MLOPS Pipeline Runbook

Version History

Sr No	Version ID	Date	Author	Remark
1	1.0	21-07-2025	Manoj P	First Version

TABLE OF CONTENT

OVERVIEW 2

PURPOSE 3

OUTCOME 3

TECHNICAL OUTCOMES 3

ARCHITECTURE DIAGRAM 3

PIPELINE COMPONENTS 4

DEPLOYMENT PROCESS 5

CREATE AND ORGANIZE GITHUB REPOSITORY 5

RUNBOOK: CREATE GITHUB CONNECTION USING CLOUDFORMATION 5

GOAL 5

DEPLOYMENT STEPS 5

RUNBOOK: CREATE AWS SERVICE CATALOG PORTFOLIO, AWS SERVICE CATALOG PRODUCT AND SAGEMAKER CUSTOM PROJECT, INFRASTRUCTURE FOR SWITCH BASED RETRAINING OF MODEL AND STOP INITIAL EXECUTION OF CODEPIPELINE USING SINGLE CLOUDFORMATION TEMPLATE 12

GOAL 13

DEPLOYMENT STEPS 13

RUNBOOK: IF YOU WANT TO CREATE AN AWS SERVICE CATALOG PORTFOLIO SEPARATELY FOR SAGEMAKER MLOPS 15

GOAL 15

PREREQUISITES 16

DEPLOYMENT STEPS 16

RUNBOOK: IF YOU WANT TO CREATE SERVICE CATALOG PRODUCT SEPARATELY FOR MULTI-MODEL TRAINING AND BATCH INFERENCE PIPELINE 17

GOAL 17

PREREQUISITES 17

DEPLOYMENT STEPS 18

RUNBOOK: IF YOU WANT TO CREATE SAGEMAKER ML PROJECT SEPARATELY. 19

GOAL 20

PREREQUISITES.....20
TEMPLATE DESCRIPTION.....20
DEPLOYMENT STEPS.....20
RUNBOOK: DEPLOYING LAMBDA + EVENTBRIDGE RULE FOR CROSS ACCOUNT MODEL DEPLOYMENT (PRODUCTION).....21
GOAL.....21
PREREQUISITES.....21
DEPLOYMENT STEPS.....21
CREATE ONE ROLE FOR SAGEMAKER TASKS IN PRODUCTION ACCOUNT.....21
RUNBOOK: PERMISSIONS FOR CROSS ACCOUNT SAGEMAKER MODEL DEPLOYMENT.....21

OVERVIEW

The MLOps Pipeline Asset Runbook provides comprehensive guidance for Creating, implementing, and managing machine learning operations pipelines in AWS environments. This document serves as the primary reference for MLops pipeline implementation.

The pipeline encompasses the complete ML workflow from data ingestion and model training to deployment, monitoring, and continuous improvement. It implements industry best practices for scalability and reliability while ensuring efficient resource utilization and cost optimization.

PURPOSE

The primary purpose of this runbook is to:

- Standardize MLOps Processes: Establish consistent procedures for machine learning model development, deployment, and lifecycle maintenance.
- Enable Automation: Provide guidelines for implementing CI/CD pipelines tailored for ML workloads.
- Ensure Reliability: Define best practices for monitoring model quality, monitoring data quality and retraining model.

OUTCOME

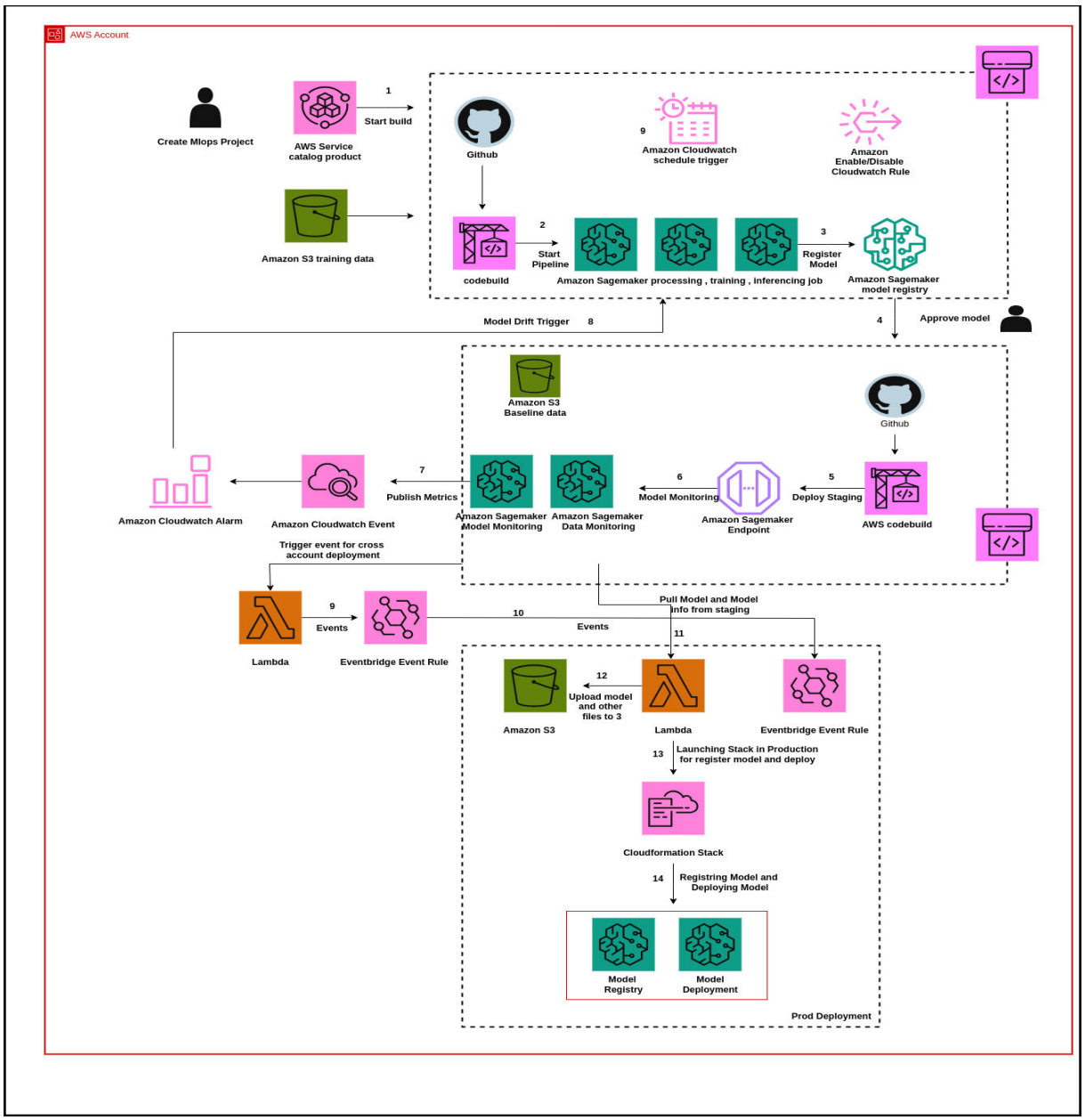
Upon successful implementation of this asset, the user will be able to create the complete infrastructure required for an MLOps pipeline.

