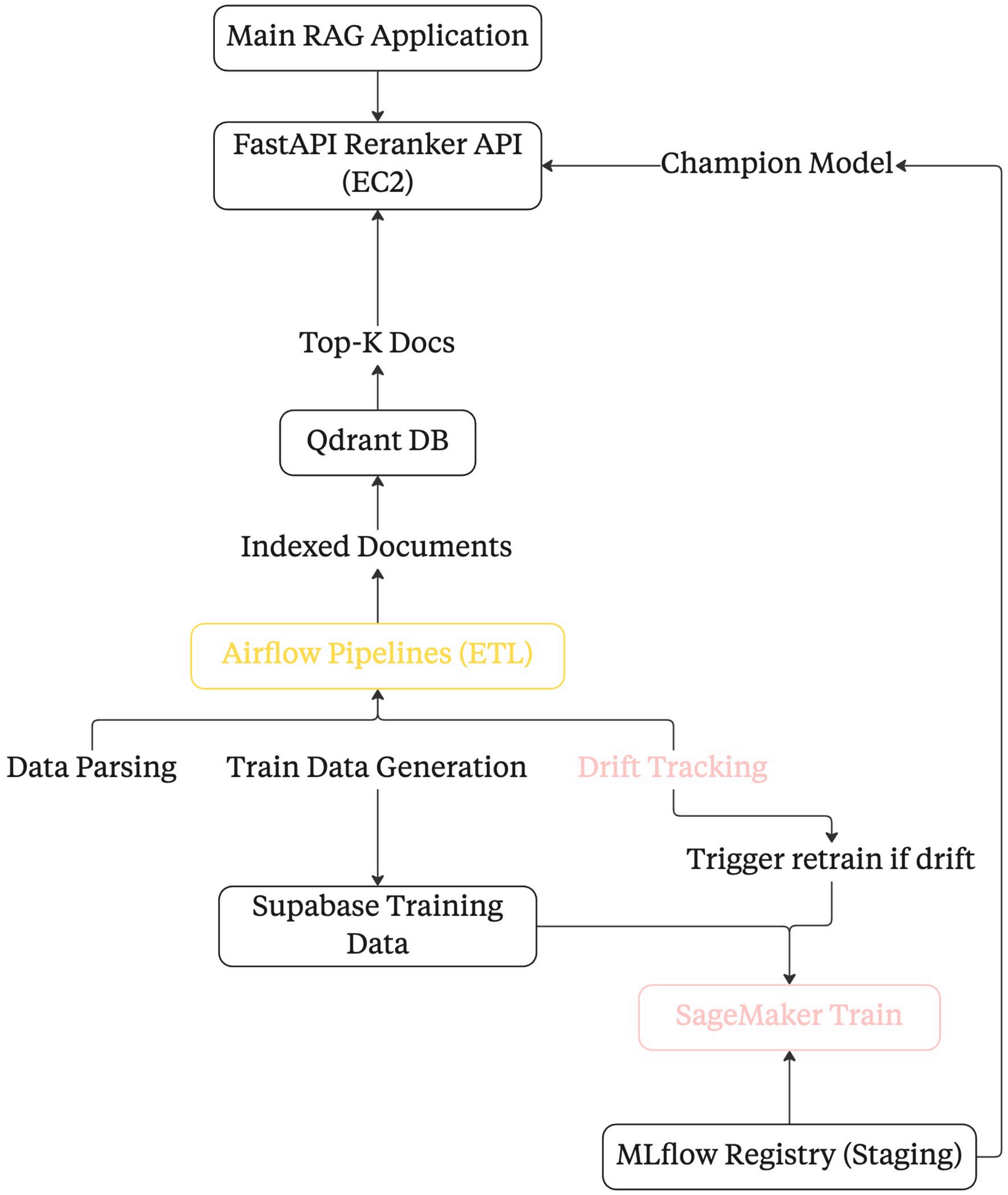
MLOps Lab4 Assignment

Task Summary

In this task, you will need to finish the full pipeline for the project from assignment 1. Your pipeline should include the following: gather the full pipeline of the model on the sagemaker, databricks or Kubeflow, and combine with the previous tasks using you favourite tools for everything (or tools that make the most sense) and prepare a short presentation/demo. The pipeline should include the following steps: data gathering (using airflow or other orchestrator, from prev. assignment), data processing step, model training step, parameters tuning step (this one is optional, if it makes sense), registering the model in model registry, logging all needed metrics, evaluating to detect if it is better then previous model, serving the best model.

