Amazon SageMaker Hands-on Guidebook

26 Nov 2019

Table of Contents

Lab 개요

Amazon SageMaker 는 데이터 사이언티스트와 개발자들이 쉽고 빠르게 구성, 학습하고 어떤 규모로든 기계 학습된 모델을 배포할 수 있도록 해주는 관리형 서비스 입니다. 이 워크샵을 통해 SageMaker notebook instance 를 생성하고 샘플 Jupyter notebook 을 실습하면서 SageMaker 의 일부 기능을 알아보도록 합니다.

목표

- **SageMaker 에 내장된 학습 기능을 사용하여 모델 훈련 Job 을 생성 합니다.**
- **SageMaker 의 endpoint 기능을 사용하여 생성된 모델이 예측에 사용될 수 있도록 endpoint 를 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: SageMaker 는 지원되는 region 은 https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/ 에서 확인하실 수 있습니다. 이번 실습은 버지니아 북부 (us-east-1) region 에서 실행 합니다.**
- **Browser: 최신 버전의 Chrome, Firefox 를 사용하세요.**

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

LAB 실습 가이드

실습은 총 4 개 모듈로 구성되어 있습니다. 1 번 완료 후 다음 Lab 을 진행하셔야 합니다.

2,3,5,6 번 모듈은 원하는 순서대로 진행하실 수 있습니다. 다만 4 번 모듈은 3 번 모듈 진행 후 실행하셔야 합니다.

Module 1: Notebook Instance 생성하기

1. S3 Bucket 생성하기

SageMaker 는 S3 를 데이터와 모델 저장소로 사용합니다. 여기서는 해당 목적으로 S3 Bucket 을 생성합니다.

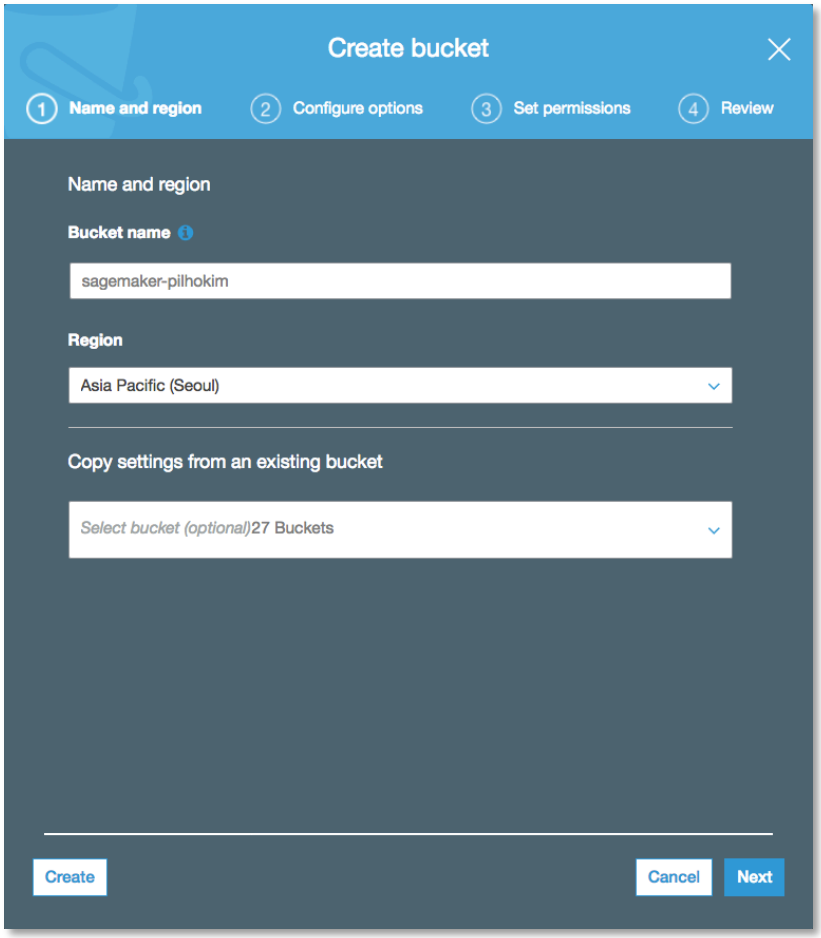
1) AWS 관리 콘솔 (https://console.aws.amazon.com/)에 Sign in 합니다.

2) AWS Services 리스트에서 S3 로 이동합니다.

3) "+ Create Bucket" 버튼을 선택합니다.

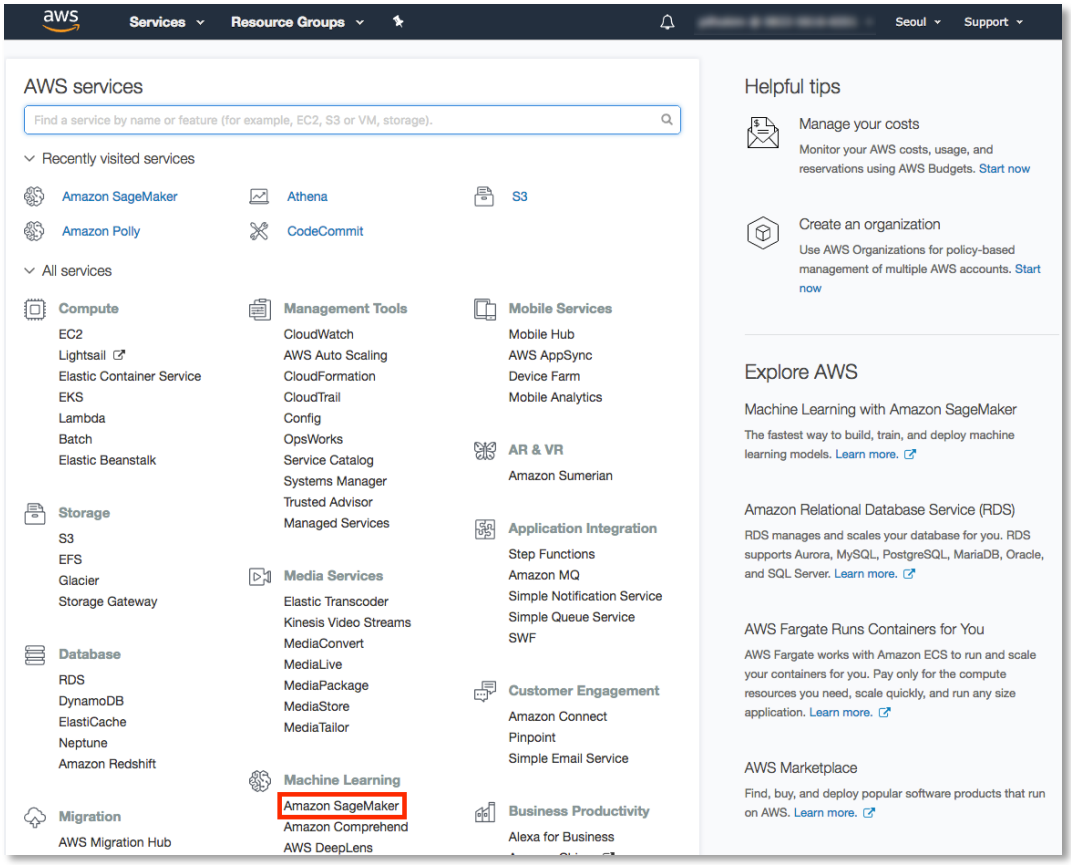
4) 아래 내용 설정 후 화면 왼쪽 아래 Create 클릭합니다.

