MLOPS pipeline with spark streaming and Kafka on cloudera

```
from pyspark.sql.functions import from_json, col, explode
from pyspark.sql.types import StructType, StructField, StringType, DoubleType
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
import mlflow
from datetime import datetime, timedelta
```

Define constants

```
SPARK_APP_NAME = "MLOps-Spark-Streaming-Kafka"
SPARK_MASTER = "local[*]"
SPARK_BATCH_INTERVAL = 1800 # 30 minutes
MODEL_PATH = "/path/to/model"
FEATURE_COLUMNS = ["feature1", "feature2", "feature3", "feature4"]
KAFKA_BOOTSTRAP_SERVERS = "kafka-broker1:9092,kafka-broker2:9092"
KAFKA_TOPIC = "my-topic"
KAFKA_STARTING_OFFSETS = "earliest"
KAFKA_CHECKPOINT_LOCATION = "/path/to/checkpoint"
```

Define schema for incoming Kafka messages

```
kafka_schema = StructType([
StructField("label", DoubleType()),
StructField("feature1", DoubleType()),
StructField("feature2", DoubleType()),
StructField("feature3", DoubleType()),
StructField("feature4", DoubleType())
])
```

Create Spark session

```
spark = SparkSession.builder \
.appName(SPARK_APP_NAME) \
.master(SPARK_MASTER) \
.getOrCreate()
```

Create MLflow experiment

mlflow.set_experiment("MLOps-Spark-Streaming-Kafka")

Define Kafka data stream source and preprocessing

raw_stream = spark.readStream.format("kafka") \

.option("kafka.bootstrap.servers", KAFKA_BOOTSTRAP_SERVERS) \

.option("subscribe", KAFKA_TOPIC) \

.option("startingOffsets", KAFKA_STARTING_OFFSETS) \

.load()

preprocessed_stream = raw_stream.selectExpr("CAST(value AS STRING)") \

.select(from_json(col("value"), kafka_schema).alias("json")) \

.select("json.*")

Define model training function

def train_model():

with mlflow.start_run():

 # Define model and training parameters

lr = LogisticRegression()

assembler = VectorAssembler(inputCols=FEATURE_COLUMNS, outputCol="features")

evaluator = BinaryClassificationEvaluator()

 # Split preprocessed data into training and testing sets

train_data, test_data = preprocessed_stream.randomSplit([0.7, 0.3], seed=123)

 # Assemble feature vector

train_data = assembler.transform(train_data).select("features", "label")

test_data = assembler.transform(test_data).select("features", "label")

 # Train model

model = lr.fit(train_data)

 # Evaluate model

auc = evaluator.evaluate(model.transform(test_data))

mlflow.log_metric("auc", auc)

 # Save model

model.write().overwrite().save(MODEL_PATH)

mlflow.log_artifact(MODEL_PATH, "model")

Define streaming query and start it

```
training_query = preprocessed_stream.writeStream \
.trigger(processingTime=f"{SPARK_BATCH_INTERVAL} seconds") \
.option("checkpointLocation", KAFKA_CHECKPOINT_LOCATION) \
.foreachBatch(lambda batch_df, batch_id: train_model()) \
.start()
```

Wait for the query to terminate

```
training_query.awaitTermination()
```

we read streaming data from Kafka and preprocess it using a specified schema. We then define a training function that trains

Code for an MLOps pipeline which reads a Kafka topic into Spark structured streaming, performs classification with a classification model, and then writes the output to a new Kafka topic every 30 minutes

```
from pyspark.sql.functions import from_json, col, to_json, struct
from pyspark.sql.types import StructType, StructField, StringType, DoubleType
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegressionModel
from pyspark.ml import PipelineModel
from pyspark.sql import SparkSession
from pyspark.streaming import StreamingContext
from kafka import KafkaProducer
from datetime import datetime, timedelta
```

Define constants

```
APP_NAME = "MLOps-Kafka-Spark-Streaming"
KAFKA_BOOTSTRAP_SERVERS = "localhost:9092"
INPUT_TOPIC = "input-topic"
OUTPUT_TOPIC = "output-topic"
BATCH_DURATION = 1800 # 30 minutes
MODEL_PATH = "/path/to/model"
FEATURE_COLUMNS = ["feature1", "feature2", "feature3", "feature4"]
```

Create Spark session and streaming context

```
spark = SparkSession.builder.appName(APP_NAME).getOrCreate()
```

ssc = StreamingContext(spark.sparkContext, BATCH_DURATION)

Define input stream from Kafka topic

kafka_stream = spark \

.readStream \

.format("kafka") \

.option("kafka.bootstrap.servers", KAFKA_BOOTSTRAP_SERVERS) \

.option("subscribe", INPUT_TOPIC) \

.load()

Parse input stream data from JSON to DataFrame

input_schema = StructType([

StructField("feature1", DoubleType()),

StructField("feature2", DoubleType()),

StructField("feature3", DoubleType()),

StructField("feature4", DoubleType())

])

parsed_stream = kafka_stream \

.selectExpr("CAST(value AS STRING)") \

.select(from_json(col("value"), input_schema).alias("parsed_data")) \

.select("parsed_data.*")

Assemble features vector

assembler = VectorAssembler(inputCols=FEATURE_COLUMNS, outputCol="features")

assembled_stream = assembler.transform(parsed_stream).select("features")

Load model

model = PipelineModel.load(MODEL_PATH)

Predict on streaming data and write output to **new Kafka topic**

def classify_and_write_output(batch_df, batch_id):

predictions = model.transform(batch_df)

output_df = predictions.select(to_json(struct(col("*"))).alias("value"))

output_df \

.write \

.format("kafka") \

```
.option("kafka.bootstrap.servers", KAFKA_BOOTSTRAP_SERVERS) \
.option("topic", OUTPUT_TOPIC) \
.save()
```

Create streaming job

```
query = assembled_stream \
.writeStream \
.foreachBatch(classify_and_write_output) \
.trigger(processingTime=str(BATCH_DURATION) + " seconds") \
.start()
```

Wait for the streaming job to finish

```
query.awaitTermination()
```

Here is an example of a Jenkins pipeline that can trigger a Spark job to write data from a Kafka topic to Elasticsearch with Spark structured streaming every 30 minutes:

```
pipeline {
agent any
environment {
SPARK_HOME = "/path/to/spark"
KAFKA_BOOTSTRAP_SERVERS = "kafka-server:9092"
KAFKA_TOPIC = "my-topic"
ELASTICSEARCH_HOST = "elasticsearch-host"
ELASTICSEARCH_PORT = "9200"
ELASTICSEARCH_INDEX = "my-index"
}
stages {
stage('Build') {
steps {
sh "mvn clean package"
}
}
stage('Deploy') {
steps {
```

```
script {
// Submit Spark job to run every 30 minutes
sh "${SPARK_HOME}/bin/spark-submit" \
"--master" "local[*]" \
"--class" "com.example.KafkaToElasticsearch" \
"--packages" "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1,org.elasticsearch:elasticsearch-hadoop:7.14.0" \
"/path/to/your/spark/job.jar" \
"${KAFKA_BOOTSTRAP_SERVERS}" \
"${KAFKA_TOPIC}" \
"${ELASTICSEARCH_HOST}:${ELASTICSEARCH_PORT}/${ELASTICSEARCH_INDEX}"
}
}
}
}
}
```