Note: all artefacts for lab results demo are located in the `MLOps/reranker_cloud_demo` folder

Final MLOps Pipeline is as follows:



The AWS cloud infrastructure is as follows:

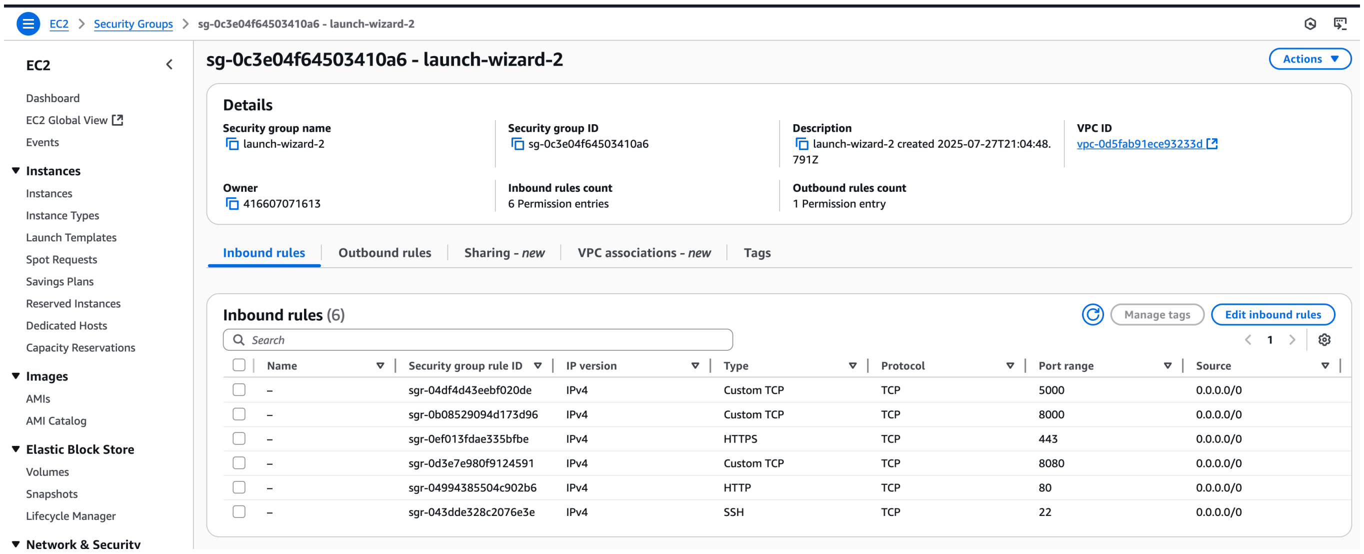
```
┌──────────────────────────────────────────────────────────────┐
│                          AWS CLOUD                           │
├──────────────────────────────────────────────────────────────┤
```

```
│  ┌─────────────────┐  ┌─────────────────┐  ┌───────────────┐│
│  │    SageMaker    │  │       ECR       │  │   S3 Buckets  ││
│  │   Pipelines     │  │  Docker Images  │  │   Storage     ││
│  │                 │  │                 │  │               ││
│  │ • Champion-     │  │ • crossencoder- │  │ • Training    ││
│  │   Challenger    │  │   sagemaker:    │  │   Data        ││
│  │ • Data Drift    │  │   latest        │  │ • Models      ││
│  │   Detection     │  │                 │  │ • Artifacts   ││
│  │ • Auto Training │  │                 │  │               ││
│  └─────────────────┘  └─────────────────┘  └───────────────┘│
│                                                             │
│  ┌─────────────────────────────────────────────────────────┐│
│  │                 EC2 Instance                            ││
│  │              (t3.medium - Production)                   ││
│  │                                                         ││
│  │  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐     ││
│  │  │   Airflow    │ │    MLflow    │ │  FastAPI     │     ││
│  │  │   :8080      │ │    :5000     │ │   :8000      │     ││
│  │  └──────────────┘ └──────────────┘ └──────────────┘     ││
│  │                                                         ││
│  │  ┌──────────────┐ ┌──────────────┐                      ││
│  │  │ PostgreSQL   │ │    MinIO     │                      ││
│  │  │   :5432      │ │  :9000/9001  │                      ││
│  │  └──────────────┘ └──────────────┘                      ││
│  └─────────────────────────────────────────────────────────┘│
└─────────────────────────────────────────────────────────────┘
```

Side Preparations

Firstly we have to create a new EC2 instance with custom security group to open all necessary ports for future services.