- Bucket name: sagemaker-{userid} [반드시 고유한 값 설정]
- Region : US-East-1 (North Virginia)

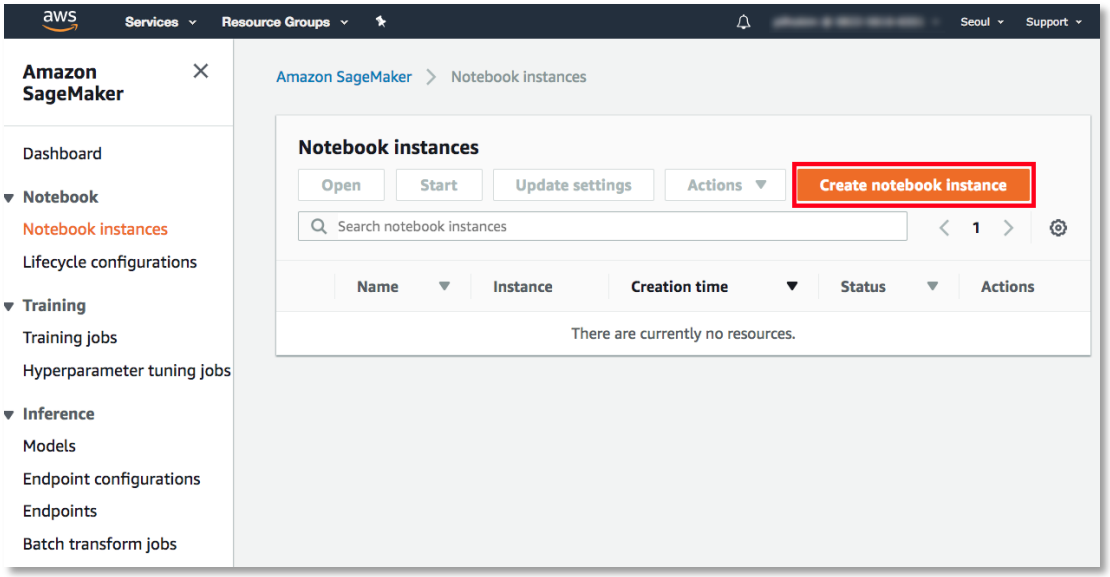


2. Notebook instance 생성

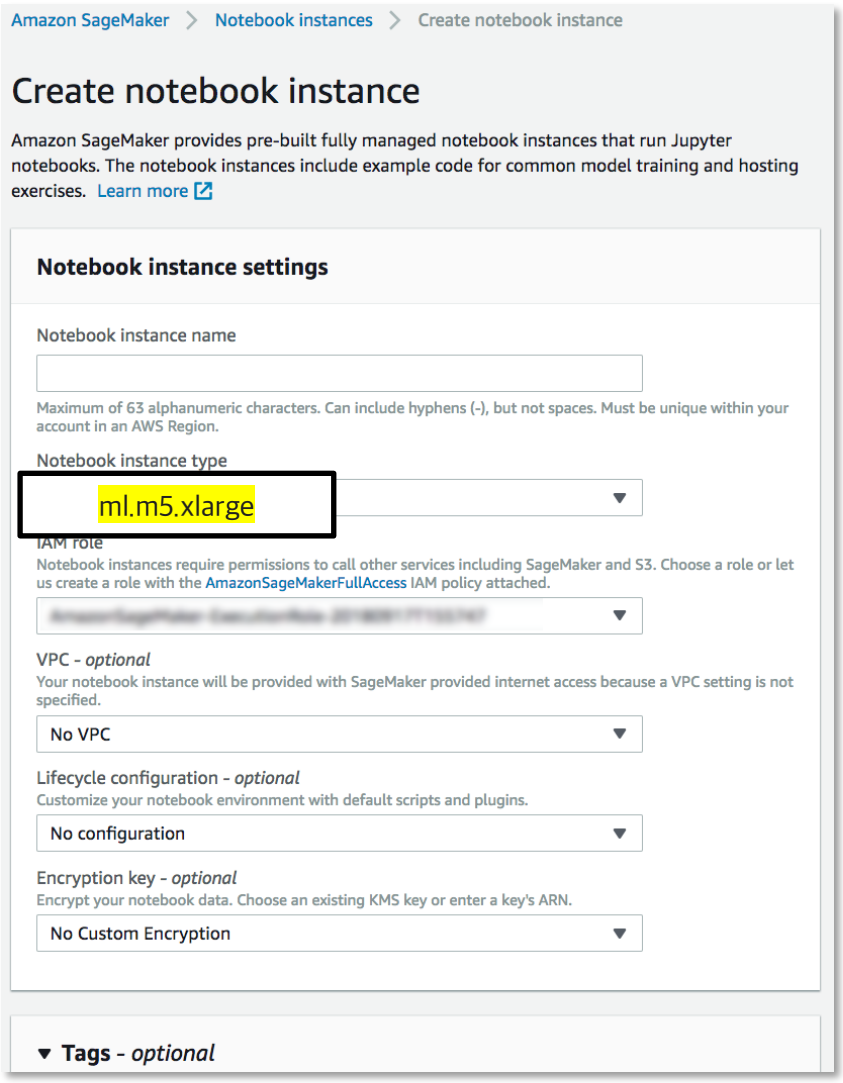
1) AWS 관리 콘솔에서 오른쪽 상단에서 North Virginia Region 선택 후 AWS Services 리스트에서 Amazon SageMaker 서비스를 선택합니다.