TECHNICAL OUTCOMES

- Orchestrated model training using Amazon SageMaker Pipelines.
- Model version control and lifecycle management via SageMaker Model Registry.
- Deployment of models to Staging and Production environments configured for batch inference workloads.
- Automated ML Pipeline: Seamless automation from data ingestion to model deployment.
- Continuous Integration/Deployment: Automated testing, validation, and release of ML models.

- Real-time Monitoring: Integrated monitoring of model performance, including data drift detection.
- Version Control: Comprehensive versioning of both model artifacts and code.

ARCHITECTURE DIAGRAM



PIPELINE COMPONENTS

- Source: GitHub repository
- CodePipeline: Full orchestration
- CodeBuild: Build phase
- SageMaker: Pipelines, Model Registry, Custom Project
- Lambda Functions
- Eventbridge Rules
- Amazon S3
- Amazon ECR
- AWS Service Catalog
- IAM Roles and Policies
- AWS Cloudformation
- SNS Topics
- API Gateways

DEPLOYMENT PROCESS

CREATE AND ORGANIZE GITHUB REPOSITORY

1. Create a new GitHub repository in your GitHub account (if it does not already exist).
2. Once the repository is created, add "axcess-Mlops" folder to it.

RUNBOOK: CREATE GITHUB CONNECTION USING CLOUDFORMATION

GOAL

Set up a GitHub connection in AWS using a CloudFormation template, then complete the GitHub authorization process to make the connection usable in services like CodePipeline using a console.

DEPLOYMENT STEPS

1. **Install AWS CLI**
    a. Installation Guide

2. **Configure AWS CLI (One-Time Setup)/Launch using Cloud shell in AWS environment.**
    a. Launch resources from local.
        i. Before deploying the CloudFormation stack, ensure your AWS CLI is configured with the necessary credentials and region.
        ii. Type bash command given below.
        aws configure

        iii. You will be prompted to enter:
            1. **AWS Access Key ID**
            2. **AWS Secret Access Key**
            3. **Default region name** (e.g., us-west-2)
            4. **Default output format** (you can press Enter to leave it as None or type json)
        iv. Make sure the credentials have sufficient permissions to deploy CloudFormation stacks and manage CodeStar Connections.
        v. Run CLI command given below.

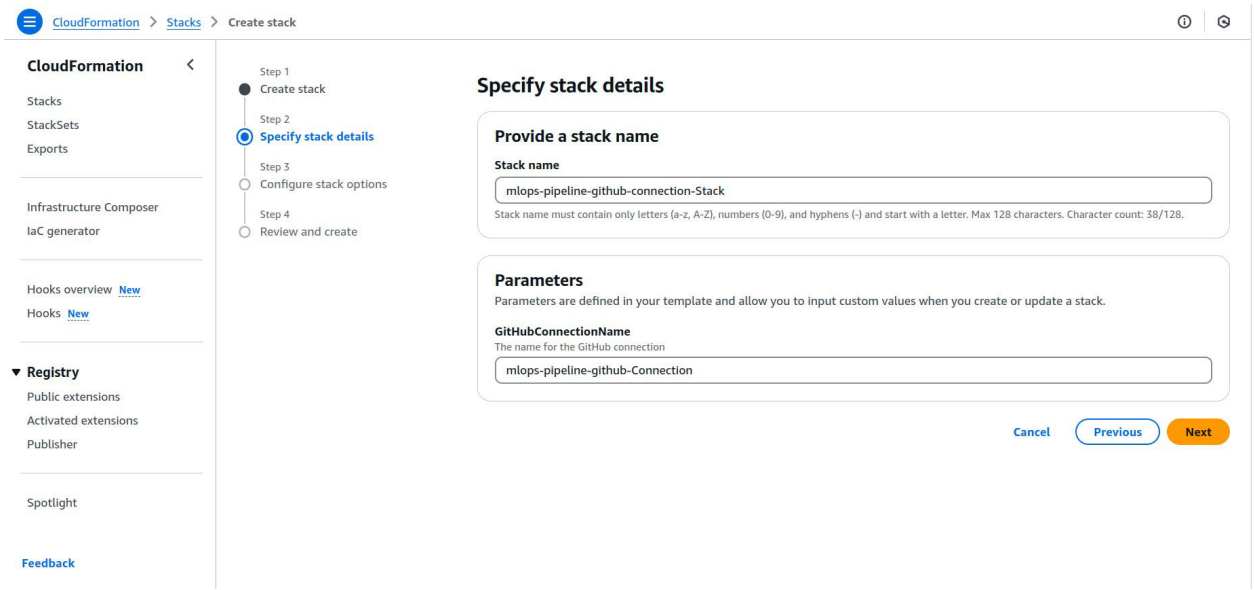
```
aws cloudformation create-stack\
  --template-url https://mlops-pipeline-template-files.s3.us-east-1.amazonaws.com/create-github-connection.yaml \
  --stack-name github-connection-stack \
  --parameters \

ParameterKey=GitHubConnectionName,ParameterValue=mlops-pipeline-github-connection  \
  --capabilities CAPABILITY_NAMED_IAM
```

            i. In template-url: add url of create-github-connection.yaml template which present in S3 bucket.
    b. Launch resources from Cloudshell in AWS account, then directly run the command given below in the Cloudshell.

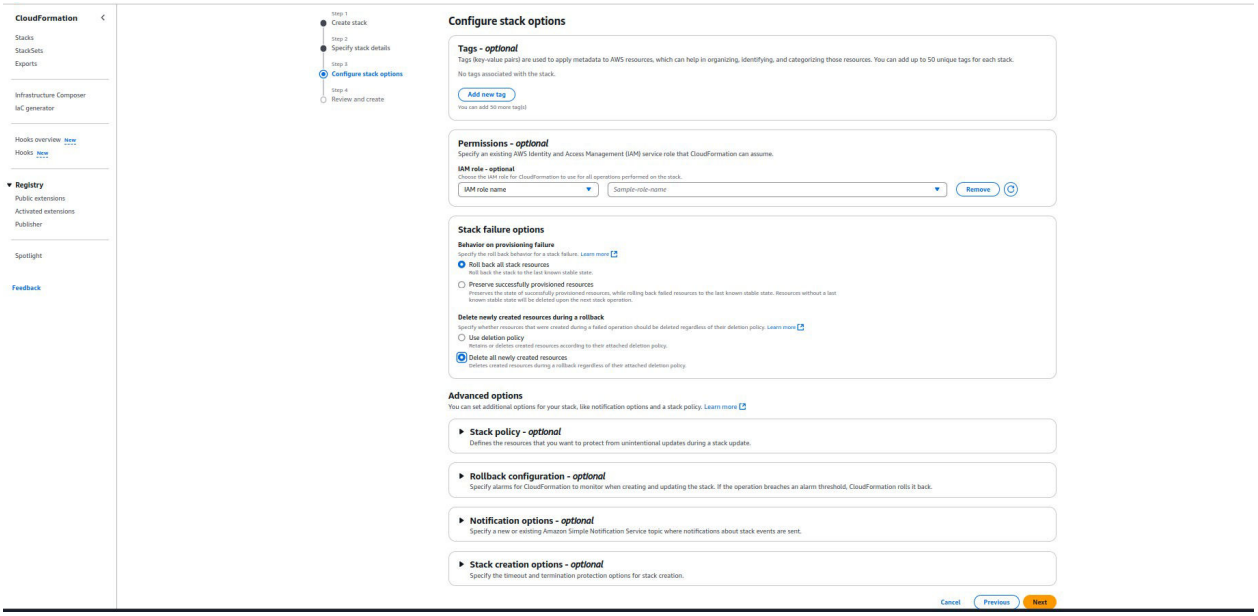
```
aws cloudformation create-stack\
  --template-url https://mlops-pipeline-template-files.s3.us-east-1.amazonaws.com/create-github-connection.yaml \
  --stack-name github-connection-stack \
  --parameters \
     ParameterKey=GitHubConnectionName,ParameterValue=mlops-pipeline-github-connection  \
  --capabilities CAPABILITY_NAMED_IAM
```

i. In template-url: add url of create-github-connection.yaml template which present in S3 bucket.

