PRJ-MLS-042: Centralized ML Feature Store

Certification: AWS Certified Machine Learning – Specialty
Domain: Data Engineering and Feature Engineering

1. Project Overview

This project demonstrates how to solve a common and critical problem in MLOps: **training-serving skew**. This skew occurs when the features used to train a model differ from the features used for inference in production, leading to a significant drop in model performance. **Amazon SageMaker Feature Store** is a fully managed repository that solves this by providing a centralized place to store, retrieve, share, and manage curated ML features for the entire lifecycle.

We will design and implement a feature store that serves both batch training and real-time inference. We will create a **Feature Group** and populate it with data. The Feature Store automatically creates two storage tiers: an **Online Store** for low-latency, real-time feature retrieval, and an **Offline Store** that archives all historical feature data in S3. We will then show how a training pipeline can use the Offline Store to build a training dataset, and how a real-time inference endpoint can use the Online Store to enrich incoming data before making a prediction.

Key Objectives

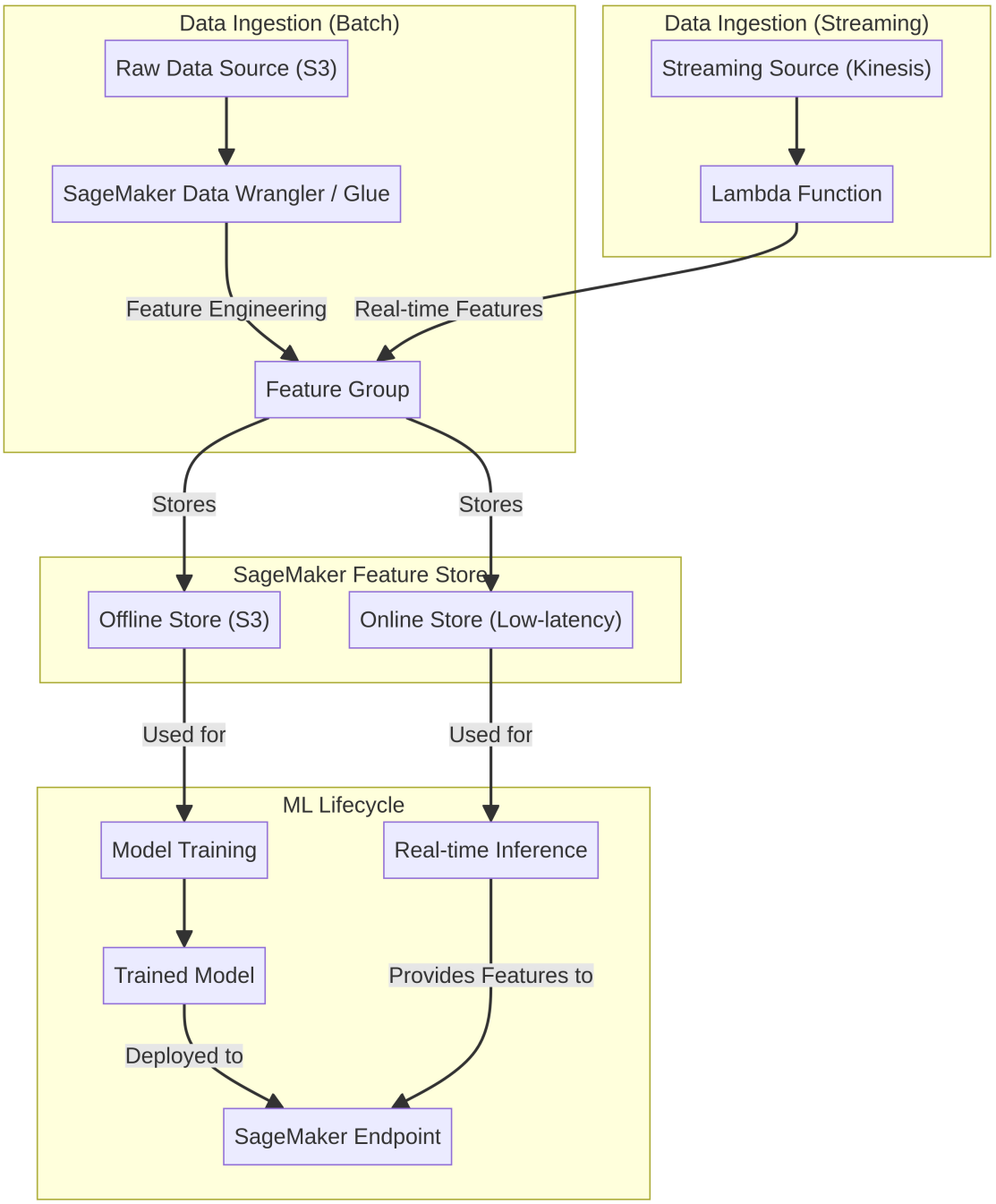
- Understand the problem of training-serving skew and how a feature store solves it.
- Create a Feature Group in SageMaker Feature Store.
- Ingest data into the Feature Store from both batch (e.g., a CSV file) and streaming (e.g., Kinesis) sources.
- Use the Offline Store to create a point-in-time correct training dataset.