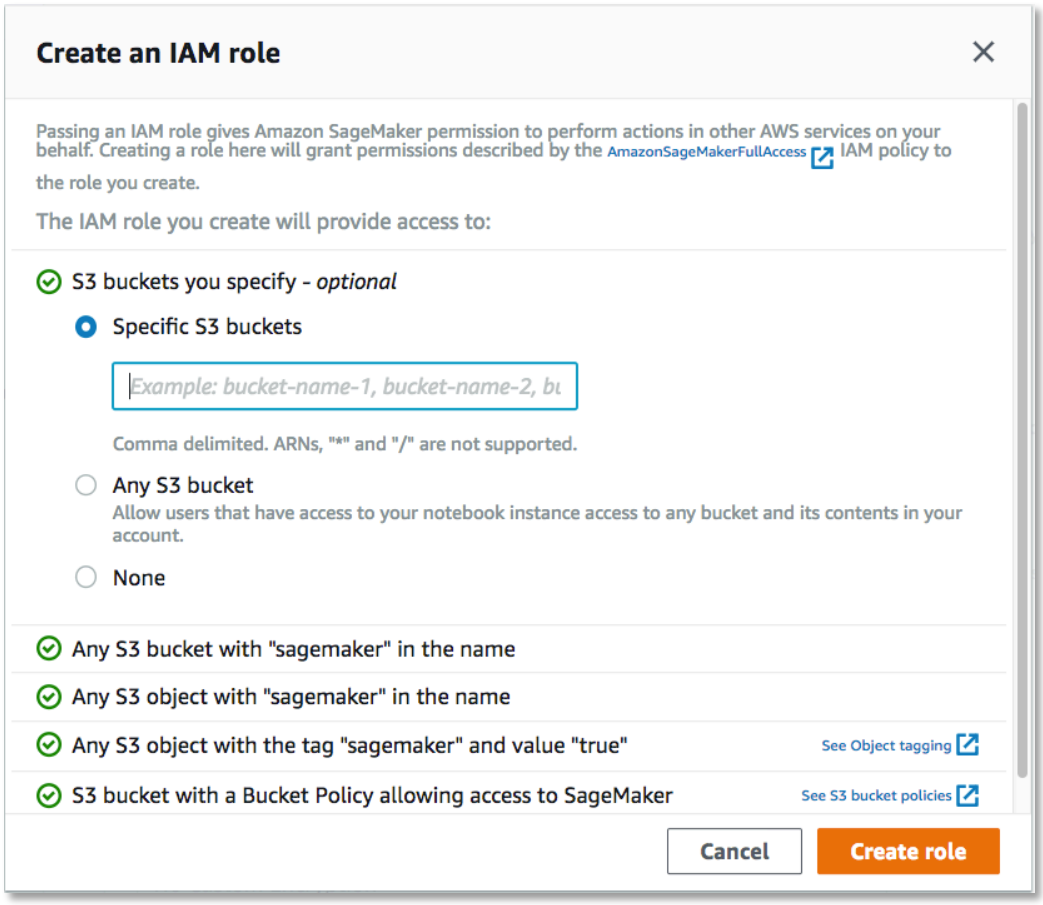
2) 새로운 Notebook instance 를 생성하기 위해 왼쪽 패널 메뉴 중 Notebook Instances 선택 후 오른쪽 상단의 Create notebook instance 버튼을 클릭 합니다.



3) Notebook instance 이름으로 [First Name]-[Last Name]-workshop 으로 넣은 뒤 ml.m5.xlarge 인스턴스 타입을 선택 합니다.



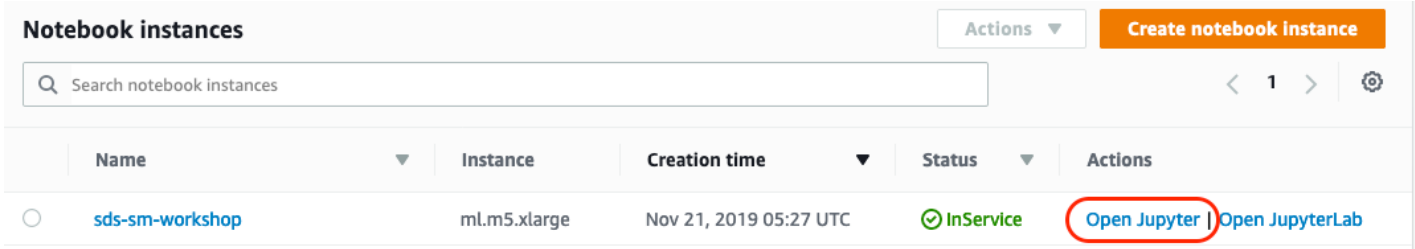
4) IAM role 은 Create a new role 을 선택하고, 생성된 팝업 창에서는 S3 buckets you specify – optional 밑에 Specific S3 Bucket 을 선택 합니다. 그리고 텍스트 필드에 위에서 만든 S3 bucket 이름(예: sagemaker-xxxxx)을 선택 합니다. 이후 Create role 을 클릭합니다.



