<http://bit.ly/awskr201>



Amazon SageMaker  
MLOps Workshop

Hands-on Guidebook

8th May 2019

김필호 AI Specialist SA / 서지혜 Partner SA

Table of Contents

[Lab 개요 3](#_Toc8208423)

[목표 3](#_Toc8208424)

[준비 조건 3](#_Toc8208425)

[LAB 실습 가이드 5](#_Toc8208426)

[Architecture 5](#_Toc8208427)

[핸드온 실습 8](#_Toc8208428)

[Module 1: CloudFormation을 이용하여 필요한 모든 서비스 생성하기 8](#_Toc8208429)

[서비스 종료 가이드 9](#_Toc8208430)

# Lab 개요

Data Scientists and ML developers need more than a Jupyter notebook to create a ML model, test it, put into production and integrate it with a portal and/or a basic web/mobile application in a reliable and flexible way.

# 목표

* SageMaker에 내장된 학습 기능을 사용하여 모델 훈련 Job을 생성 합니다.
* SageMaker의 endpoint 기능을 사용하여 생성된 모델이 예측에 사용될 수 있도록 endpoint를 생성합니다.
* 머신 러닝이 정형 데이터(e.g. CSV 파일)와 비정형 데이터(e.g. 이미지)에 모두 적용 될수 있음을 확인 합니다.

# 준비 조건

* AWS 계정: AWS IAM, S3, SageMaker 자원을 생성할 수 있는 권한이 필요합니다.
* AWS Region: SageMaker는 지원되는 region은 <https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/> 에서 확인하실 수 있습니다. 이번 실습은Seoul (ap-northeast-2) region에서 실행 합니다.
* Browser: 최신 버전의 Chrome, Firefox를 사용하세요.

You should have some basic experience with:

* Train/test a ML model
* Python (scikit-learn)
* Jupyter Notebook
* AWS CodePipeline
* AWS CodeCommit
* AWS CodeBuild
* Amazon ECR
* Amazon SageMaker
* AWS CloudFormation

Some experience working with the AWS console is helpful as well.

**※ 주의 사항:** Notebook 안의 Cell에서 코드 실행후 결과 값이 나오는 데는 수 초가 걸립니다. 훈련 Job을 실행하는 경우 수 분이 걸릴 수도 있습니다. 실습 완료 후에는 아래 가이드에 따라 생성된 자원을 꼭 종료/삭제해 주세요.

# LAB 실습 가이드

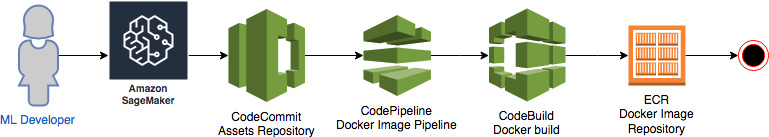
In this workshop you'll implement and experiment a basic MLOps process, supported by an automated infrastructure for training/testing/deploying/integrating ML Models. It is comprised into four parts:

1. You'll start with a WarmUp, for reviewing the basic features of Amazon Sagemaker;
2. Then you will create a basic Docker Image for supporting any scikit-learn model;
3. Then you will create a dispatcher Docker Image that supports two different algorithms;
4. Finally you will train the models, deploy them into a DEV environment, approve and deploy them into a PRD environment with High Availability and Elasticity;

Parts 2, 3 and 4 are supported by automated pipelines that reads the assets produced by the ML devoloper and execute/control the whole process.