- Use the Online Store for low-latency feature lookups during real-time inference.
- Ensure consistency of features between training and serving environments.

---

2. Architecture

The architecture revolves around the Feature Group, which acts as the single source of truth for ML features, serving both the training and inference pipelines.



Architectural Components:

1. **Data Ingestion:**
    - **Batch Ingestion:** A scheduled job (e.g., using **SageMaker Data Wrangler** or **AWS Glue**) runs periodically. It pulls raw data from a source like S3, performs feature engineering (e.g., calculating aggregations, creating embeddings), and ingests the computed features into the **Feature Group**.
    - **Streaming Ingestion:** For real-time features (e.g., a user's most recent activity), a streaming source like **Amazon Kinesis** can trigger a **Lambda function**. The Lambda performs the feature transformation and ingests the data into the Feature Group in real-time.
2. **SageMaker Feature Store:**
    - The **Feature Group** is the core component. It defines the schema for a set of related features (e.g., all features for a `customer` entity).
    - When data is ingested, the Feature Store automatically writes it to two destinations:
        - **Online Store:** A low-latency, key-value database designed for fast lookups (single-digit millisecond latency). It always holds the latest value for each feature record.
        - **Offline Store:** A historical archive of all feature data, stored in an S3 bucket. It is organized by event time, allowing for point-in-time queries.
3. **ML Lifecycle Integration:**
    - **Model Training:** A data scientist uses the **Offline Store** to build a training dataset. They can query the data "as of" a specific time, ensuring that the training data does not contain future information that would cause data leakage. SageMaker provides tools to easily join feature data with ground truth labels.
    - **Real-time Inference:** A trained model is deployed to a **SageMaker Endpoint**. When the endpoint receives an inference request (e.g., containing just a `customer_id`), the inference code first makes a `GetRecord` call to the **Online Store**. It retrieves the latest features for that

customer, combines them with the request payload, and then passes the complete feature vector to the model for prediction.

This architecture guarantees that the feature `avg_purchase_value` used during training is derived and accessed in the exact same way as the `avg_purchase_value` feature used during inference.

---

3. Prerequisites

- An AWS account with administrative permissions.
- A dataset to use for creating features (e.g., a CSV file in S3).
- A SageMaker Studio environment or a configured SageMaker Notebook instance.

---

4. Step-by-Step Implementation Guide

Step 4.1: Define and Create the Feature Group

1. In a SageMaker Notebook, import the necessary libraries.

```
import boto3
import sagemaker
from sagemaker.feature_store.feature_group import FeatureGroup


sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
```

2. **Define the Feature Group Schema:**
    - Define the names and data types of the features you will store. You must also specify a `RecordIdentifierFeatureName` (the primary key for your records) and an `EventTimeFeatureName` (the timestamp when the feature was generated).

```
feature_group_name = "customer-features"
feature_group = FeatureGroup(name=feature_group_name,
sagemaker_session=sagemaker_session)


feature_definitions = [
    {"FeatureName": "customer_id", "FeatureType": "String"},
    {"FeatureName": "age", "FeatureType": "Integral"},
    {"FeatureName": "avg_monthly_spend", "FeatureType": "Fractional"},
    {"FeatureName": "is_prime_member", "FeatureType": "Integral"},
    {"FeatureName": "last_updated_time", "FeatureType": "String"}
]
```

3. **Create the Feature Group:**

```
feature_group.create(

s3_uri=f"s3://{sagemaker_session.default_bucket()}/{feature_group_name}",
    record_identifier_name="customer_id",
    event_time_feature_name="last_updated_time",
    feature_definitions=feature_definitions,
    role_arn=role,
    online_store_config={"EnableOnlineStore": True}
)
```

This will provision the necessary resources. Wait for the `CreationStatus` to become `Created`.