4. **Launch resources using AWS Console (**Note: Step 4-6: Launching resources using AWS Console)
    a. Open AWS CloudFormation Console.
    b. Upload the template you created (create-github-connection.yaml).
    c. When prompted, provide a value for stack name and GitHubConnectionName (e.g., mlops-pipeline-github-connection) and click **Next**.

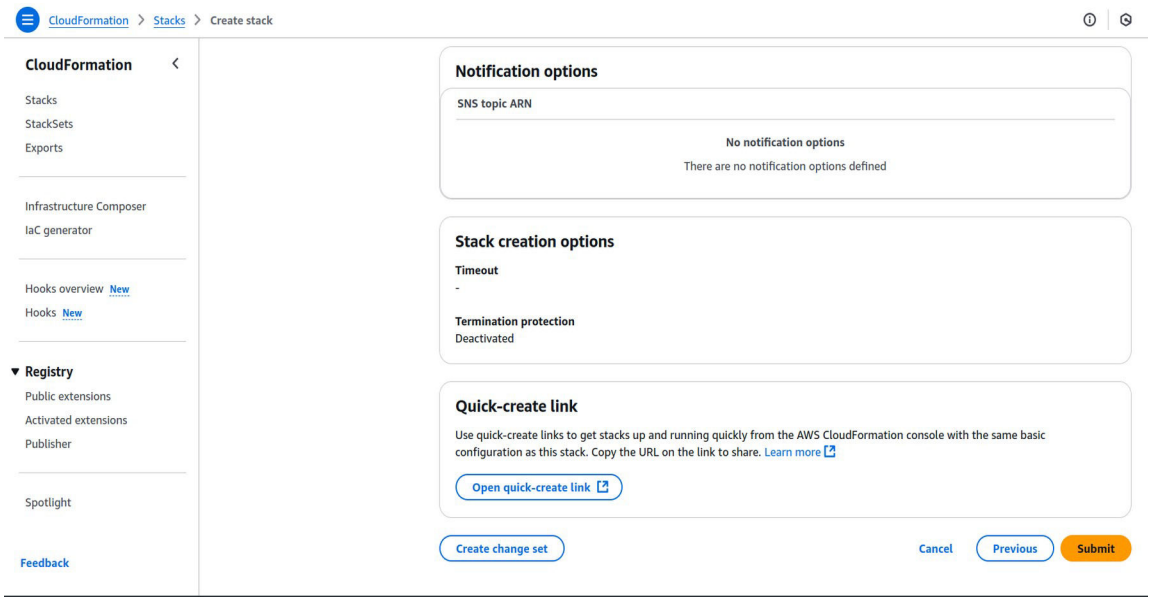


5. **Configure Stack Options**



a. Leave the default options or add tags if required.
b. Click **Next**.

6. **Review and Submit**

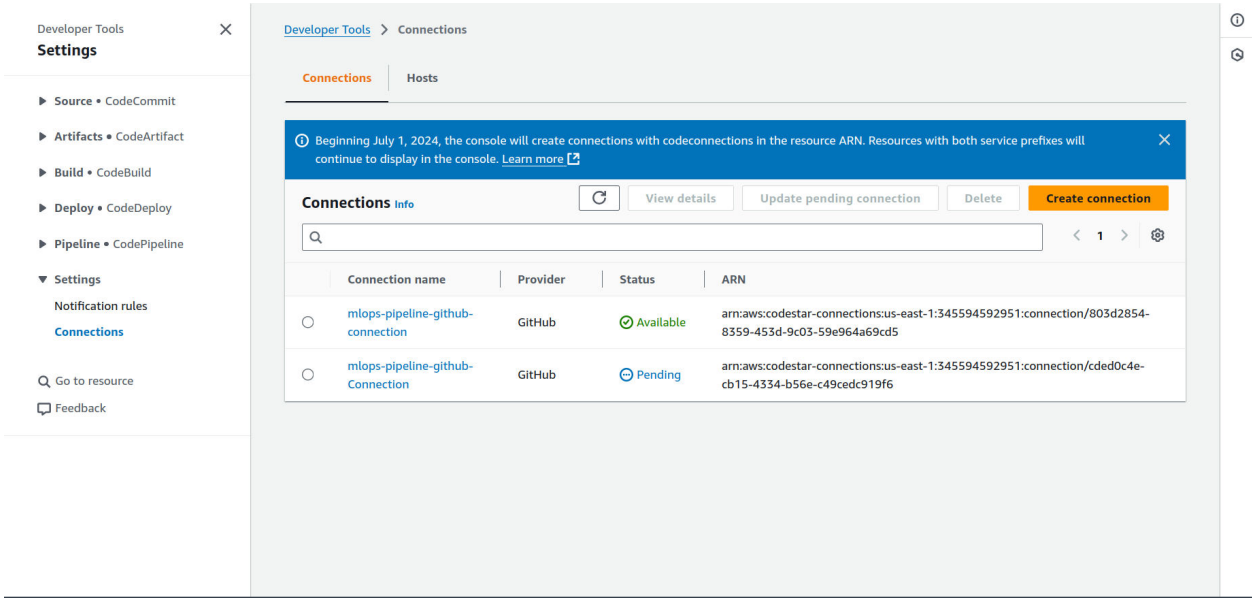


a. Review the stack configuration.
b. Click **Submit** to launch the stack.

7. **Navigate to CodePipeline Settings**
    a. After the stack is created, go to the AWS Console.
    b. Open the **CodePipeline** service.
    c. Go to the **Settings** tab on the left.
8. **Access Connection Settings**