# Architecture

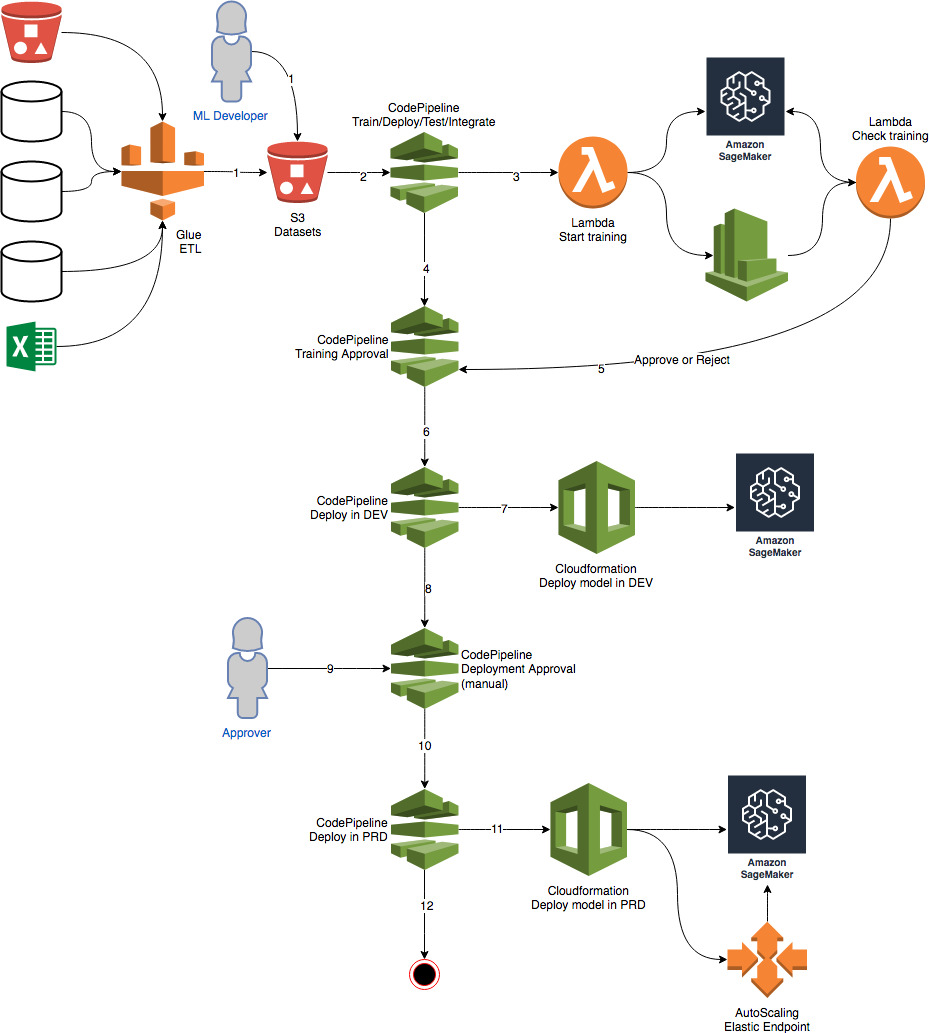
For parts 2 and 3 the following architecture will support the process. In part 2 you'll create an Abstract ScikitLearn Docker Image. In part 3 you'll extend that Abscract image and create the final image using two distinct Scikit Learn algorithms.



1. The ML Developer creates the assets for Docker Image based on Scikit Learn, using Sagemaker, and pushes all the assets to a Git Repo (CodeCommit);
2. CodePipeline listens the push event of CodeCommit, gets the source code and launches CodeBuild;
3. CodeBuild authenticates into ECR, build the Docker Image and pushes it into the ECR repository
4. Done.

For part 4 you'll make use of the following structure for training the model, testing it, deploying it in two different environments: DEV - QA/Development (simple endpoint) and PRD - Production (HA/Elastic endpoint).

Altough there is an ETL part in the Architecture, we'll not use Glue or other ETL tool in this workshop. The idea is just to show you how simple it is to integrate this Architecture with your Data Lake and/or Legacy databases using an ETL process



1. An ETL process or the ML Developer, prepares a new dataset for training the model and copies it into an S3 Bucket;
2. CodePipeline listens to this S3 Bucket, calls a Lambda function for start training a job in Sagemaker;
3. The lambda function send a training job to Sagemaker, enables a rule in CloudWatchEvents that will check each minute, through another Lambda Function, if the training job has finished or failed
4. CodePipeline will awaits for the training approval with success or failure;
5. This lambda will approve or reject this pipeline, based on the Sagemaker results;
6. If rejected the pipeline stops here; If approved it goes to the next stage;
7. CodePipeline calls CloudFormation to deploy a model in a Development/QA environment into Sagemaker;
8. After finishing the deployment in DEV/QA, CodePipeline awaits for a manual approval
9. An approver approves or rejects the deployment. If rejected the pipeline stops here; If approved it goes to the next stage;
10. CodePipeline calls CloudFormation to deploy a model into production. This time, the endpoint will count with an AutoScaling policy for HA and Elasticity.
11. Done.

# 핸드온 실습

## Module 1: CloudFormation을 이용하여 필요한 모든 서비스 생성하기

**1. Select the below to launch CloudFormation stack.**

| **Region** | **Launch** |
| --- | --- |
| US East (N. Virginia) | [Launch MLOps solution in us-east-1](https://console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/new?stackName=AIWorkshop&templateURL=https://s3.amazonaws.com/aws-ai-ml-aod-latam/mlops-workshop/m.yml) |

**2. Then open the Jupyter Notebook instance in Sagemaker and start doing the exercises:**

## Module 2: Warmup

This is a basic exercise for exploring the Sagemaker features like: training, deploying and optmizing a model. If you already have experience with Sagemaker, you can skip this exercise.

