

# Multimodal Reasoning AI Agent

Upload an image and provide a reasoning-based task for the AI Agent. The AI Agent will analyze the image and respond based on your input.

Upload Image



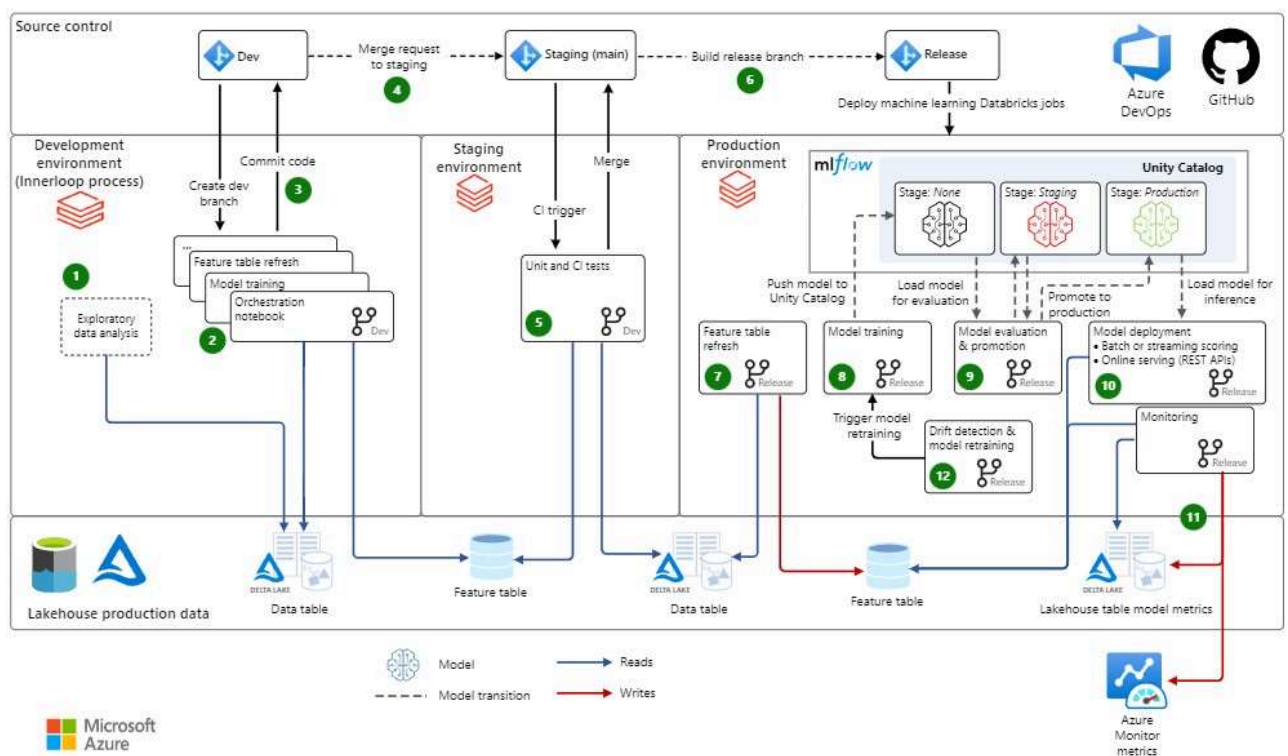
Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files



MLOps.png 77.6KB



Uploaded Image

Enter your task/question for the AI Agent:

generate code for this data flow

Analyze Image

AI Response:

```

# This code provides a high-level representation of the data flow.
# It's not meant to be directly executable but rather to illustrate the stages
# and data movement depicted in the image.

# --- Source Control ---
print("--- Source Control ---")
print("1. Developer works on a feature branch (Dev)")
print("2. Developer commits code changes")
print("3. Merge request to staging (main) branch")

# --- Development Environment (Innerloop process) ---
print("\n--- Development Environment ---")

# Stage 1: Exploratory Data Analysis
print("1. Exploratory data analysis on Lakehouse production data")
# Assume reading data from a Delta Lake table using a library like pyspark
# from pyspark.sql import SparkSession
# spark = SparkSession.builder.appName("EDA").getOrCreate()
# production_data = spark.read.format("delta").load("dbfs:/mnt/lakehouse/productio
# production_data.show()

# Stage 2: Feature Table Refresh, Model Training, Orchestration Notebook
print("2. Feature table refresh (writes to Feature table)")
# Assume writing to a Delta Lake table
# feature_data = production_data.select("feature1", "feature2", ...)
# feature_data.write.format("delta").mode("overwrite").save("dbfs:/mnt/lakehouse/f
print("    Model training (reads Feature table, writes Model)")
# Assume using a machine learning library like scikit-learn or TensorFlow
# from sklearn.linear_model import LogisticRegression
# model = LogisticRegression()
# model.fit(feature_data.toPandas()[["feature1", "feature2"]], feature_data.toPanda
print("    Orchestration notebook (manages Feature refresh and Model training)")

# --- Staging Environment ---
print("\n--- Staging Environment ---")
print("4. Merge to Staging (main) triggers CI")
print("5. Unit and CI tests (reads Feature table, uses Model for testing)")
# Assume running unit tests on data transformations and model predictions
# def test_model_predictions(model, test_data):
#     predictions = model.predict(test_data)
#     assert len(predictions) > 0

# --- Production Environment ---
print("\n--- Production Environment ---")

# Stage 6: Build Release Branch & Deploy
print("6. Build release branch from Staging")
print("    Deploy machine learning Databricks jobs")

```

```

# MLflow and Unity Catalog
print("\n--- MLflow and Unity Catalog ---")
print("    MLflow tracks model versions, experiments, and runs")
print("    Unity Catalog manages model lineage and access control")

# Stage 7: Feature Table Refresh (Production)
print("7. Feature table refresh (writes to Feature table)")
# Similar to development, but in the production environment
# production_feature_data = spark.read.format("delta").load("dbfs:/mnt/production/

# Stage 8: Model Training (Production)
print("8. Model training (reads Feature table, writes Model)")
# Retraining model in production
# retrained_model = LogisticRegression()
# retrained_model.fit(production_feature_data.toPandas()[["feature1", "feature2"]])

# Model Transition through Unity Catalog
print("    Push model to Unity Catalog (Stage: None)")
print("    Load model for evaluation (Stage: Staging)")
print("9. Model evaluation & promotion (reads Feature table, uses Model)")
# Evaluate model performance
# from sklearn.metrics import accuracy_score
# predictions = retrained_model.predict(evaluation_data)
# accuracy = accuracy_score(actual_labels, predictions)
print("    Promote to production (Stage: Production)")

# Stage 10: Model Deployment
print("10. Model deployment:")
print("    - Batch or streaming scoring (reads Feature table, uses Model)")
# Batch inference example
# predictions = retrained_model.predict(new_data)
print("    - Online serving (REST APIs) (uses Model)")
# Deploying model as a REST API using MLflow or other serving frameworks

# Stage 11: Monitoring
print("11. Monitoring (reads Lakehouse table model metrics)")
# Assume using Azure Monitor to collect and visualize metrics
# print("Metrics available in Azure Monitor")

# Stage 12: Drift Detection & Model Retraining
print("12. Drift detection & model retraining (reads Feature table, Lakehouse tabl
# Logic to detect data drift or model performance degradation
# if drift_detected:
#     print("Trigger model retraining (back to stage 8)")

# --- Data Storage ---
print("\n--- Data Storage ---")
print("Lakehouse production data (Delta Lake)")

```

```
print("Data table (Delta Lake)")
print("Feature table (Delta Lake)")
print("Lakehouse table model metrics (Delta Lake)")

# --- Monitoring ---
print("\n--- Monitoring ---")
print("Azure Monitor metrics")
```