Note: base for this EC2 instance is Amazon Linux (random choice, but it will affect the syntax of the following commands)



sg-0c3e04f64503410a6 - launch-wizard-2

Details

Security group name launch-wizard-2	Security group ID sg-0c3e04f64503410a6	Description launch-wizard-2 created 2025-07-27T21:04:48.791Z	VPC ID vpc-0d5fab91ece93233d
Owner 416607071613	Inbound rules count 6 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules (6)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
–	sgr-04df4d43eebf020de	IPv4	Custom TCP	TCP	5000	0.0.0.0/0
–	sgr-0b08529094d173d96	IPv4	Custom TCP	TCP	8000	0.0.0.0/0
–	sgr-0ef013fdae335bfbe	IPv4	HTTPS	TCP	443	0.0.0.0/0
–	sgr-0d3e7e980f9124591	IPv4	Custom TCP	TCP	8080	0.0.0.0/0
–	sgr-04994385504c902b6	IPv4	HTTP	TCP	80	0.0.0.0/0
–	sgr-043dde328c2076e3e	IPv4	SSH	TCP	22	0.0.0.0/0

Then standart connection procedure:

```
chmod 400 cloud-demo-key.pem
ssh -i "cloud-demo-key.pem" ec2-user@ec2-16-170-226-195.eu-north-
1.compute.amazonaws.com
```

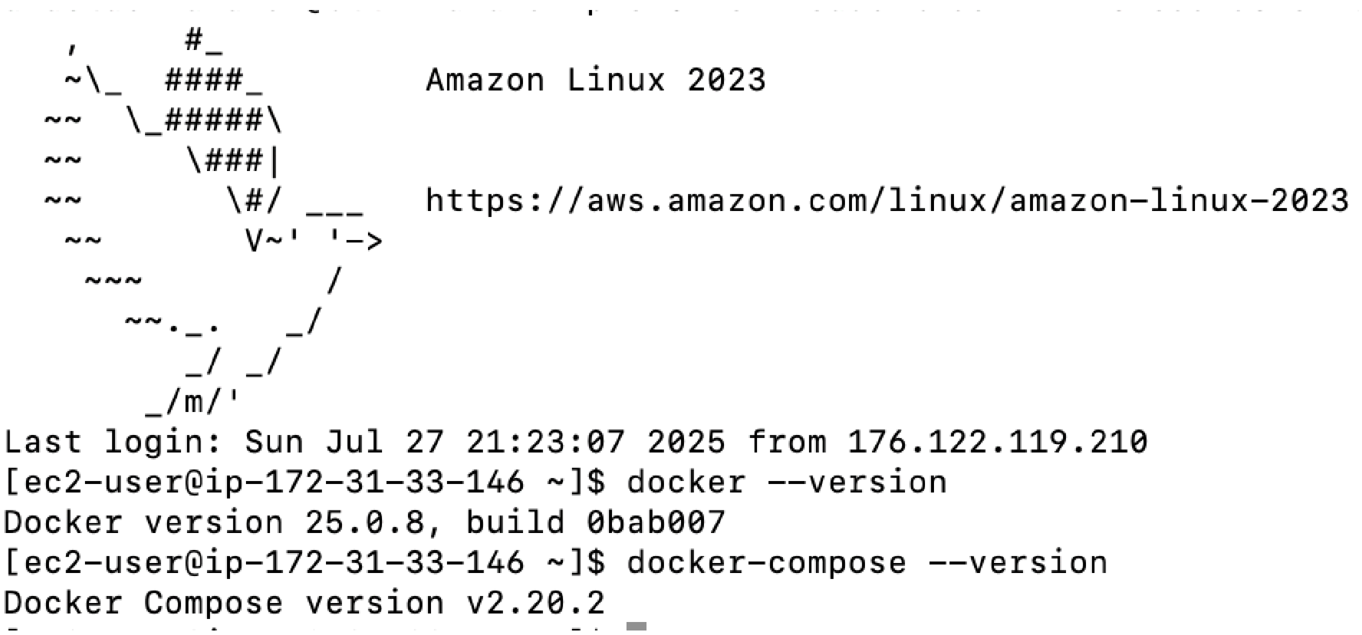
Now we can upgrade the environment and install docker to build main services for next steps:

```
sudo yum update -y

sudo yum install -y docker
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker ec2-user

sudo curl -L
"https://github.com/docker/compose/releases/download/v2.20.2/docker-compose-
linux-x86_64" -o /usr/local/bin/docker-compose sudo chmod +x
/usr/local/bin/docker-compose

sudo yum install -y git vim curl wget htop
```