## Module 3: Abstract Scikit-learn Image

Here we'll create an abstract docker image with the codebase for virtually any Scikit-learn algorithm. This is not the final Image. We need to complete this exercise before executing the next one.

## Module 4: Concrete Scikit-learn models

In this exercise we'll inherits the Docker image from the previous step and create a concrete Docker image with two different algorithms: Logistic Regression and Random Forest Tree in a dispatcher architecture.

## Module 4: Test the models locally

This is part of Module #3. You can use this jupyter to test your local WebService, to simulate how Sagemaker will call it when you ask it to create an Endpoint or launch a Batch job for you.

## Module 5: Train your models

In this exercise you'll use the training pipeline. You'll see how to train or retrain a particular model by just copying a zip file with the required assets to a given S3 bucket.

## Module 6: Check Training progress and test

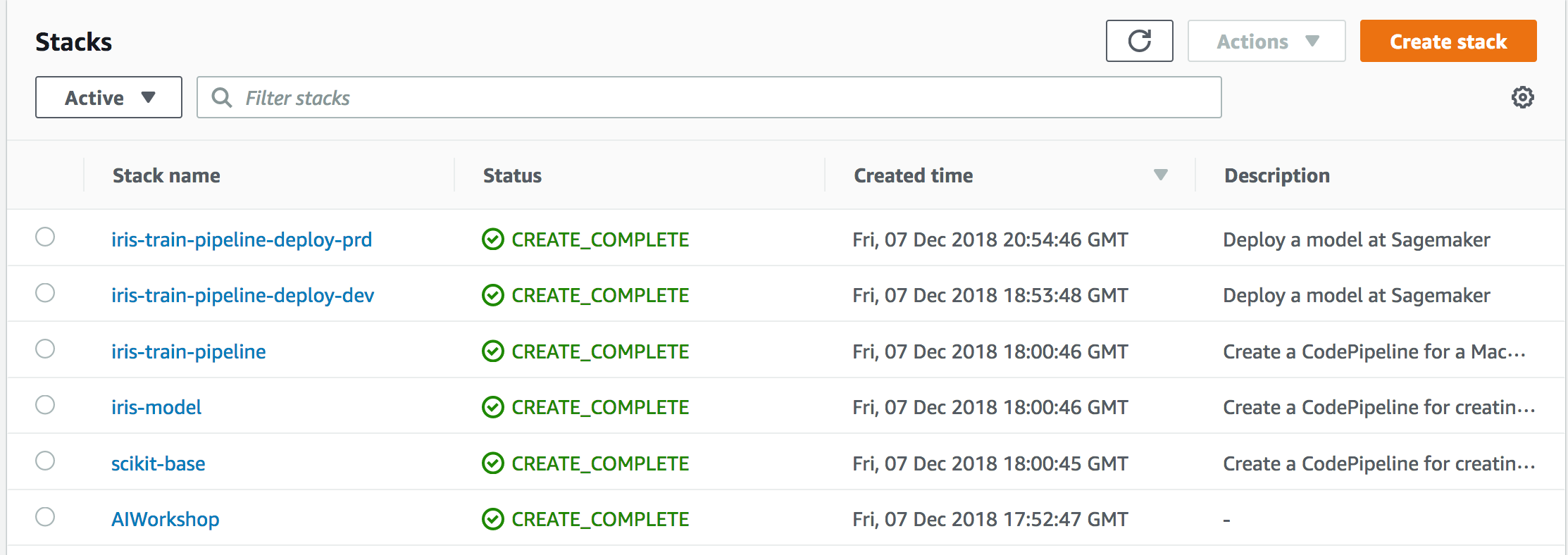
Here you can monitor the training process, approve the production deployment and test your endpoints.

## Module 7: Stress Test

Here you can execute stress tests to see how well your model is performing.

# 서비스 종료 가이드

워크 샵 이후 발생 되는 비용을 방지하기 위해서 아래의 단계에 따라 모두 종료/삭제 해 주세요.  
In order to delete all the assets, created by this workshop, delete the following cloudformation stacks.



Just follow the reverse order

1. AIWorkshop
2. scikit-base
3. iris-model
4. iris-train-pipeline
5. iris-train-pipeline-deploy-dev
6. iris-train-pipeline-deploy-prd

Also, delete the S3 bucket created by the first Cloudformation: mlops-<region>-<accountid>

* Notebook instance:

1) 만약 향후 사용을 위해 인스턴스를 저장하고 싶다면 **stop**을 하시면 됩니다. 이 경우 스토리지 비용은 발생합니다. 향후 다시 재가동 하시려면 Start button을 클릭하면 됩니다.