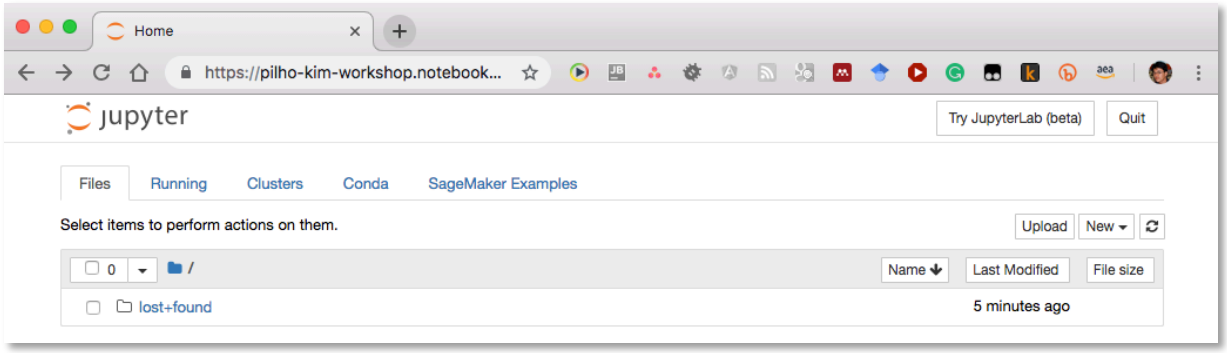
5) 다시 Create Notebook instance 페이지로 돌아온 뒤 Create notebook instance 를 클릭합니다.

3. Notebook Instance 접근하기

1) 서버 상태가 InService 로 바뀔 때까지 기다립니다. 보통 5 분정도의 시간이 소요 됩니다.



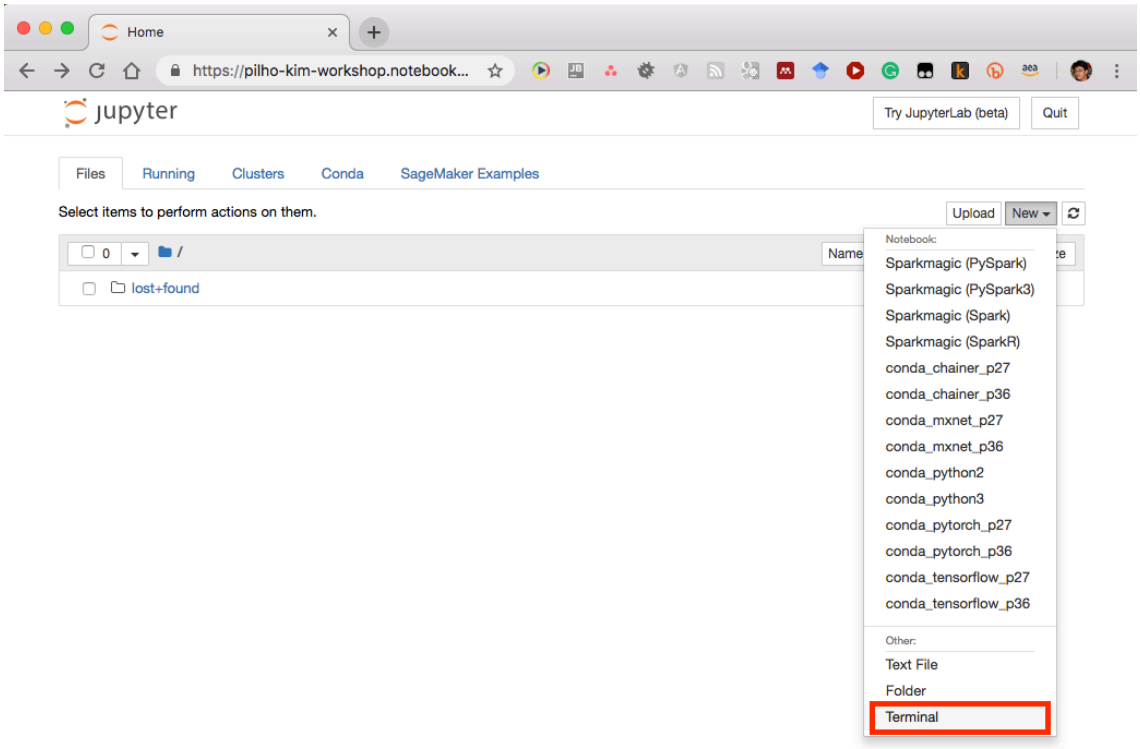
2) Open 을 클릭하면 방금 생성한 notebook instance 의 Jupyter 홈페이지로 이동하게 됩니다.



Moduel 2. XGBoost 를 활용한 비디오 게임 세일즈 예측

이 모듈에서는 Jupyter notebook 예제를 통해 어떻게 아마존이 제공하는 알고리즘을 SageMaker 에서 사용할 수 있는지 알아 봅니다. 특히 SageMaker 버전의 XGBoost 알고리즘을 사용하게 되는데, XGBoost 는 Gradient boosted decision tree 알고리즘을 구현한 유명하고 효율적인 오픈 소스버전입니다. Gradient boosting 은 supervised learning 알고리즘 중에 하나로 단순하고 weak 한 모델들의 예측치를 결합하여 타겟 변수를 예측합니다. XGBoost 는 다양한 종류의 데이터 타입과, 관계, 분산을 처리할 수 있기 때문에 많은 머신 러닝 경진 대회에서 우수한 결과를 낸 알고리즘 입니다. 이 알고리즘은 관계형 데이터 베이스 또는 Flat 파일등과 같은 정형 데이터를 다룰 경우 바로 사용 할 수 있는 알고리즘입니다.

1) SageMaker 의 Jupyter 노트북도 Linux 기반의 서버입니다. Jupyter 노트북에서 서버의 Terminal 을 바로 실행하는 기능을 제공하고 있습니다. **Error! Reference source not found.**와 같이 Terminal 을 선택합니다.



