to prove you understand the startup latency problem of Spark on EKS and to design a way to measure + optimize it.

**Measure Startup Latency**

How long does it take from when Airflow triggers a Spark job → until the Spark driver and executors are actually running and doing work?

Break it into timestamps:

T0 = Airflow task starts

T1 = Pod scheduled by Kubernetes

T2 = Container starts pulling image

T3 = Driver pod ready

T4 = Executors running

T5 = Spark job starts processing data

→ They want you to collect these timings so they can see where the delay is happening.

**Use Proper Tools**

Airflow DAG (with KubernetesPodOperator or SparkKubernetesOperator) should log timestamps.

Kubernetes events + kubectl describe pod show delays (like scheduling or image pulling).

Metrics can be pushed to Prometheus/Grafana or at least logged into Airflow task logs.

**Propose Optimizations**

Reduce cold start delay: pre-pull images, keep a warm pool of nodes, use smaller base images.

Compare cold start vs warm start latency.

Maybe try Spark Operator to see if it manages lifecycle better.

Cold start = default (pods start from scratch, images not cached).

Optimized = using pre-pulled images, node pools kept warm, or Spark Operator.

They want data showing improvement and recommendations to reduce latency.

**Produce metrics + insights.**

Store the timings somewhere (S3, Prometheus, DB).

Maybe make a Grafana/Metabase dashboard.

Share acceptance criteria (e.g., “startup latency < 60s with warm pool”).

**How you can achieve this step by step:**

Step 1: Add timing in Airflow DAG

- Use KubernetesPodOperator (or SparkKubernetesOperator if installed).

- Before submitting, log T0 = Airflow trigger time.

- After submission, log T1 = pod created.

- Write these timestamps into XCom or S3.

Step 2: Capture Kubernetes Events

- Enable kubectl get events --timestamps -n <namespace> in the DAG or sidecar script.

- Collect events like “Pod scheduled”, “Pulling image”, “Started container”.

- These give you T2, T3, T4.

Step 3: Capture Spark driver events

- Spark writes driver logs (stdout).

- Add log parsing to extract when SparkContext starts. That’s T5.

- Optionally enable Spark metrics → Prometheus.

Step 4: Store data

- Push all these timestamps into:

- A Postgres table (job\_id, T0…T5, duration).

- Or even just S3 CSVs.

Step 5: Dashboard/Analysis

- Use Grafana/Metabase to visualize job startup latencies across runs.

------------------------------------

EXAMPLE

------------------------------------

from airflow import DAG

from airflow.providers.cncf.kubernetes.operators.kubernetes\_pod import KubernetesPodOperator

from airflow.utils.dates import days\_ago

import datetime

default\_args = {"owner": "airflow", "retries": 0}

with DAG(

dag\_id="spark\_job\_latency\_tracking",

default\_args=default\_args,

schedule\_interval=None,

start\_date=days\_ago(1),

catchup=False,

) as dag:

# T0: Airflow task starts

def airflow\_start\_callback(context):

t0 = datetime.datetime.utcnow()

print(f"T0 (Airflow task starts): {t0}")

spark\_task = KubernetesPodOperator(

task\_id="spark-on-eks",

name="spark-driver",

namespace="spark-job",

image="your\_ecr\_repo/your\_spark\_image:latest",

cmds=["/bin/sh", "-c"],

arguments=[

"""

echo "T1 (Pod scheduled): $(date -u +%Y-%m-%dT%H:%M:%SZ)"

echo "T2 (Start pulling image): $(date -u +%Y-%m-%dT%H:%M:%SZ)"

# Run spark-submit (driver pod starts here)

/opt/spark/bin/spark-submit \

--master k8s://https://$KUBERNETES\_SERVICE\_HOST:$KUBERNETES\_SERVICE\_PORT \

--deploy-mode cluster \

--class org.example.YourSparkJob \

--conf spark.kubernetes.namespace=spark-job \

--conf spark.kubernetes.container.image=your\_ecr\_repo/your\_spark\_image:latest \

local:///opt/spark/app.jar

echo "T3 (Driver pod ready): $(date -u +%Y-%m-%dT%H:%M:%SZ)"

"""

],

get\_logs=True,

is\_delete\_operator\_pod=True,

startup\_timeout\_seconds=600,

on\_execute\_callback=airflow\_start\_callback,

)

spark\_task

What happens here

T0: Logged when Airflow task starts (via on\_execute\_callback).

T1: Printed as soon as pod starts (in container entrypoint).

T2: Printed before Spark starts pulling driver/executor images.

T3: Logged once driver pod is ready (after spark-submit launches).

T4: You can modify your Spark job to log "Executors running" using SparkListener (see below).

T5: Your Spark job logs "Processing started" when actual computation begins.

Spark job instrumentation (inside Scala/PySpark app)

from pyspark.sql import SparkSession

import datetime

spark = SparkSession.builder.appName("LatencyTestJob").getOrCreate()

print(f"T4 (Executors running): {datetime.datetime.utcnow()}")

df = spark.read.csv("s3a://your-bucket/input.csv", header=True)

print(f"T5 (Job starts processing data): {datetime.datetime.utcnow()}")

df.groupBy("category").count().show()