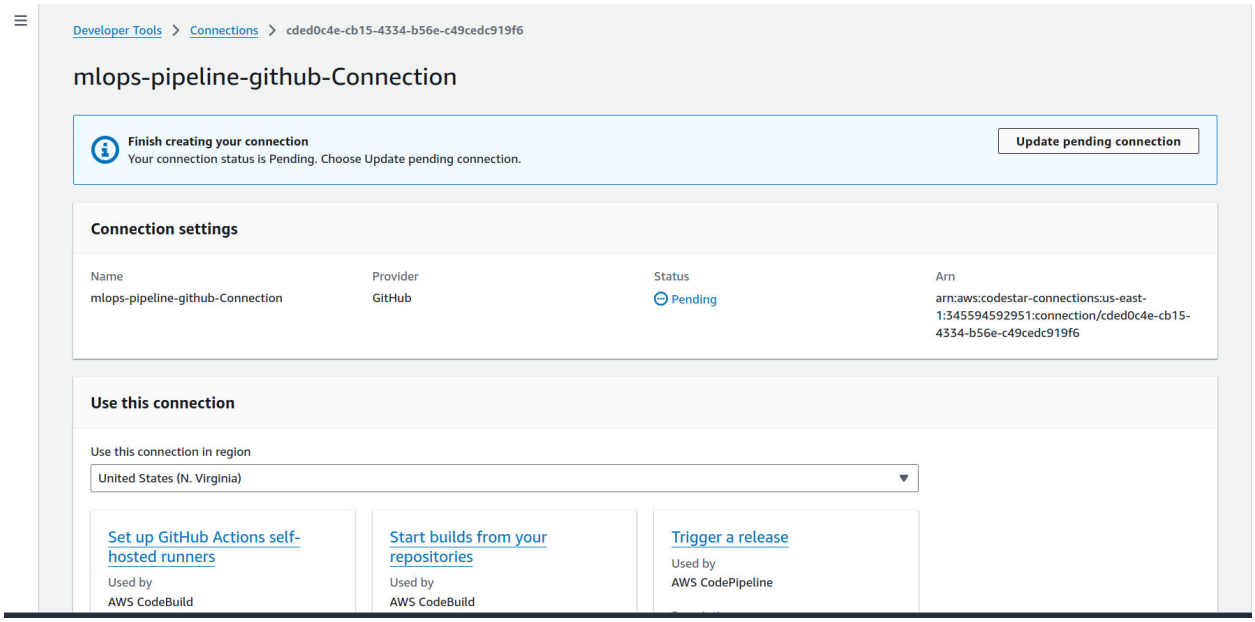
a. In the settings panel, click on the **Connections** section.

9. Check Connection State



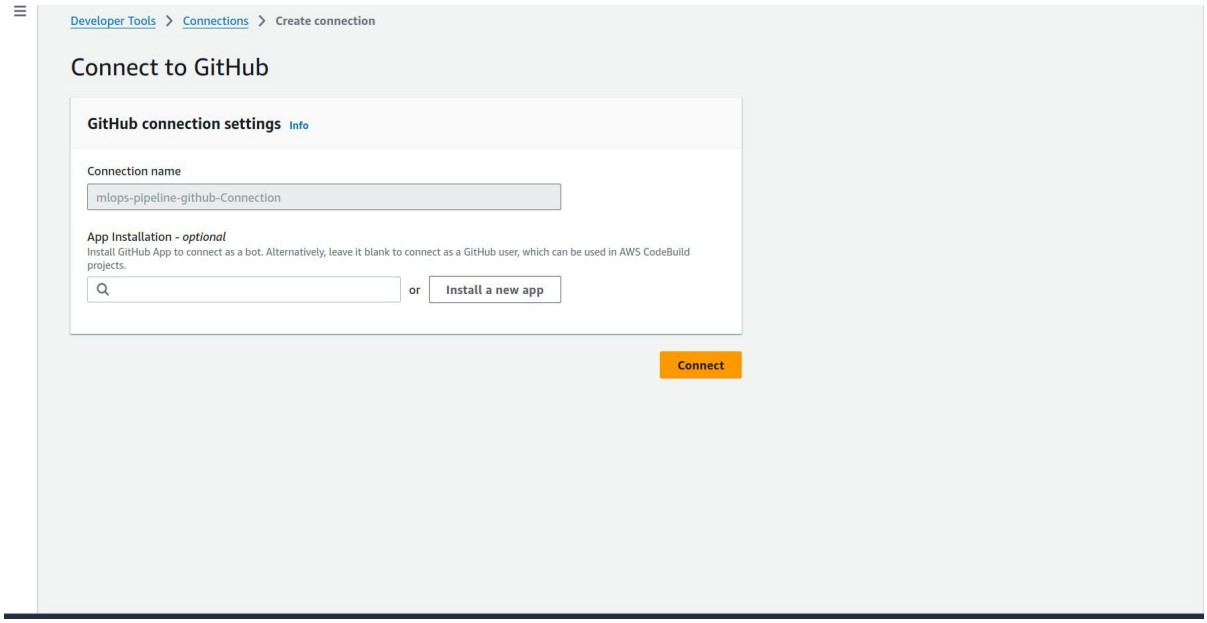
a. You will see the newly created connection listed.
b. Its status will be **Pending**.

10. Update Pending Connection



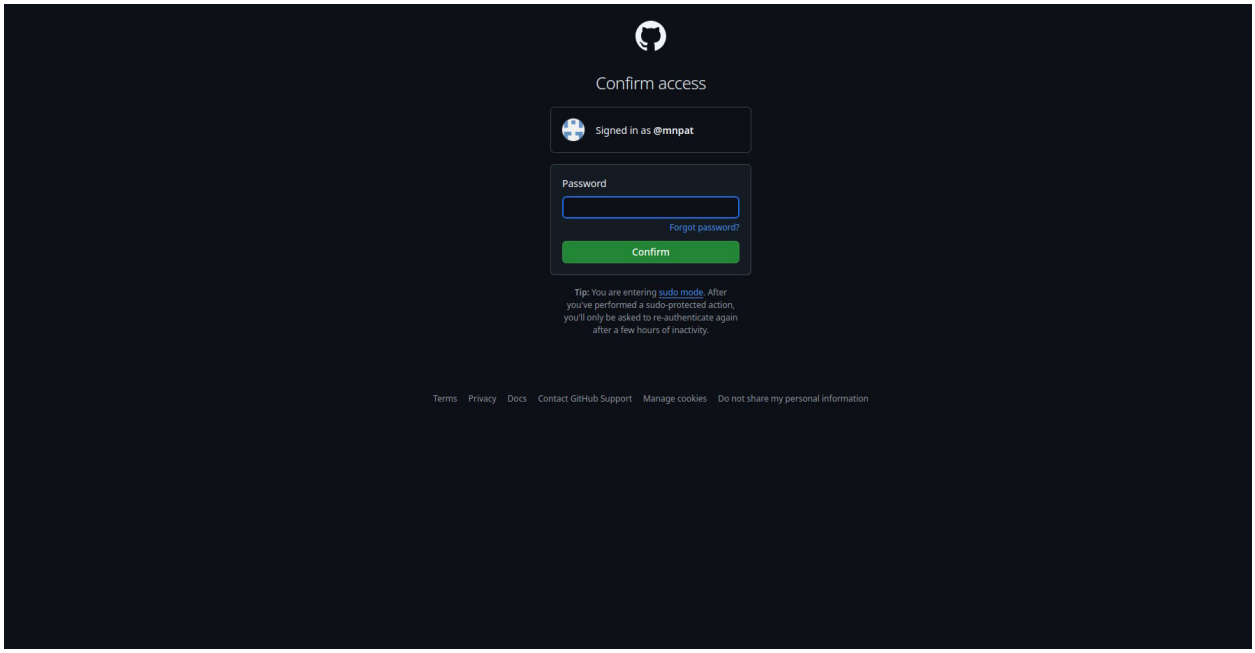
a. Click on the connection name.
b. Then click on the **Update pending connection** button.