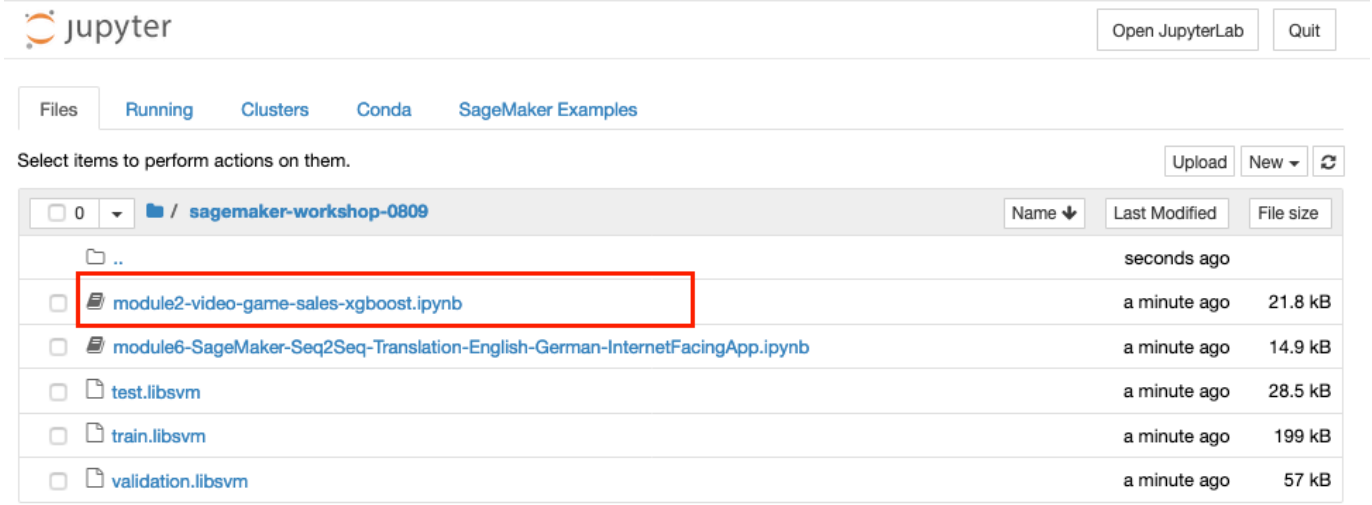
SageMaker 노트북 서버에 접속하기 위한 Terminal 실행 화면.

2) 터미널이 실행되면 아래의 명령어들을 입력해서 실행합니다.

```
cd SageMaker/
git clone https://github.com/jihys/sagemaker-workshop-0809
```

3) 실습을 위해서 현재 설치되어 있는 SageMaker 의 Jupyter 노트북의 예제들 중 아래의 디렉토리에 위한 Jupyter 노트북을 실행하시면 됩니다.

/sagemaker-workshop-0809/module2-video-game-sales-xgboost.ipynb



4) 두번째 Cell 에서 **bucket= '<bucket-name>'** 라인에서 **<bucket-name>**을 모듈 1 에서 만든 S3 bucket 이름(예: sagemaker-xxxxx)을 적고 주석을 해제합니다. S3://....와 같은 경로 이름은 적지 않습니다.

** 변수 bucket 2 개 중 하나만 사용해야 합니다.

Setup

Let's start by:

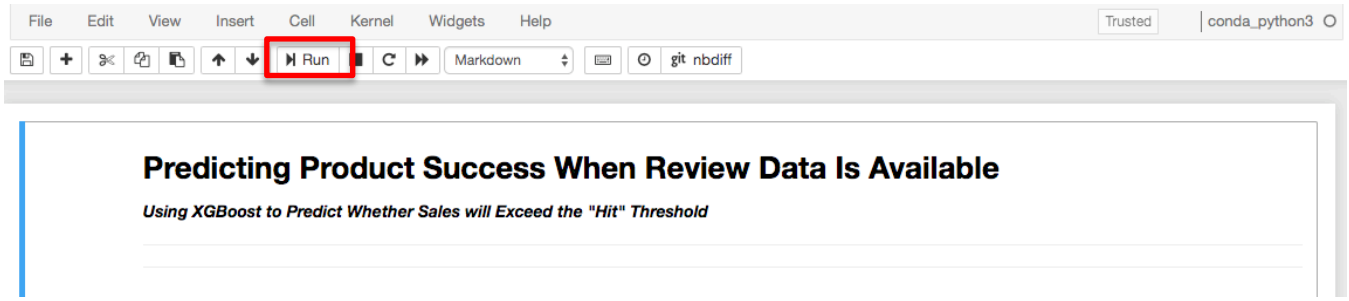
- Importing various Python libraries we'll need.
- Instantiate a SageMaker session for various tasks within this notebook, and get the AWS Region.
- Specifying a S3 bucket and bucket prefix to use for training and model data.
- Defining an IAM role for S3 data access, which is pulled in from the SageMaker notebook instance.

In [ ]:
```
import timeit
start_time = timeit.default_timer()
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from sklearn.datasets import dump_svmlight_file
from time import gmtime, strftime
import sys
import math
import json
import boto3
import sagemaker

session = sagemaker.Session()
region = session.boto_region_name
#bucket='<bucket-name>'
bucket = session.default_bucket()
prefix = 'sagemaker/videogames-xgboost'
role = sagemaker.get_execution_role()

print('Bucket:\n{}'.format(bucket))
```

Module 1 에서 생성한 Bucket 이름으로 바꿔주세요

5) 첫번 째 셀의 Code 부터 실행 버튼을 클릭합니다.
Jupyter notebook 은 코드와 주석을 같이 저장합니다. Jupyter notebook 에는 두 가지의 Cell(Code Cell 과 markdown Cell)이 있습니다. Code 를 실행하려면 실행 버튼을 클릭합니다. (Control+Enter 도 동일한 기능이며, 실제 사용하는 실행 후 셀을 이동하는 Shift+Enter 가 더 편리합니다.)



Code 가 실행되면 Code Cell 왼쪽의 "In [ ]" 라는 부분이 "In [*]"로 변경이 되고 완료시에는 실행 순서를 나타내는 숫자로 변경 됩니다.

※ 코드는 Code Cell 에 나타난 순서대로 실행하고 반복 작업을 피하기 위해서 한 번만 실행합니다. 같은 훈련 job cell 을 반복 실행하게 되면 두 개의 훈련 job 을 실행하게 되어 서비스 제한을 넘을 수도 있습니다.