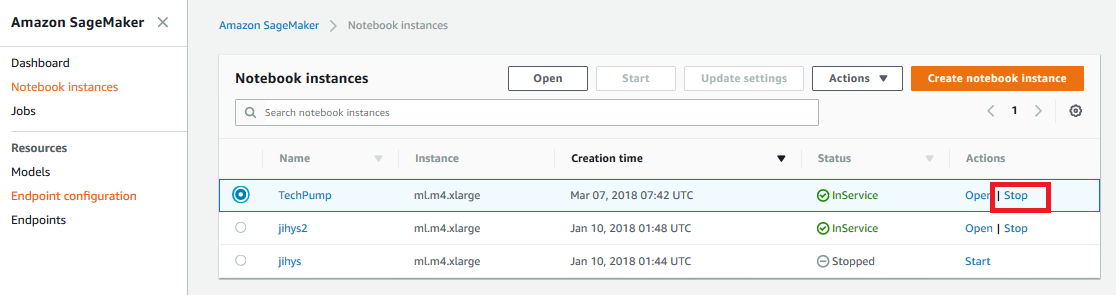


Figure 1. SageMaker 노트북 인스턴스 중단 화면.

2) 삭제를 할 경우는 **stop** 되어 있는 해당 notebook instance를 선택하고 **Action**  Dropdown 메뉴에서 **Delete** 선택 하시면 됩니다.

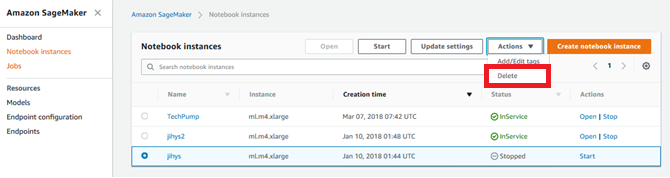


Figure 2. SageMaker 노트북 인스턴스 삭제 화면.

* SageMaker Endpoints:

훈련된 모델을 실제 예측 업무를 위해 배포된 한대 이상으로 구성된 클러스터입니다. Notebook안에서 명령어로 삭제하거나 SageMaker console에서 삭제 하실 수 있습니다. 삭제 하시기 위해서는 왼쪽 패널의 Endpoints를 선택 하신 후 해당 endpoints들 옆에 radio button을 클릭 하신 후 Action Dropdown 메뉴에서 Delete 선택 하시면 됩니다.

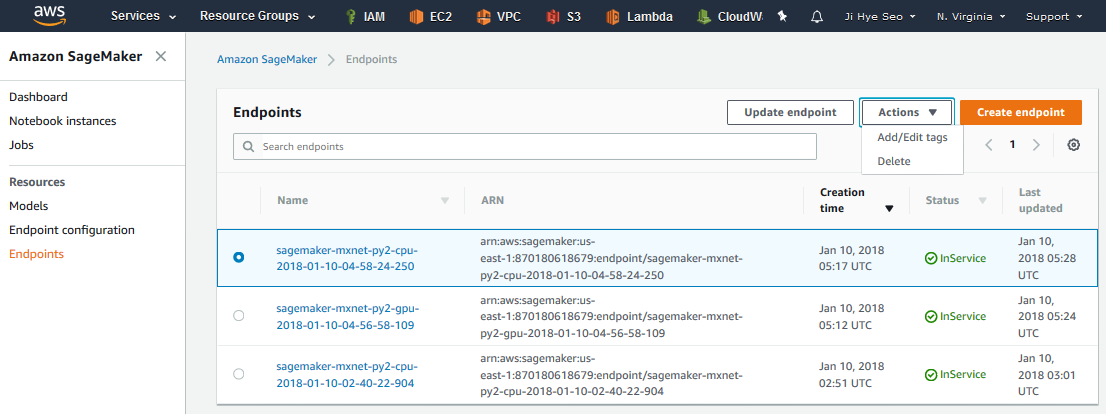


Figure 3. SageMaker Endpoint 삭제 화면.

* Lambda instance: 생성하신 Lambda instance를 삭제합니다.