11. Install GitHub App



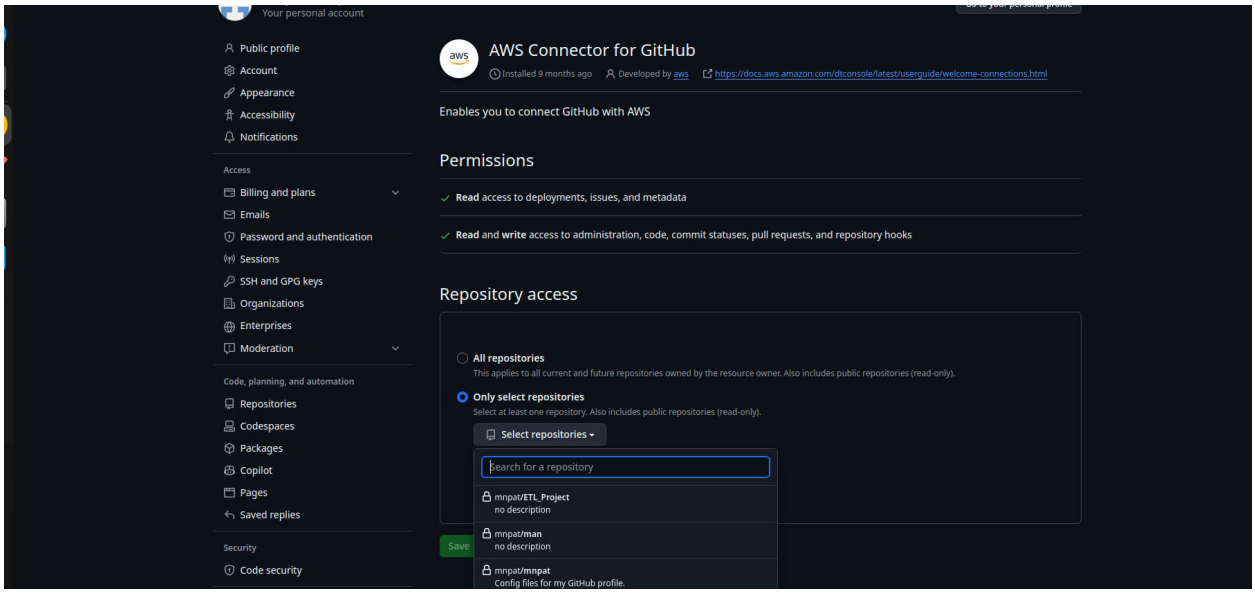
a. In the popup, select **Install a new app**.

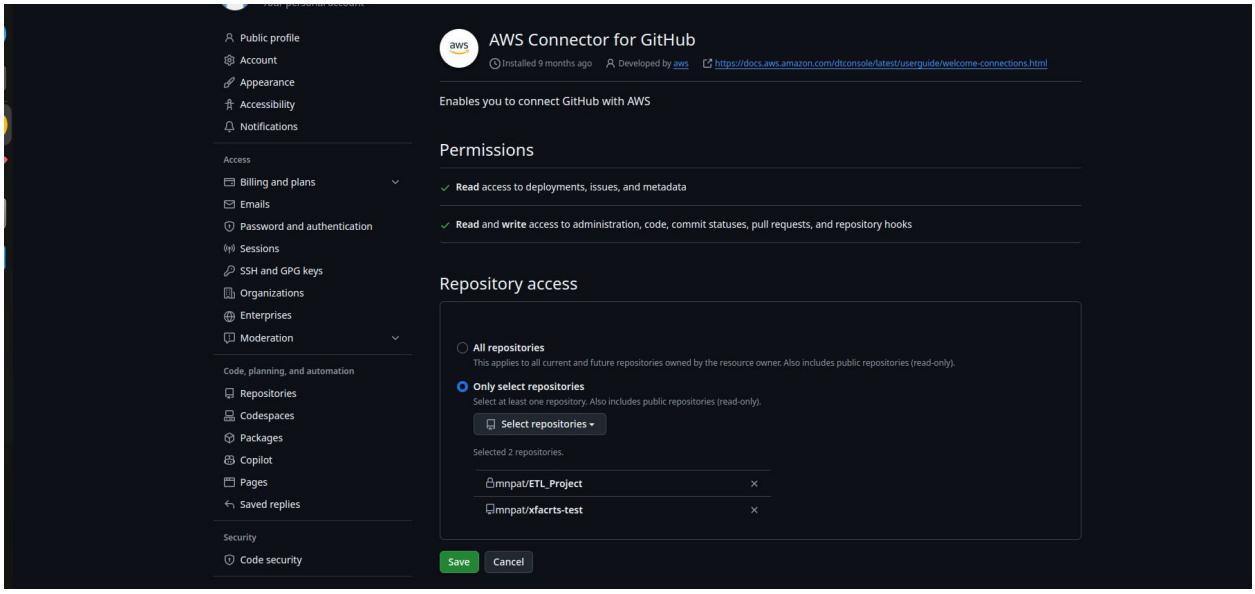
12. Authenticate with GitHub



a. A GitHub login window will open.
b. Enter your **GitHub username and password**.