※ 이 모델을 훈련하는데는 약 10 분에서 15 분이 소요됩니다.

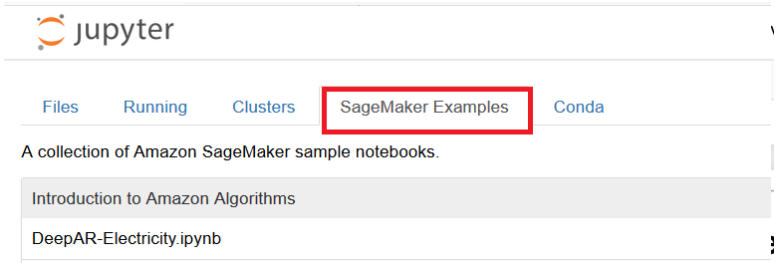
※ (참고) 본 노트북 한글버전

https://github.com/aws-samples/aws-ai-ml-workshop-kr/blob/master/contribution/seongshin/aws-ai-ml-immersionday-kr/video-game-sales-xgboost.ipynb

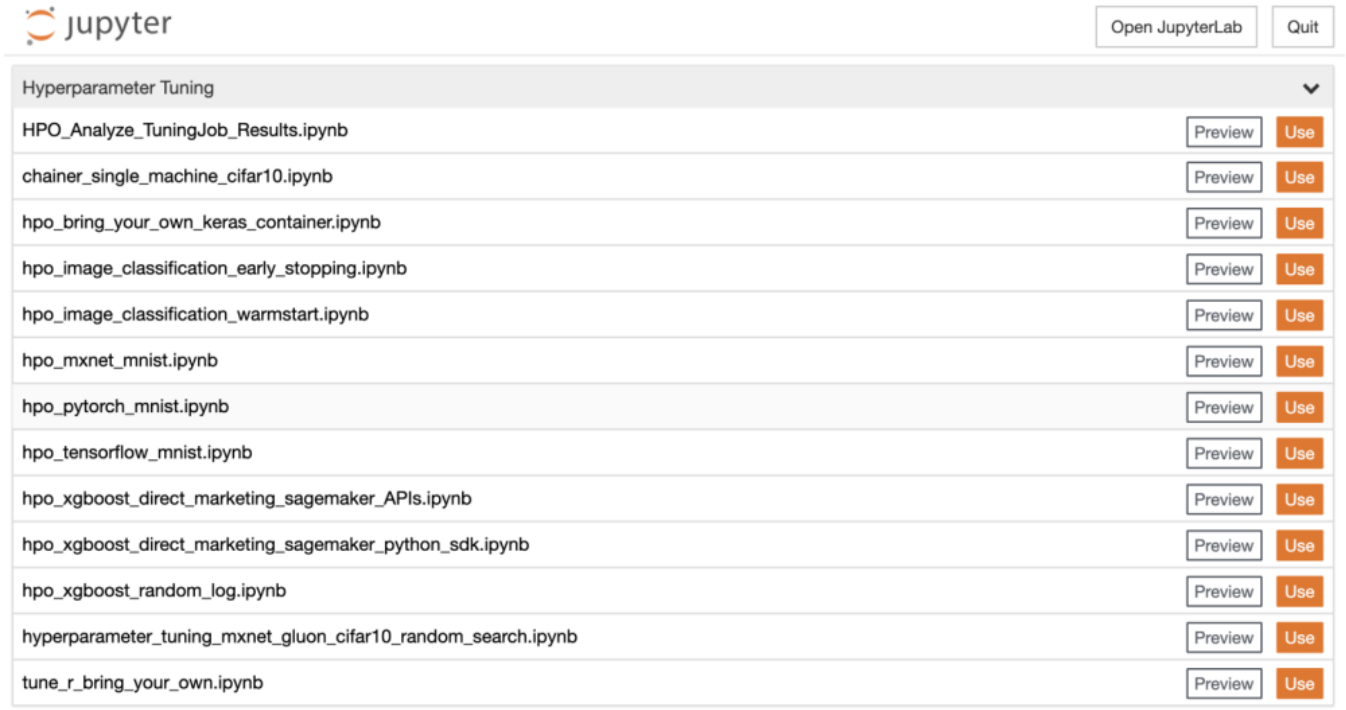
Module 3: TensorFlow MNIST 로 자동 모델 튜닝하기

이 모듈에서는 TensorFlow MNIST 이미지 분류 예제를 기반으로 SageMaker 의 자동 모델 튜닝 기능을 실습합니다. 이 기능은 기계 학습 알고리즘의 최적의 하이퍼파라미터 (Hyperparameter) 값을 베이지안 최적화 기법을 통해 찾아줍니다.

1) SageMaker 의 Jupyter 노트북 페이지 상단의 탭메뉴에서 "SageMaker Examples"를 클릭합니다.

