 2024AA05466**
- **Rohit Garg - 2024AA05835**
- **PRAVEEN CHANDRAKUMAR MEENAKSHI MULCHANDA - 2024AB05115**
- **BHARADWA NAYANKUMAR PRAFULLABHAI - 2024AA05492**
- **KUMAR VINEETH - 2024AA05620**

MLOps Design Document: Cat vs. Dog Image Classification Pipeline

Github: https://github.com/rtgarg1991/MLOps_Assignment2/tree/main

System Architecture Overview

It implements a separation of concerns, partitioning the lifecycle into a centralized orchestration layer and a high-performance execution environment.

- **Control Plane (GitHub Actions):** This plane serves as the orchestration and CI/CD engine. Its objective is to manage the lifecycle of code-driven events, automate quality gates, and trigger infrastructure-level operations within the data plane.
- **Data Plane (Google Cloud Platform):** This plane acts as the persistence and compute layer. Its objective is to execute resource-intensive training workloads, manage the artifact lifecycle, and provide a scalable environment for model inference and observability.

Control Plane: CI/CD Pipeline (GitHub Actions)

The automated workflow is initiated by developer actions (push/PR) and follows a sequence of stages to ensure code and artifact integrity:

- Quality Assurance: The pipeline begins with the Lint-and-test phase, ensuring that only code meeting predefined standards proceeds to the build stage.
- Trainer Image Management: Following QA, the system executes the build trainer image and push trainer image (push only) steps. These package the training logic into immutable containers for the execution environment.
- Training Orchestration: The pipeline triggers the submit GKE training job, which hands off the execution context to the Kubernetes cluster.
- Serving Image Management: The pipeline handles the inference logic through the build serving image and push serving image steps, staging the API container for deployment.

- Deployment and Validation: The final stage is the deploy API to GKE step.

Data Plane: Google Cloud Infrastructure

The Google Cloud Platform (GCP) infrastructure provides the necessary compute and storage resources to facilitate the MLOps lifecycle.

- Artifact Registry: This component serves as the decoupling point between the CI pipeline and the CD environment, staging container images that are pulled by GKE nodes during job execution and deployment.
- GKE Namespace: This managed Kubernetes environment houses all active workloads, including the Training Job Pod, the cats-dogs-api Deployment, the cats-dogs-api LoadBalancer Service, and the PodMonitoring scrape: /metrics agent.
- Cloud Storage: GCS provides the underlying persistence layer, organized to support rigorous data and model versioning.

Model Architecture & Training Logic

- **Model Architecture:** The system employs a Simple CNN (Convolutional Neural Network) designed for binary classification (Cat vs. Dog).
- **Artifact Format:** Upon successful completion of the training phase, the job outputs a `model.pt` (PyTorch) artifact to the versioned models path in Google Cloud Storage.
- **Training Execution:** Training is executed as a GKE Training Job Pod, which manages the end-to-end data-to-model transformation.

Data Management & Versioning

The infrastructure ensures data traceability through a structured hierarchy in Google Cloud Storage (GCS)

- **DVC Integration:** Data versioning is managed via DVC (Data Version Control) at multiple stages.
- **Artifact Registry:** All Docker images (Trainer and Serving) are published to the GCP Artifact Registry, serving as the immutable staging point for GKE deployments.

Training Workflow and Data Versioning

The Training Job Pod within GKE manages the end-to-end data-to-model transformation logic. This process involves specific state transitions within the GCS hierarchy to ensure traceability:

- Ingestion: The pod reads data from the raw path and writes the versioned output to the ingested directory. The version is managed via DVC
- Preprocessing: The pod reads the versioned ingested data, applies transformations, and persists the result to the processed directory. The version is managed via DVC
- EDA: data exploration is performed to explore the dataset and understand the distribution.
- Model Output: Upon completion of training on the processed data, the job outputs the final model.pt artifact to the versioned models path.

Model Serving and API Infrastructure

The production inference environment is designed for high availability and automated verification.

- Deployment Specs: The cats-dogs-api Deployment is configured to provide redundancy and ensure service availability during node maintenance or scaling events.
- Traffic Management: The cats-dogs-api LoadBalancer Service exposes the deployment to external traffic, managing request distribution across the active replicas.

Experiment Tracking and Monitoring

It integrates observability to maintain a continuous feedback loop between training and serving.

- MLflow Integration: The mlflow-service acts as a centralized tracking server, mediating communication between the Training Job Pod and the mlflow-artifacts/ GCS bucket for experiment logging.
- Observability: For real-time monitoring, the PodMonitoring component performs a scrape of the /metrics endpoint on the API pods, capturing performance and health telemetry.
- Tracking Components:
    - mlflow-service: Manages experiment metadata and parameter logging.

    - PodMonitoring (google-pod-monitor): Facilitates real-time metrics collection within the GKE namespace.
    - External UIs: Provide human-in-the-loop oversight of metrics and experiment history.

CI/CD Pipeline & Automated Validation

The GitHub Actions workflow automates the transition from code change to a verified production API.

- **Quality Assurance:** Executes linting and unit tests
- **Image Management:** Builds and pushes the Simple CNN trainer and serving images to the GCP Artifact Registry.
- **Deployment:** Deploys the inference API to the GKE cluster
- **Smoke Test:** After deployment, the pipeline pauses to ensure the API is fully available. It then executes a validation suite
    - **Health Check:** Calls the `/health` endpoint to verify 2XX responses.
    - **Inference Validation:** Sends a sample generated RGB image to the `/predict` endpoint to ensure the `model.pt` artifact is loading and responding correctly.

Observability

The production environment is optimized for high availability and continuous monitoring.

- **Monitoring Stack: API Monitoring:** Utilizes GCP-native observability tools (GCP Observability Monitoring) for real-time telemetry .
- **Experiment Tracking: MLflow** tracks experiment metadata and logs parameters between the training pod and GCS.