Step 4.2: Ingest Data into the Feature Group

1. **Prepare your data:** Load your data into a Pandas DataFrame that matches the feature definitions.
2. **Ingest the data:** Use the `ingest` method of the feature group.

```
# Assuming 'customer_df' is your Pandas DataFrame
feature_group.ingest(data_frame=customer_df, max_workers=3, wait=True)
```

This will ingest the data into both the Online and Offline stores.

Step 4.3: Use the Offline Store for Training

1. **Create a Training Dataset:** Use the `athena_query()` method to get a point-in-time correct snapshot of the features.

```
offline_store_query = feature_group.athena_query()
offline_store_table = offline_store_query.table_name

# Example: Select features as of a specific date
query_string = f'SELECT * FROM "{offline_store_table}" WHERE
time_travel(to_date(\'2023-10-26\'))'

# You can now use this query with Athena or SageMaker Data Wrangler
# to join with your labels and create a final training set.
```

Step 4.4: Use the Online Store for Inference

1. **Deploy a Model:** Train a model using the dataset from the previous step and deploy it to a SageMaker Endpoint.
2. **Write the Inference Script (`inference.py`):**
    - The inference script will receive a request (e.g., with just a `customer_id`).
    - It will use the Boto3 client to call `get_record` on the Feature Store's Online Store.
    - It will then combine the retrieved features with the input and pass them to the model.

```
import boto3
import json


sagemaker_featurestore_runtime = boto3.client("sagemaker-featurestore-runtime")


def handler(event, context):
    customer_id = event["customer_id"]


    # Retrieve features from the Online Store
    feature_record = sagemaker_featurestore_runtime.get_record(
        FeatureGroupName="customer-features",
        RecordIdentifierValueAsString=str(customer_id)
    )


    # Extract features and prepare the model input
    # ... (logic to parse feature_record and create the feature vector)


    # Invoke the model
    # ... (logic to call the loaded model for prediction)


    return prediction_result
```

---

5. Best Practices

- **Feature Governance:** Use tags and descriptions to document your features. The Feature Store provides a searchable catalog, making features discoverable across teams.
- **Tiering Features:** Not all features need to be in the Online Store. For features only used in batch training, you can disable the online store for that feature group to save costs.

- **Monitoring:** Monitor the age of your features in the Online Store to detect if your ingestion pipelines are stale.

---

6. Cleanup

1. **Delete the SageMaker Endpoint and Model.**
2. **Delete the Feature Group:**

```
feature_group.delete()
```

3. **Clean up the Offline Store data** from the S3 bucket.
4. Stop the SageMaker Notebook instance.

Business Context

The Problem

Organizations want to leverage AI/ML for business insights but lack the expertise to build secure, scalable ML systems. ML projects fail due to poor data quality, model drift, and lack of monitoring. Security and compliance requirements for ML systems are often overlooked.

The Solution

Production-ready ML system with automated data pipelines, model training, deployment, and monitoring. Implements MLOps best practices with version control, A/B testing, and automated retraining. Ensures data privacy and model security throughout the ML lifecycle.

Business Value

- **Business Intelligence:** AI-driven insights improve decision-making and outcomes

- **Operational Efficiency:** Automated ML workflows reduce manual data science effort by 60%
- **Model Reliability:** Continuous monitoring detects and corrects model drift automatically
- **Compliance Assurance:** Built-in controls for data privacy and model governance

Risk Mitigation

Protects sensitive data in ML pipelines, prevents biased or inaccurate models, ensures model explainability, and maintains compliance with AI regulations.

GRC Mapping

Compliance Frameworks

- **NIST AI RMF:** Trustworthy and responsible AI
- **ISO/IEC 23894:** AI risk management
- **NIST CSF:** PR.DS-2 (Data in transit), PR.DS-5 (Data leak protection)
- **Responsible AI Principles:** Fairness, accountability, transparency, ethics

Security Controls Implemented

- Data anonymization and pseudonymization
- Model explainability and interpretability
- Bias detection and mitigation
- Model versioning and rollback
- Automated model monitoring and alerting

Audit Evidence

- Model cards documenting intended use and limitations
- Bias and fairness evaluation reports
- Model performance metrics over time

- Data processing and transformation logs

Regulatory Alignment

- **GDPR:** Article 22 (Right to explanation), Article 25 (Privacy by design)
- **AI Act (EU):** Transparency and accountability requirements
- **CCPA:** Data privacy for ML training data
- **SOC 2:** CC6.1 (Access to sensitive data)