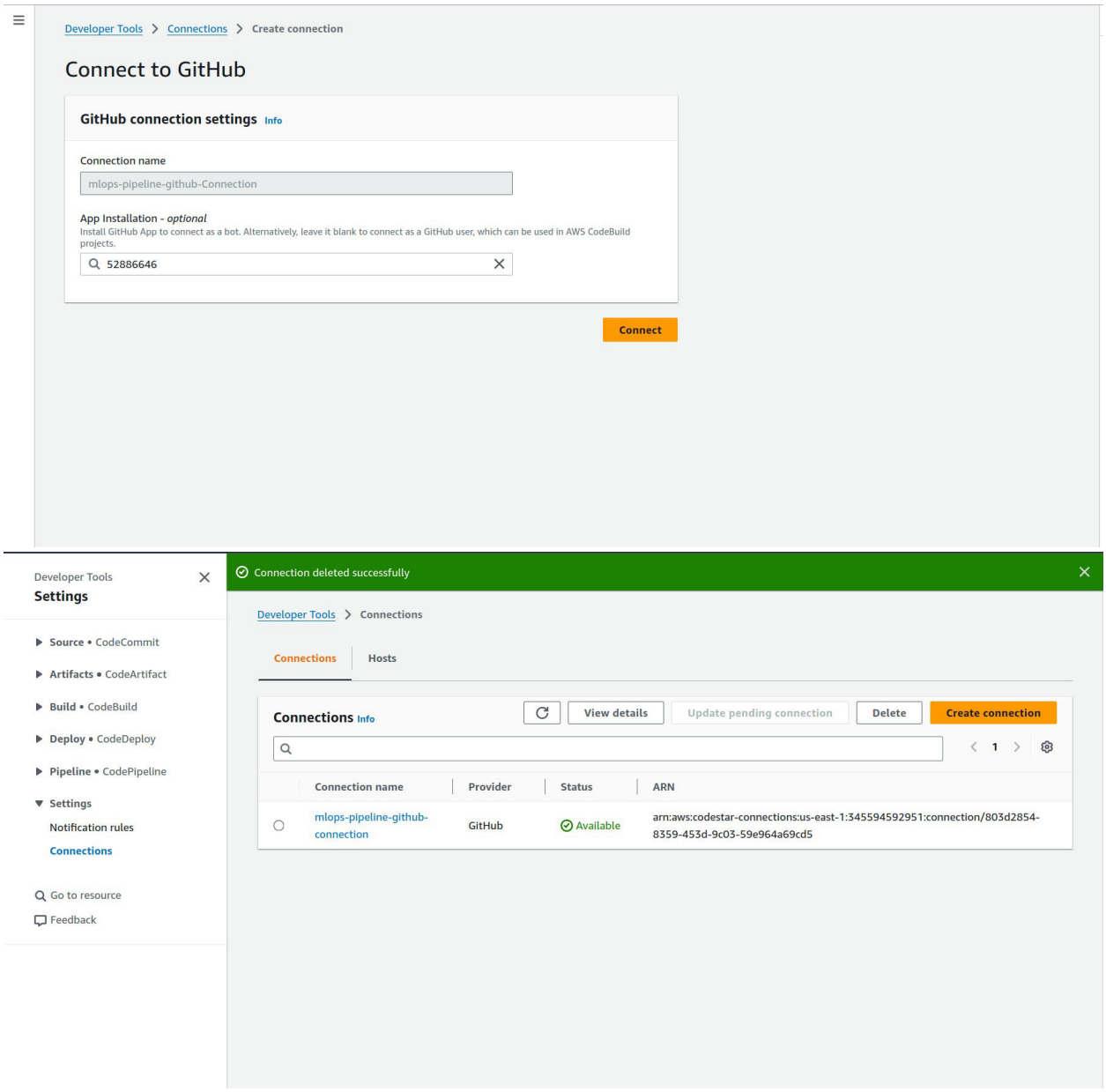
13. Authorize Repository Access





a. Choose the **GitHub repository** you want to allow access to.
b. Click **Save**.

14. Connect and Finalize



a. Back in the AWS Console, click **Connect**.
b. The connection will now move to **Available** state.

RUNBOOK: CREATE AWS SERVICE CATALOG PORTFOLIO, AWS SERVICE CATALOG PRODUCT AND SAGEMAKER CUSTOM PROJECT, INFRASTRUCTURE FOR SWITCH BASED RETRAINING OF MODEL AND STOP INITIAL EXECUTION OF CODEPIPELINE USING SINGLE CLOUDFORMATION TEMPLATE.

GOAL

This runbook provisions a Service Catalog Portfolio, Product, SageMaker custom project, and infrastructure for switch-based model retraining, while preventing initial CodePipeline execution — all via a single CloudFormation template.

DEPLOYMENT STEPS

1. **Install AWS CLI**
    a. Installation Guide
    b. Skip this step if you have already installed the AWS CLI.
2. **Configure AWS CLI (One-Time Setup)/Launch using Cloudshell in AWS environment.**
    a. Steps for Configure AWS CLI already given in above step.
    b. Launch resources from local/from cloudshell in an AWS account.

```
aws cloudformation create-stack \
  --template-url https://axcess-devst-sagemaker.s3.us-east-1.amazonaws.com/mlops-pipeline-resource-templates/Complete_Mlops_Package.yaml \
  --stack-name complete-mlops-resources-stack \
  --parameters \
    ParameterKey=PortfolioName,ParameterValue=mlops-service-catalog-portfolio \

ParameterKey=SageMakerExecutionRoleArn,ParameterValue=arn:aws:iam::345951:role/service-role/AmazonSageMaker-ExecutionRole-20250325T120134 \

ParameterKey=PrincipalArn,ParameterValue=arn:aws:iam::345951:role/aws-reserved/sso.amazonaws.com/AWSReservedSSO_AWSAdministratorAccess_834ce3a515a6f00d \
    ParameterKey=ProductName,ParameterValue=mlops-service-catalog-product \

ParameterKey=GithubUrl,ParameterValue="https://github.com/
```

```
<user-name>/<repo-name>" \

ParameterKey=ZipUrlGithubRepo,ParameterValue="https://github.com/<user-name>/<repo-name>/archive/refs/heads/<brach-name>.zip" \
    ParameterKey=RepoBranchName,ParameterValue="<repo-name>-<branch-name>" \
    ParameterKey=MainFolderName,ParameterValue="<main-folder-name-from-repo>" \


ParameterKey=ServiceCatalogPortfolioTemplateURL,ParameterValue=https://axcess-devst-sagemaker.s3.us-east-1.amazonaws.com/mlops-pipeline-resource-templates/create-service-catalog-portfolio.yaml \

ParameterKey=ServiceCatalogProductTemplateURL,ParameterValue=https://axcess-devst-sagemaker.s3.us-east-1.amazonaws.com/mlops-pipeline-resource-templates/create-product.yaml \

ParameterKey=StopPipelineTemplateURL,ParameterValue=https://axcess-devst-sagemaker.s3.us-east-1.amazonaws.com/mlops-pipeline-resource-templates/Stop-specific-codepipeline-executions.yaml \

ParameterKey=SageMakerProjectName,ParameterValue="axcess-sagemaker-project-8" \
    ParameterKey=GitHubOwner,ParameterValue="<github-user-name>" \
    ParameterKey=GitHubRepo,ParameterValue="<github-repo-name>" \
    ParameterKey=GitHubBranch,ParameterValue="<branch-name>" \

ParameterKey=GithubConnArn,ParameterValue="arn:aws:codeconnections:us-east-1:345951:connection/0cdb21f1-6062-433b-bdea-f4bde8ea6f07" \
    ParameterKey=UseCase,ParameterValue="trip-prediction" \
    ParameterKey=S3Prefix,ParameterValue="NA" \

ParameterKey=ProcessingInstanceType,ParameterValue="ml.t3.large" \

ParameterKey=TrainingInstanceType,ParameterValue="ml.m5.large" \
```

```
            ParameterKey=ProcessingInstanceCount,ParameterValue="1" \

            ParameterKey=TrainingInstanceCount,ParameterValue="1" \

            ParameterKey=ModelPackageGroupName,ParameterValue="trip-prediction" \
                ParameterKey=ModelVersion,ParameterValue="1" \
        ParameterKey=
Approveremail,ParameterValue="approver@example.com" \
        ParameterKey=
Executoremail,ParameterValue="executor@example.com" \

        ParameterKey=DeployStackName,ParameterValue="trip-prediction-1-deploy-staging" \
        ParameterKey=S3Bucket,ParameterValue="axcess-devst-sagemaker-bucket-1" \
    --capabilities CAPABILITY_NAMED_IAM \
    --region us-east-1
```

i. In template-url: add url of Complete_Mlops_Package.yaml template which present in S3 bucket. (using this template will launch all resources)
ii. In SageMakerExecutionRoleArn: add arn of sagemaker domain execution role.
iii. In PrincipalArn: add arn of user or sso user role.
iv. In ZipUrlGithubRepo: add url of zipped repo.
v. In RepoBranchName: add "RepoName"-"BrachName"
vi. In ServiceCatalogPortfolioTemplateURL: add url of create-service-catalog-portfolio.yaml template present in S3 bucket.
vii. In ServiceCatalogProductTemplateURL: add url of create-product.yaml template which present in S3 bucket.
viii. In GithubConnArn: add arn of github connection which created first step.
ix. In StopPipelineTemplateURL: add url of Stop-specific-codepipeline-executions.yaml template which present in S3 bucket.
x. In Approveremail: add email id of Approver/Administrator
xi. In Executoremail: add email id of Pipeline Executor/Engineer

3. Manual Configuration Steps After Stack Launch
    1. Update processing.py with the S3 Bucket Name
        - Add or update the S3 bucket name in the processing.py file to ensure it points to the correct data location.
    2. Build and Push Updated Docker Image for Processing

- Make the required changes to your local environment.
- Build a new Docker image for the processing step.
- Push the updated image to your container registry with a new version tag.