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Jul 27 21:23:07 2025 from 176.122.119.210
[ec2-user@ip-172-31-33-146 ~]$ docker --version
Docker version 25.0.8, build 0bab007
[ec2-user@ip-172-31-33-146 ~]$ docker-compose --version
Docker Compose version v2.20.2
```

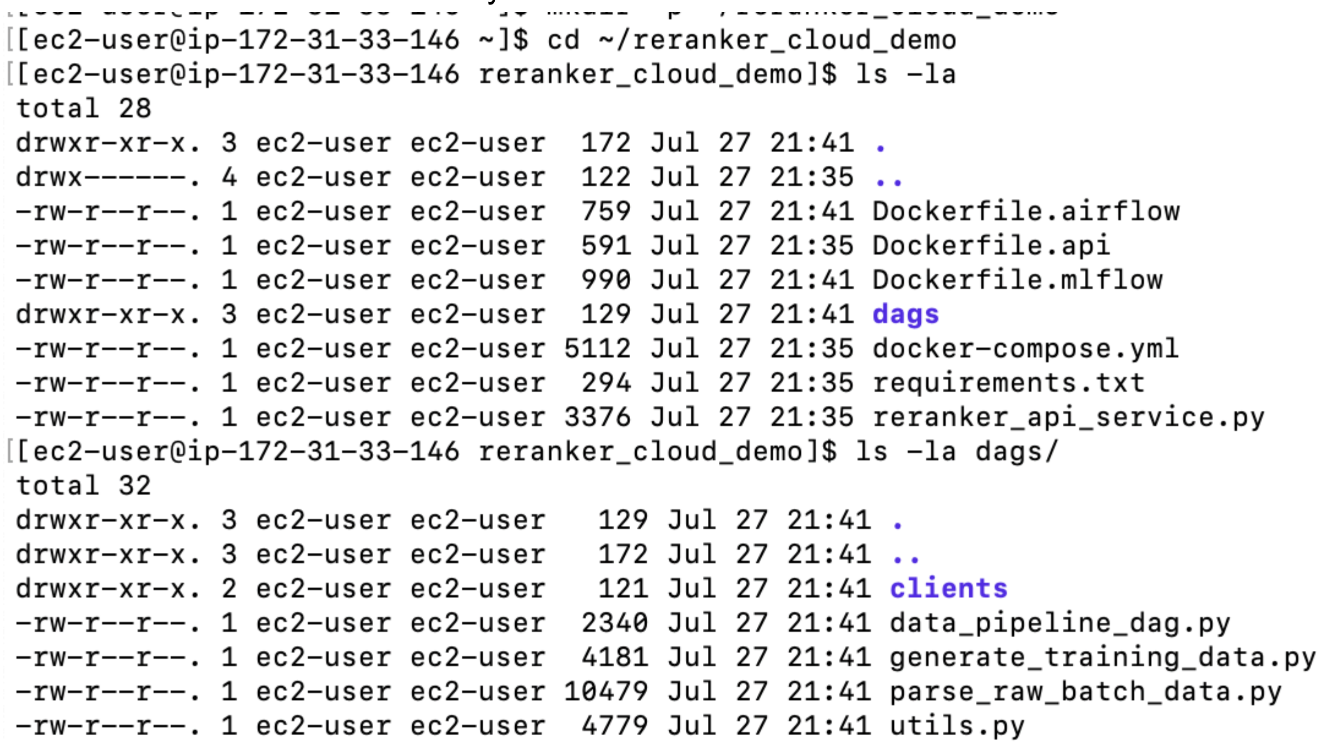
Create project folder on EC2:

```
mkdir -p ~/reranker_cloud_demo
```

Transfer the necessary files to the EC2 instance using SCP:

```
scp -i ~/Downloads/cloud-demo-key.pem docker-compose.yml ec2-
user@16.170.226.195:~/reranker_cloud_demo/
scp -i ~/Downloads/cloud-demo-key.pem Dockerfile.airflow ec2-
user@16.170.226.195:~/reranker_cloud_demo/
scp -i ~/Downloads/cloud-demo-key.pem Dockerfile.api ec2-
user@16.170.226.195:~/reranker_cloud_demo/
scp -i ~/Downloads/cloud-demo-key.pem Dockerfile.mlflow ec2-
user@16.170.226.195:~/reranker_cloud_demo/
scp -i ~/Downloads/cloud-demo-key.pem requirements.txt ec2-
user@16.170.226.195:~/reranker_cloud_demo/
scp -i ~/Downloads/cloud-demo-key.pem reranker_api_service.py ec2-
user@16.170.226.195:~/reranker_cloud_demo/
scp -r -i ~/Downloads/cloud-demo-key.pem dags/ ec2-
user@16.170.226.195:~/reranker_cloud_demo/
```

Check all files are transfer correctly:



```
[[ec2-user@ip-172-31-33-146 ~]$ cd ~/reranker_cloud_demo
[[ec2-user@ip-172-31-33-146 reranker_cloud_demo]$ ls -la
total 28
drwxr-xr-x. 3 ec2-user ec2-user  172 Jul 27 21:41 .
drwx------. 4 ec2-user ec2-user  122 Jul 27 21:35 ..
-rw-r--r--. 1 ec2-user ec2-user  759 Jul 27 21:41 Dockerfile.airflow
-rw-r--r--. 1 ec2-user ec2-user  591 Jul 27 21:35 Dockerfile.api
-rw-r--r--. 1 ec2-user ec2-user  990 Jul 27 21:41 Dockerfile.mlflow
drwxr-xr-x. 3 ec2-user ec2-user  129 Jul 27 21:41 dags
-rw-r--r--. 1 ec2-user ec2-user 5112 Jul 27 21:35 docker-compose.yml
-rw-r--r--. 1 ec2-user ec2-user  294 Jul 27 21:35 requirements.txt
-rw-r--r--. 1 ec2-user ec2-user 3376 Jul 27 21:35 reranker_api_service.py
[[ec2-user@ip-172-31-33-146 reranker_cloud_demo]$ ls -la dags/
total 32
drwxr-xr-x. 3 ec2-user ec2-user   129 Jul 27 21:41 .
drwxr-xr-x. 3 ec2-user ec2-user   172 Jul 27 21:41 ..
drwxr-xr-x. 2 ec2-user ec2-user   121 Jul 27 21:41 clients
-rw-r--r--. 1 ec2-user ec2-user  2340 Jul 27 21:41 data_pipeline_dag.py
-rw-r--r--. 1 ec2-user ec2-user  4181 Jul 27 21:41 generate_training_data.py
-rw-r--r--. 1 ec2-user ec2-user 10479 Jul 27 21:41 parse_raw_batch_data.py
-rw-r--r--. 1 ec2-user ec2-user  4779 Jul 27 21:41 utils.py
```

Finally, magic is going to happen:

```
docker-compose up --build -d
```

SageMaker Pipeline

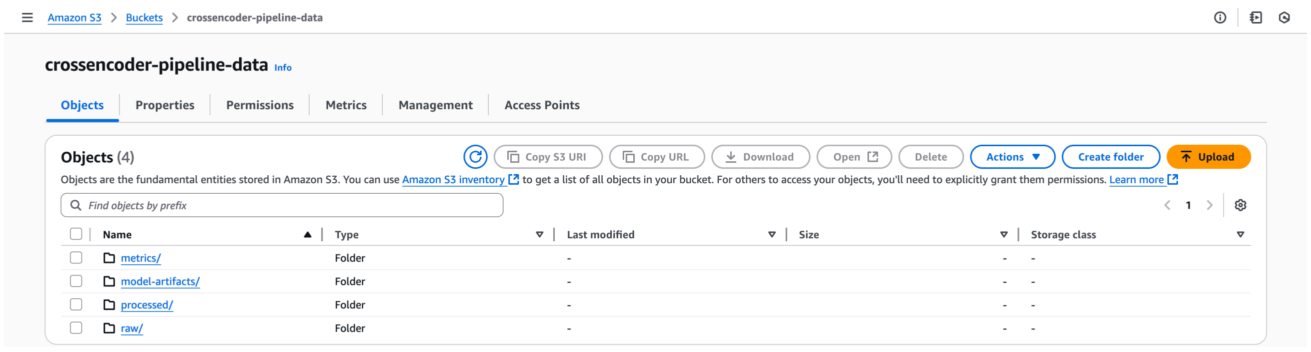
Each stage of the pipeline—data preparation, training, drift tracking, and deployment—is encapsulated in a separate, reusable component, and then defined as a single step in the final pipeline.
Additionally some preparation files are present in main directory to setup the environment for execution.

```
sagemaker_pipeline/
├── scripts/
│   ├── deploy.py
│   ├── evaluate.py
│   ├── load_data.py
│   ├── prepare.py
│   ├── supabase_client_wrapper.py
│   ├── track_shift.py
│   └── train.py
├── build_and_push.sh            # Shell script to build and push the
Docker image to ECR
├── Dockerfile                   # Dockerfile used to create a container
for pipeline steps
```