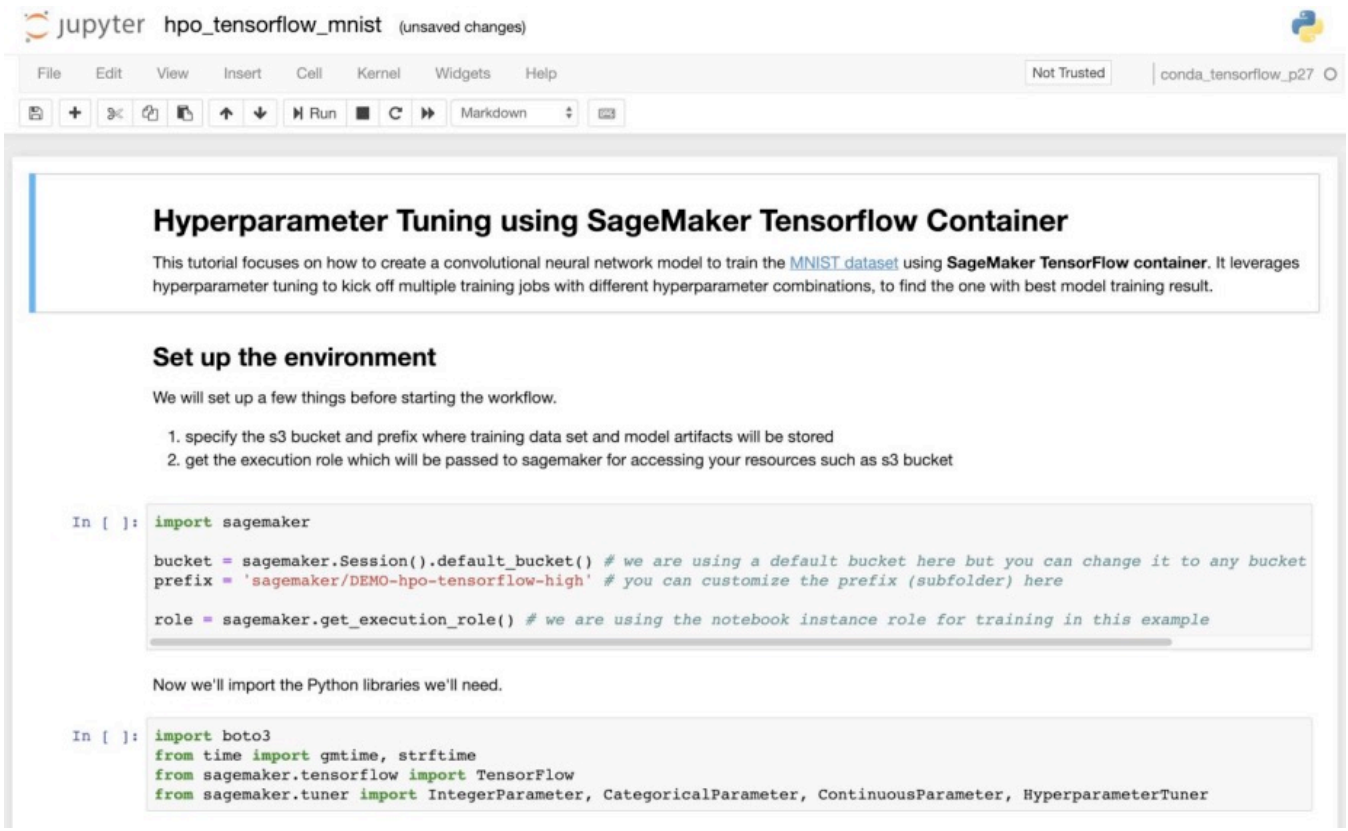


2) 샘플 노트북 목록에서 "Hyperparameter Tuning"을 선택합니다.

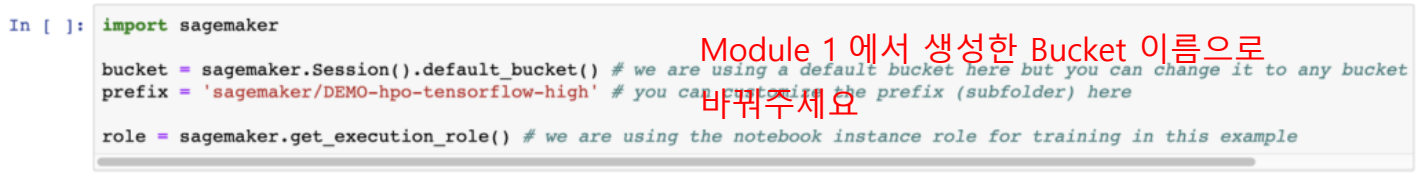


3) 샘플 목록중 "hpo_tensorflow_mnist.ipynb" 를 찾아 우측의 Use 버튼을 클릭합니다. 다음과 같은 팝업창에서 Create copy 버튼을 클릭하여 관련 파일들을 사용자의 홈디렉토리로 복사를 진행합니다.

새로운 브라우저 탭에서 노트북이 오픈되면 준비가 완료됩니다.



4) 모듈 실행중 아래 코드를 만나면 bucket = sagemaker.Session().default_bucket() 라인을
bucket = '<모듈 1 에서 생성한 s3 버킷의 이름(예: sagemaker-xxxxx)>' 으로 수정합니다.
부등호 부호('<', '>')는 넣지 않습니다.



5) 이 모듈에서는 MNIST 이미지 분류 예제의 하이퍼파라미터 중에서 learning rate 값을 자동으로 튜닝하며, 효과적인 탐색을 위해 최대값과 최소값을 아래 그림과 같이 설정합니다.

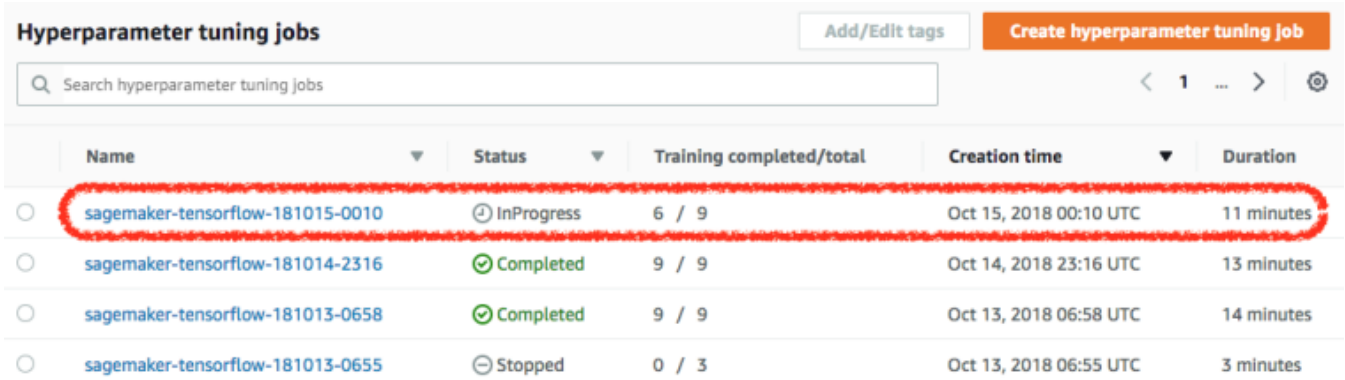
6) 베이지안 최적화 기법은 하이퍼파라미터를 변경하면서 미리 지정된 숫자만큼 실험을 반복하는 특징이 있습니다. 이번 모듈에서는 병렬로 3 개의 학습을 3 번, 즉 총 9 번의 실험을 시도하도록 아래와 같이 설정합니다.

```
In [10]: tuner = HyperparameterTuner(estimator,
                                     objective_metric_name,
                                     hyperparameter_ranges,
                                     metric_definitions,
                                     max_jobs=9,
                                     max_parallel_jobs=3,
                                     objective_type=objective_type)
```