3. Update pipeline.py with the New Image Version
   - Update the pipeline.py file in your GitHub repository to reference the new version of the processing image.
   - Commit and push the changes to your GitHub repository.
4. Upload Required Dataset to S3 for Pipeline Execution
   - Upload the data.csv file to the following S3 location so it can be used in the processing step of the pipeline
   - s3://<bucket-name>/taxi-duration/original_raw_data/
5. S3 Configuration for Pipeline Execution
   - Add the S3 bucket name and the training pipeline name in the model_manifest.json file.
   - Upload the following configuration files to their respective S3 locations:
     - model_manifest.json to "s3://<bucket-name>/taxi-duration/data/config/"
     - monitoring_config.json to "s3://<bucket-name>/taxi-duration/data/config/"
     - Upload the batch inference input file(inference_data.csv) to "s3://<bucket-name>/taxi-duration/batch_input/"
     - Upload the model monitoring dataset(monitoring_dataset.csv) to "s3://<bucket-name>/taxi-duration/model_monitor/input/"
6. Update IAM Role ARNs in Scripts for Stack Integration
   - In data_quality_monitor.py, update the IAM role ARNs within the initialize_resources function as per your environment configuration.
   - In retraining.py, update the role using the PrincipalArn provided in the CloudFormation stack parameters during stack launch, ensuring the correct IAM role is assumed during pipeline execution.
7. Parameter Store Setup for MLOps Pipeline
   - Update the switch_config and monitoring_config parameters in AWS Systems Manager Parameter Store based on your specific requirements.

RUNBOOK: IF YOU WANT TO CREATE AN AWS SERVICE CATALOG PORTFOLIO SEPARATELY FOR SAGEMAKER MLOPS

GOAL

This runbook helps you deploy a Service Catalog Portfolio using CloudFormation to support Amazon SageMaker MLOps workflows.

PREREQUISITES

- AWS CLI installed and configured
- Required IAM permissions to create Service Catalog resources

DEPLOYMENT STEPS

Step 1: Deployment Using AWS Console

1. **Open AWS CloudFormation Console**
    a. CloudFormation Console
2. **Create Stack**
    a. Click **"Create stack" → "With new resources (standard)"**
3. **Specify Template**
    a. Choose your template file (.yaml or .json)
    b. If the template is in S3, select **Amazon S3 URL** and paste the full path (e.g., https://s3.amazonaws.com/your-bucket/template.yaml)
4. **Provide Parameters**
    a. Fill out values as per the parameter.
5. **Stack Options**
    a. Leave default or add tags if needed
6. **Permissions**
    a. Ensure the deploying IAM user/role has access to servicecatalog:* and cloudformation:*
7. **Review and Submit**
    a. Click **Create stack**

Step 2: Deploy CloudFormation Stack

- Run below command to create Service Catalog Portfolio

```
aws cloudformation create-stack \
--template-url
https://mlops-pipeline-template-files.s3.us-east-1.amazonaws.com/create-service-catalog-portfolio.yaml \
```

```
--stack-name mlops-service-catalog-portfolio-stack \
--parameters \
   ParameterKey=AllowDirectUserAccessParameter,ParameterValue=false \

ParameterKey=MakePorfolioAccessibleFromSageMakerStudioParameter,ParameterValue=true \
   ParameterKey=PortfolioDescriptionParameter,ParameterValue="Custom project templates for MLOps" \

ParameterKey=PortfolioNameParameter,ParameterValue=mlops_pipeline_service_catalog_portfolio \
  ParameterKey=PortfolioOwnerParameter,ParameterValue=owner \
ParameterKey=SageMakerStudioExecutionRoleParameter,ParameterValue=arn:aws:iam::317046:role/service-role/AmazonSageMaker-ExecutionRole-20240228T142935 \
   ParameterKey=UserAccessPrincipalsParameter,ParameterValue= \
--capabilities CAPABILITY_NAMED_IAM \
--region us-east-1
```

i. In SageMakerStudioExecutionRoleParameter section add role of Sagemaker Domain.
ii. Keep UserAccessPrincipalsParameter section blank.
iii. In PortfolioNameParameter section, add the name for Service Catalog Portfolio.

Step 3: Post Deployment

After successful deployment:

1. Navigate to **AWS Console > Service Catalog > Portfolios**
2. You will find a new portfolio named mlops_pipeline_service_catalog_portfolio
3. Confirm:
    a. The execution role is associated
    b. In tag section sagemaker:studio-visibility should be true

RUNBOOK: IF YOU WANT TO CREATE SERVICE CATALOG PRODUCT SEPARATELY FOR MULTI-MODEL TRAINING AND BATCH INFERENCE PIPELINE

GOAL

This CloudFormation template deploys a **Service Catalog product** that can be used from within **Amazon SageMaker Studio** to launch a custom project for MLOps pipelines.

PREREQUISITES

- An existing **Service Catalog Portfolio**
- Portfolio ID (from CloudFormation stack output if you used a template to create it)
- AWS CLI configured (or use the AWS Console)

DEPLOYMENT STEPS

Step 1: Deployment Using AWS Console

1. **Open AWS CloudFormation Console**
    a. CloudFormation Console
2. **Create Stack**
    a. Click **"Create stack" → "With new resources (standard)"**
3. **Specify Template**
    a. Choose your template file (.yaml or .json)
    b. f the template is in S3, select **Amazon S3 URL** and paste the full path (e.g., https://s3.amazonaws.com/your-bucket/template.yaml)
4. **Provide Parameters**
    a. Fill out values as per the parameter.
5. **Stack Options**
    a. Leave default or add tags if needed
6. **Permissions**
    a. Ensure the deploying IAM user/role has access to servicecatalog:* and cloudformation:*
7. **Review and Submit**
    a. Click **Create stack**

Step 2: AWS CLI Deployment

- Run below command to create Service Catalog Portfolio

```
aws cloudformation create-stack \
--template-url
https://mlops-pipeline-template-files.s3.us-east-1.amazonaws.com/create-
product.yaml \
--stack-name mlops-service-catalog-product-stack \
--parameters \
   ParameterKey=PortfolioIDParameter,ParameterValue=port-474rj4x4tzpw2 \
   ParameterKey=ProductNameParameter,ParameterValue=mlops-pipeline-service-
catalog-product \
   ParameterKey=ProductDescriptionParameter,ParameterValue="A pipeline project
for training and batch inference" \
```

```
   ParameterKey=ProductOwnerParameter,ParameterValue="Product Owner" \
   ParameterKey=ProductDistributorParameter,ParameterValue="Product
Distributor" \
   ParameterKey=ProductSupportDescriptionParameter,ParameterValue="Support
Description" \

ParameterKey=ProductSupportEmailParameter,ParameterValue="support@example.c
om" \
ParameterKey=ProductSupportURLParameter,ParameterValue="https://github.com/
<user-name>/<repo-name>" \
ParameterKey=SageMakerProjectRepoZipParameter,ParameterValue="https://
github.com/mnpat/axcess-test/archive/refs/heads/main.zip" \

ParameterKey=SageMakerProjectRepoNameBranchParameter,ParameterValue="<rep
o-name>-<branch-name>" \
 ParameterKey=SageMakerProjectsProjectNameParameter,ParameterValue="<main-
folder-name-from-repo>" \
--capabilities CAPABILITY_NAMED_IAM \
--region us-east-1
```

i. In PortfolioIDParameter add Service Catalog Portfolio ID in which we have to create Service Catalog Product.
ii. In ProductNameParameter add name for Product.
iii. In ProductSupportURLParameter add url of Github Repo where custom project template is present.
iv. In SageMakerProjectRepoZipParameter url should like ProductSupportURLParameter/archive/refs/heads/main.zip
v. In SageMakerProjectRepoNameBranchParameter add Name/Branch of the SageMaker Projects Examples GitHub Repo. (should like RepoName-Branch)
vi. In SageMakerProjectsProjectNameParameter add Project folder inside of the GitHub repo for this project.