```
├── pipeline.py                # Main orchestration script defining the
SageMaker pipeline
└── requirements-sagemaker.txt     # Python dependencies needed in the
SageMaker environment
```

Preparations

1. Create S3 bucket with folder structure as a preparing step in advance:



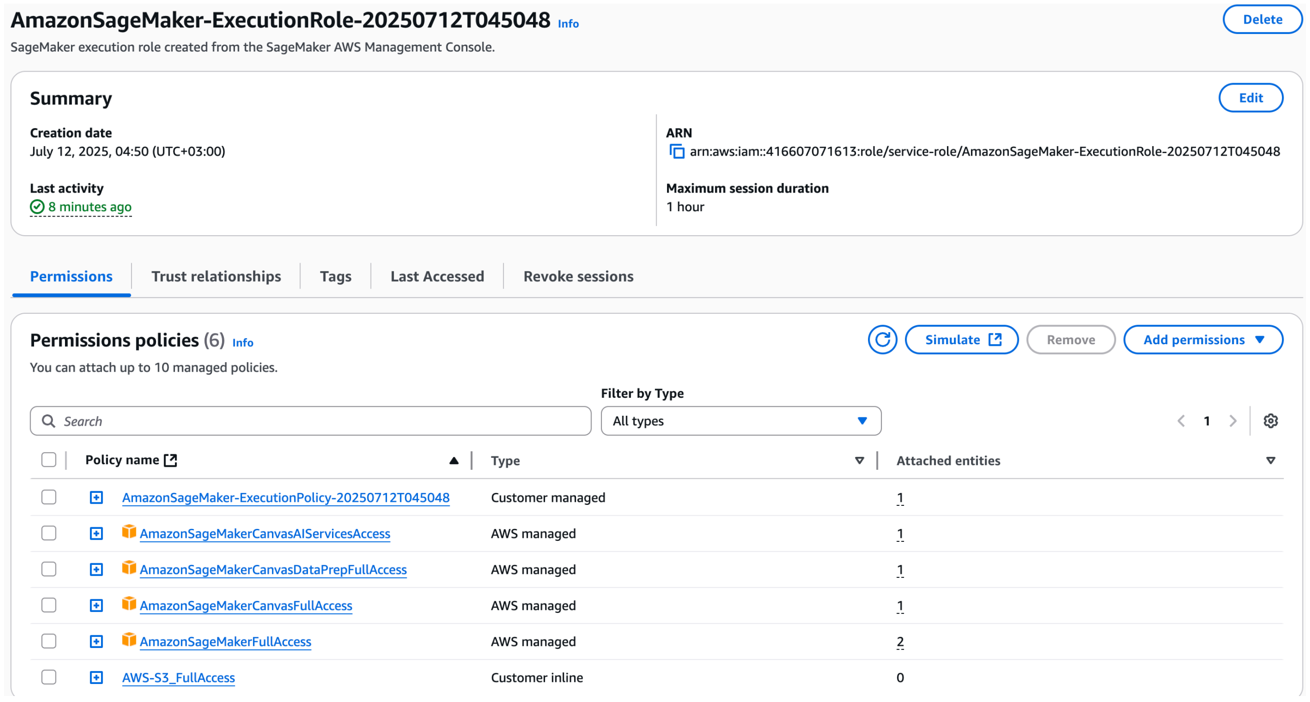
crossencoder-pipeline-data

Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
metrics/	Folder	-	-	-
model-artifacts/	Folder	-	-	-
processed/	Folder	-	-	-
raw/	Folder	-	-	-

2. Create S3 Full Access Permission for the Sagemaker Executor Role to have access to these `crossencoder-pipeline-data` bucket.



AmazonSageMaker-ExecutionRole-20250712T045048

SageMaker execution role created from the SageMaker AWS Management Console.

Summary

Creation date
July 12, 2025, 04:50 (UTC+03:00)

Last activity
8 minutes ago

ARN
arn:aws:iam::416607071613:role/service-role/AmazonSageMaker-ExecutionRole-20250712T045048

Maximum session duration