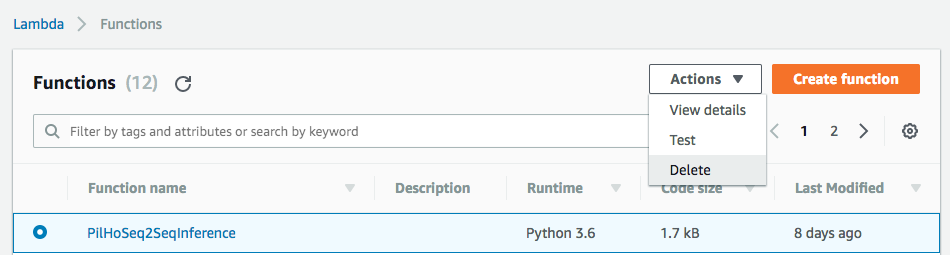


Figure 4. Lambda 인스턴스 삭제 화면.

* Amazon API Gateway instance: 생성하신 Gateway instance를 삭제합니다.

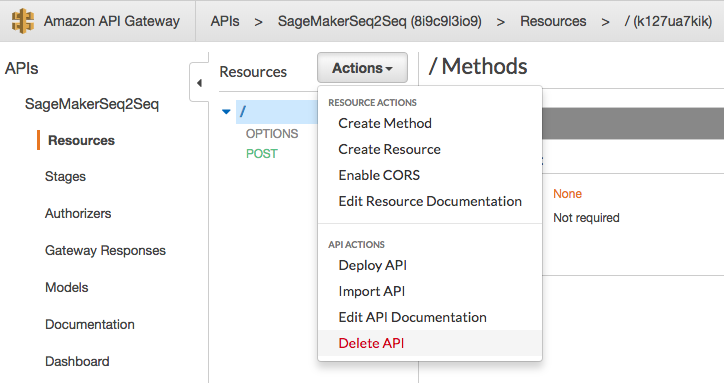


Figure 5. API Gateway 삭제 화면.

* Amazon S3 buckets: 생성하신 S3 Bucket (SageMaker용, Public Internet용)들을 모두 삭제합니다.

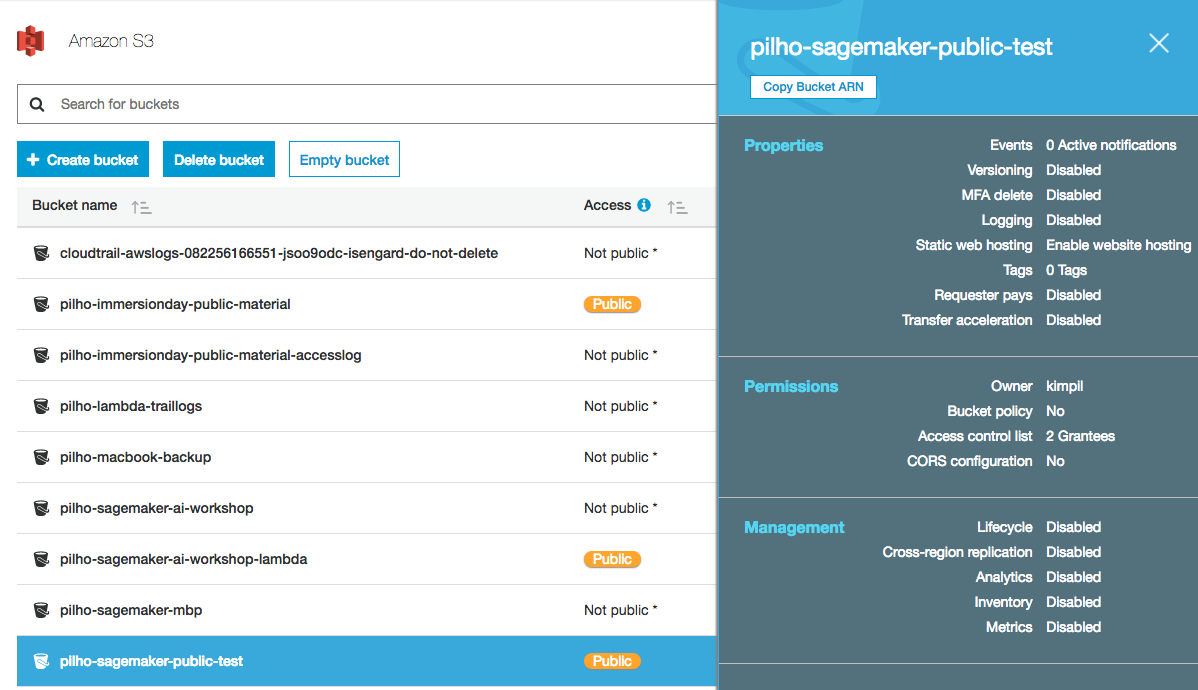


Figure 6. S3 버킷 삭제 화면.

이상으로 본 핸즈온 세션의 모든 과정을 마무리 하셨습니다. 수고하셨습니다.