Validation Steps

1. Go to **Service Catalog > Products**
2. We should see a new product with the name provided.
3. Open **Amazon SageMaker Studio**
4. Go to **Projects > Create project**
5. We should see the custom project listed in the Organization Template section.

RUNBOOK: IF YOU WANT TO CREATE SAGEMAKER ML PROJECT SEPARATELY.

GOAL

This CloudFormation template provisions all required infrastructure for managing a CI/CD pipeline in **Amazon SageMaker**, targeting batch inference jobs. It uses GitHub as the source, SageMaker Pipelines for training, and Model Registry for model versioning and deployment.

PREREQUISITES

Ensure the following are in place before deployment:

- **GitHub connection** is created via *AWS CodeStar Connections*.
- AWS CLI configured:
- IAM role has permissions to deploy CloudFormation stacks and access SageMaker, S3, and CodePipeline.

TEMPLATE DESCRIPTION

This template:

- Sets up CodePipeline for CI/CD
- Uses SageMaker Pipelines for training
- Registers models in Model Registry
- Automates deployment to **staging** environments for **batch inference**
- Setup infrastructure for switch-based retraining

DEPLOYMENT STEPS

Step 1: Deploy Step (AWS CLI)

1. **Run AWS CLI Command**

```
aws sagemaker create-project --project-name "sm-project" --service-
catalog-provisioning-details '{
    "ProductId": "prod-rcp2tgmmj32ck",
    "ProvisioningParameters": [

      {
        "Key": "GitHubOwner",
        "Value": "<github-user-name>"
      },
```

```
            {
              "Key": "GitHubRepo",
              "Value": "<github-remo-name>"
            },
            {
              "Key": "GitHubBranch",
              "Value": "<github-branch-name>"
            },
            {
              "Key": "GithubConnArn",
              "Value": "<github-connection-arn-created_in_first_step>"
            },
            {
              "Key": "UseCase",
              "Value": "<usecase-name-
which_is_folder_in_repo_includes_all_codes_like_processing_training_and
_other>"
            },
            {
              "Key": "s3Prefix",
              "Value": "NA"
            },
            {
              "Key": "ProcessingInstanceType",
              "Value": "<Processing-Instance-Type>"
            },
            {
              "Key": "TrainingInstanceType",
              "Value": "<Training-Instance-Type>"
            },
            {
              "Key": "ProcessingInstanceCount",
              "Value": "<Processing-Instance-Count>"
            },
            {
              "Key": "TrainingInstanceCount",
              "Value": "<Training-Instance-Count>"
            },
            {
              "Key": "ModelPackageGroupnameInput",
              "Value": "<Model-Package-Groupname-Input>"
            },
            {
              "Key": "ModelVersionInput",
              "Value": "<Model-Version-Input>"
            },
```

```
                {
                   "Key": "StackName",
                   "Value": "<stack-name-while_deploying_model_at_staging>"
                },
                {
                   "Key": "mls3bucket",
                   "Value": "<name-of-s3-bucket-using_for_mlops_pipeline>"
                },
                {
                    "Key": "ApproverEmail",
                    "Value": "<email-id-of-Approver>"
                 },
                 {
                   "Key": "ExecutorEmail",
                   "Value": "<email-id-of-Executor>"
            }
        ]
    }'
```

i. In the project name add name of sagemaker project which want to create.
ii. In ProductId add id of product which created in above steps.
iii. In ProvisioningParameters add required values for given keys.

Validation Steps

After stack creation:

- Go to **AWS CodePipeline** → Validate the pipeline stages
- Confirm **Source** connects to GitHub
- Go to **AWS Sagemaker > Domain > User Profile > Sagemaker Studio > Deployments > Project** where custom projects have been launched.

RUNBOOK: DEPLOYING LAMBDA + EVENTBRIDGE RULE FOR CROSS ACCOUNT MODEL DEPLOYMENT (PRODUCTION).

GOAL

This CloudFormation template provisions:

- A **Lambda function** is designed to handle specific production tasks.

- An **Amazon EventBridge rule** that triggers the Lambda function based on predefined events.

This is designed for deployment in the **Production account** as part of the overall MLOps toolchain.

PREREQUISITES

Ensure the following before deployment:

- You are operating in the **Production AWS account**.
- You have **deployment permissions** for AWS Lambda, IAM, and EventBridge.
- AWS CLI or CloudFormation console access is available.
- IAM execution role for Lambda has necessary permissions.

DEPLOYMENT STEPS

Step 1: Using AWS CLI

1. **Execute the CloudFormation deployment**:

```
aws cloudformation deploy \
  --template-url
https://axcess-devst-sagemaker.s3.us-east-1.amazonaws.com/mlops-pipeline-resource-templates/Production-account-setup.yaml \
  --stack-name mlops-pipeline-prod-account-setup \
--parameters \
  ParameterKey= SageMakerProjectName,ParameterValue= <sagemaker-proje>
--capabilities CAPABILITY_NAMED_IAM
```

Validation Checklist

After deployment:

- Go to **AWS Lambda Console** and confirm the Lambda function has been created.
- Validate the **IAM Role** associated with the Lambda (includes logs, event access, etc.).
- Navigate to **Amazon EventBridge > Rules**, and verify the rule exists and is targeting the Lambda.
- Check the **event pattern** to confirm it matches your production trigger (e.g., SageMaker job status).

CREATE ONE ROLE FOR SAGEMAKER TASKS IN PRODUCTION ACCOUNT

- Role name: AmazonSageMakerServiceCatalogProductsUseRoleMultiModelTB
- Policies:
    - AWS managed:
        - AmazonS3FullAccess
        - AmazonSageMakerFullAccess
    - Customer managed:
        - access_ecr_stage_policy {

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ecr:GetAuthorizationToken"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ecr:BatchGetImage",
      "ecr:GetDownloadUrlForLayer",
      "ecr:BatchCheckLayerAvailability"
    ],
    "Resource":
"arn:aws:ecr:us-east-1:345594592951:repository/sagemaker-inference"
  }
]
}
```

        - Below is trust relationship: {

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "",
    "Effect": "Allow",
    "Principal": {
      "Service": "sagemaker.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

RUNBOOK: PERMISSIONS FOR CROSS ACCOUNT SAGEMAKER MODEL DEPLOYMENT.

- User Guide
- In staging and prod account add select create role for eventbridge event while cross account setup.