1 hour

Permissions policies (6)

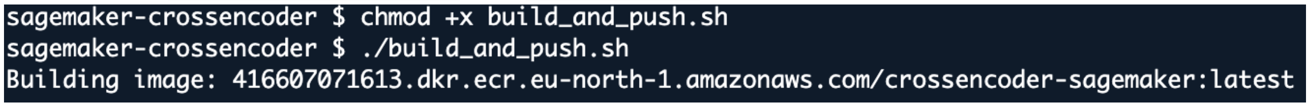
You can attach up to 10 managed policies.

Filter by Type
All types

Policy name	Type	Attached entities
AmazonSageMaker-ExecutionPolicy-20250712T045048	Customer managed	1
AmazonSageMakerCanvasAIServicesAccess	AWS managed	1
AmazonSageMakerCanvasDataPrepFullAccess	AWS managed	1
AmazonSageMakerCanvasFullAccess	AWS managed	1
AmazonSageMakerFullAccess	AWS managed	2
AWS-S3_FullAccess	Customer inline	0

3. Create custom docker image based in it to have a suitable environment with all necessary dependencies for future pipeline execution.
   1. Prepare `requirements-sagemaker.txt` for data shift/drift tracking, training, evaluation and deployment task.
   2. Prepare `Dockerfile` based on `requirements-sagemaker.txt`.

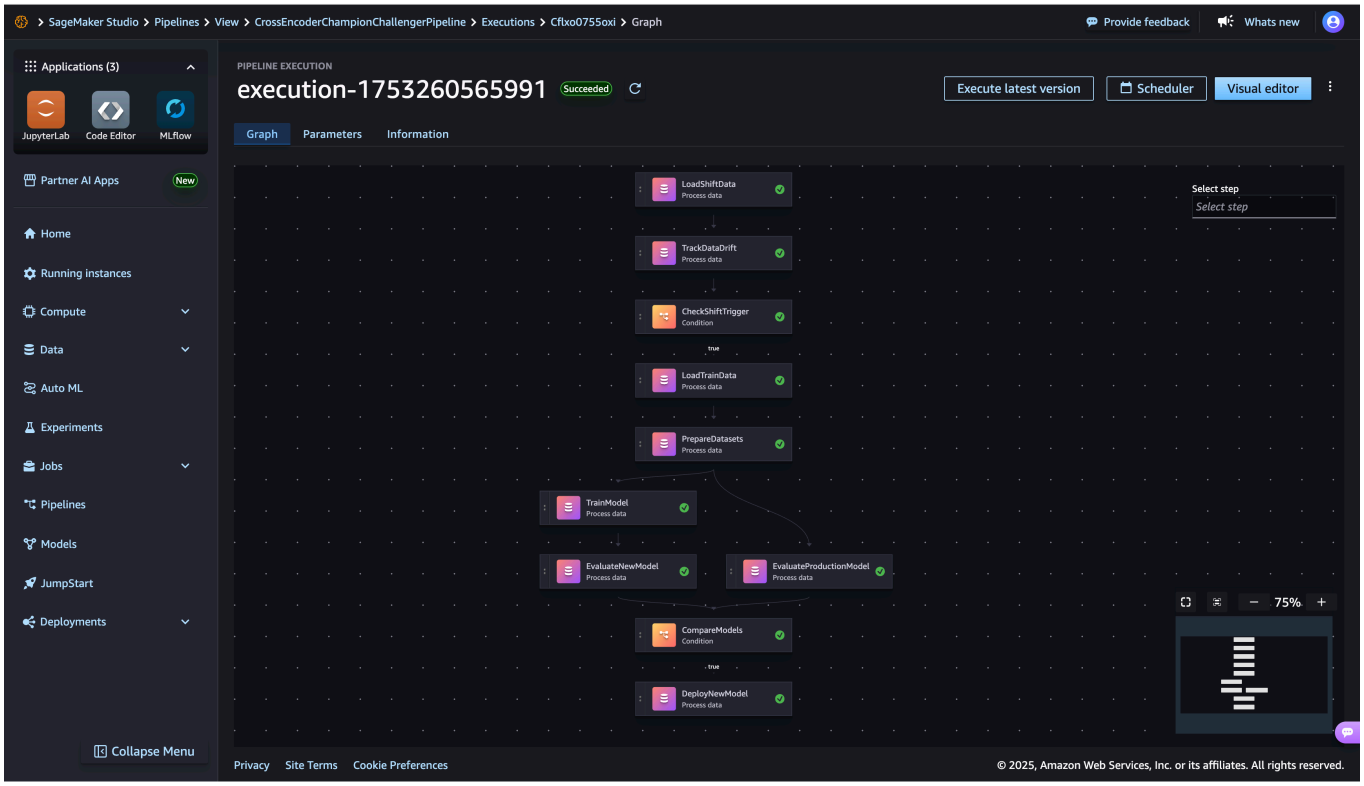
3. Prepare `build_and_push.sh` for `Dockerfile` to become an accessible image.

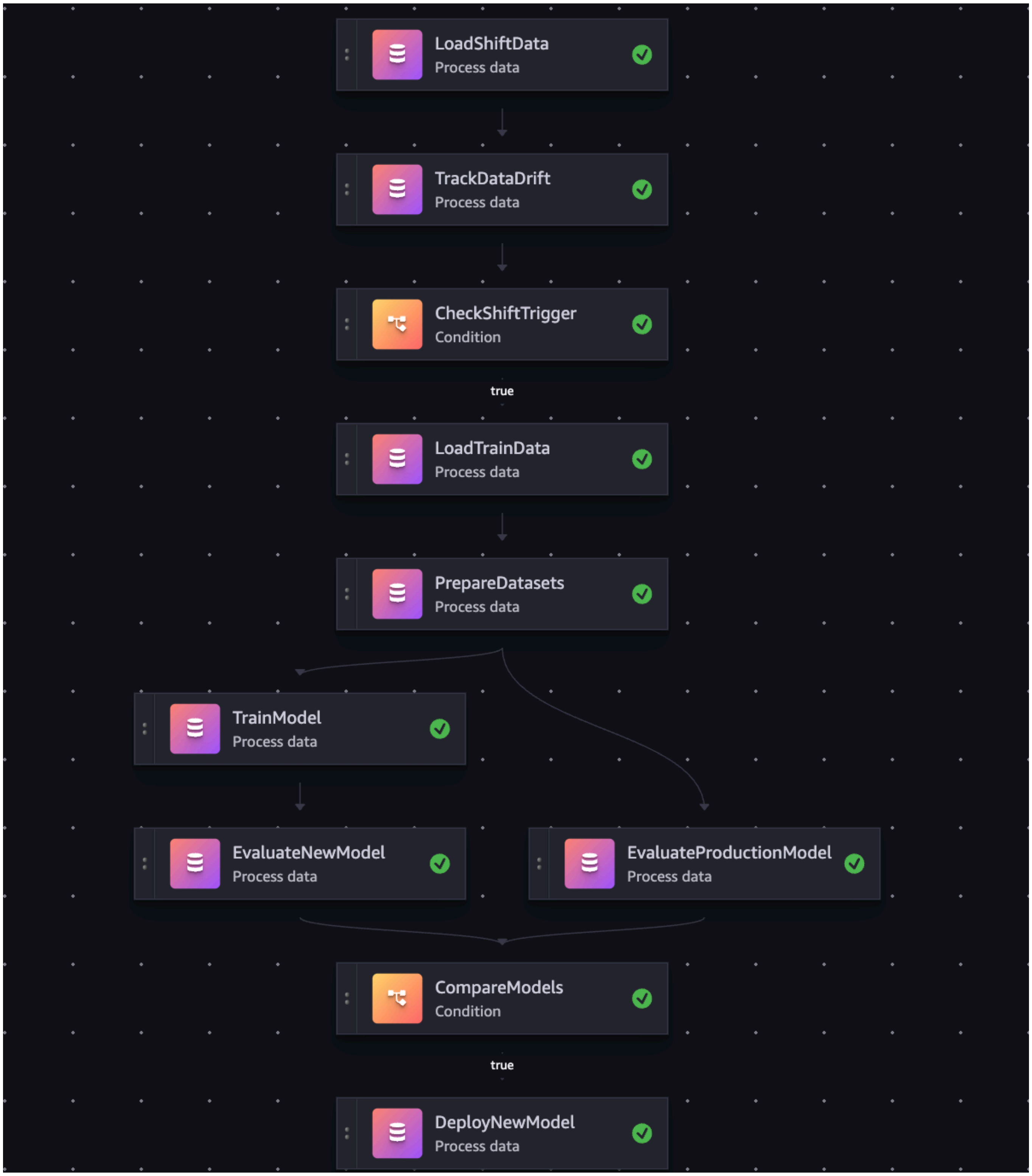


```
sagemaker-crossencoder $ chmod +x build_and_push.sh
sagemaker-crossencoder $ ./build_and_push.sh
Building image: 416607071613.dkr.ecr.eu-north-1.amazonaws.com/crossencoder-sagemaker:latest
```

4. Just copy the `scripts` directory and run `pipeline.py` in the SageMaker Studio.

Final SageMaker Pipeline Architecture


SageMaker Studio > Pipelines > View > CrossEncoderChampionChallengerPipeline > Executions > Cflxo0755oxi > Graph
Provide feedback
Whats new
Applications (3)
JupyterLab
Code Editor
MLflow
Partner AI Apps
New
Home
Running instances
Compute
Data
Auto ML
Experiments
Jobs
Pipelines
Models
JumpStart
Deployments
Collapse Menu
PIPELINE EXECUTION
execution-1753260565991
Succeeded
Execute latest version
Scheduler
Visual editor
Graph
Parameters
Information
LoadShiftData
Process data
TrackDataDrift
Process data
CheckShiftTrigger
Condition
true
LoadTrainData
Process data
PrepareDatasets
Process data
TrainModel
Process data
EvaluateNewModel
Process data
EvaluateProductionModel
Process data
CompareModels
Condition
true
DeployNewModel
Process data
Select step
Select step
75%
Privacy
Site Terms
Cookie Preferences
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.


LoadShiftData
Process data
TrackDataDrift
Process data
CheckShiftTrigger
Condition
true
LoadTrainData
Process data
PrepareDatasets
Process data
TrainModel
Process data
EvaluateNewModel
Process data
EvaluateProductionModel
Process data
CompareModels
Condition
true
DeployNewModel
Process data