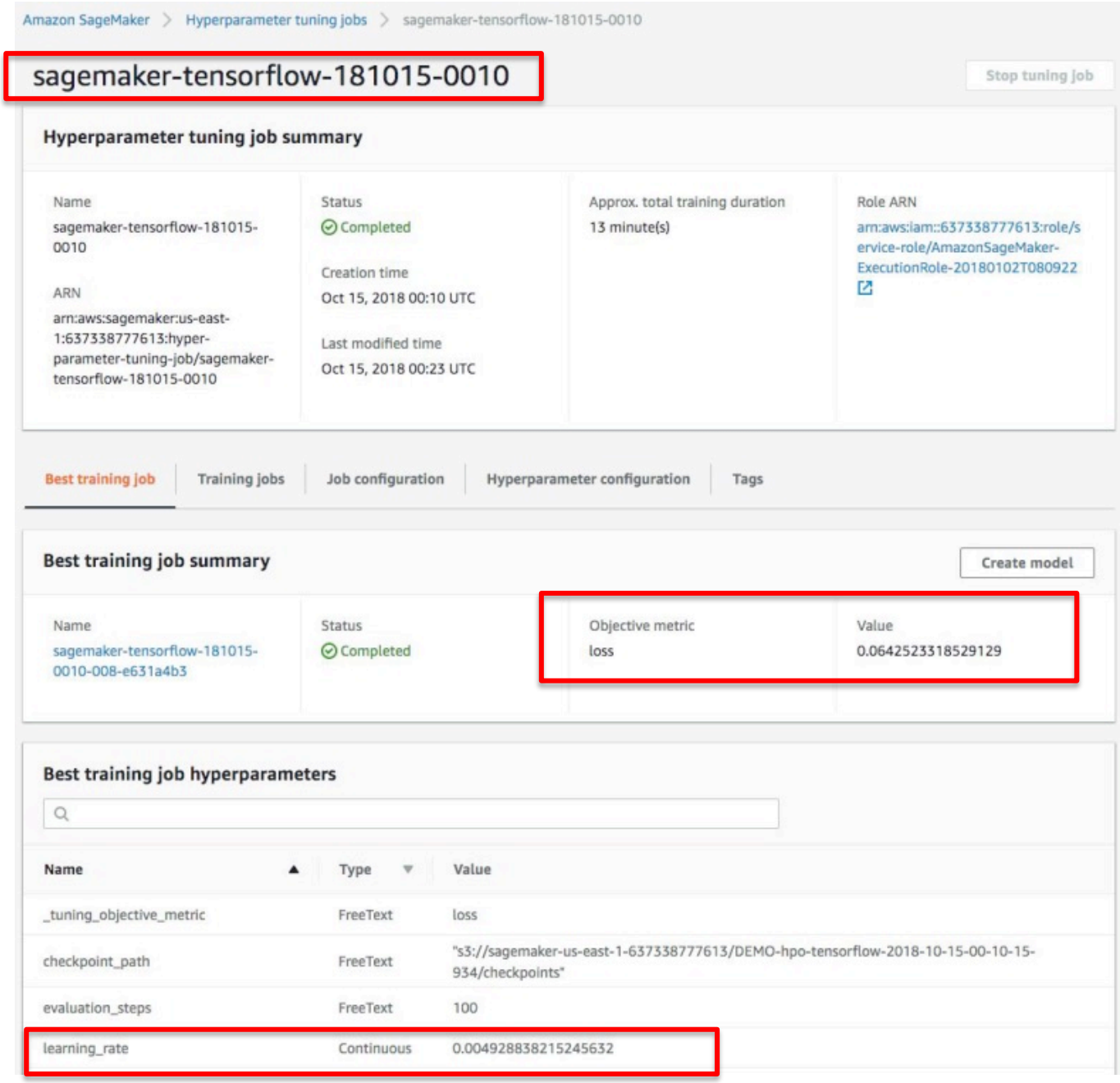
7) 하이퍼파라미터 튜닝 작업은 아래와 같은 코드로 실행하며, 실행하면 각 하이퍼파라미터 값에 대한 개별 학습이 백그라운드에서 시작됩니다.

```
In [12]: boto3.client('sagemaker').describe_hyper_parameter_tuning_job(
             HyperParameterTuningJobName=tuner.latest_tuning_job.job_name)['HyperParameterTuningJobStatus']

Out[12]: u'InProgress'
```

8) 이 때, SageMaker 의 콘솔에서 새로운 하이퍼파라미터 튜닝 작업 (Hyperparameter tuning jobs)이 생성된 것을 확인할 수 있습니다. 다음 모듈을 위해 이 작업의 이름을 메모해 놓습니다.



9) 실험이 모두 끝나면 하이퍼파라미터 튜닝 작업의 이름을 클릭해 튜닝 결과를 확인합니다. 아래 그림에서는 learning_rate 가 0.004928838215245632 가 최적의 값이며 이때의 loss 값은 0.0642523318529129 인 것을 확인할 수 있습니다.



이 노트북의 소스 파일은 아래 GitHub 에 공개되어 있습니다.

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/hyperparameter_tuning/tensorflow_mnist

※ (참고) 본 노트북 한글버전

https://github.com/aws-samples/aws-ai-ml-workshop-kr/blob/master/contribution/seongshin/aws-ai-ml-immersionday-kr/tensorflow_mnist/hpo_tensorflow_mnist.ipynb

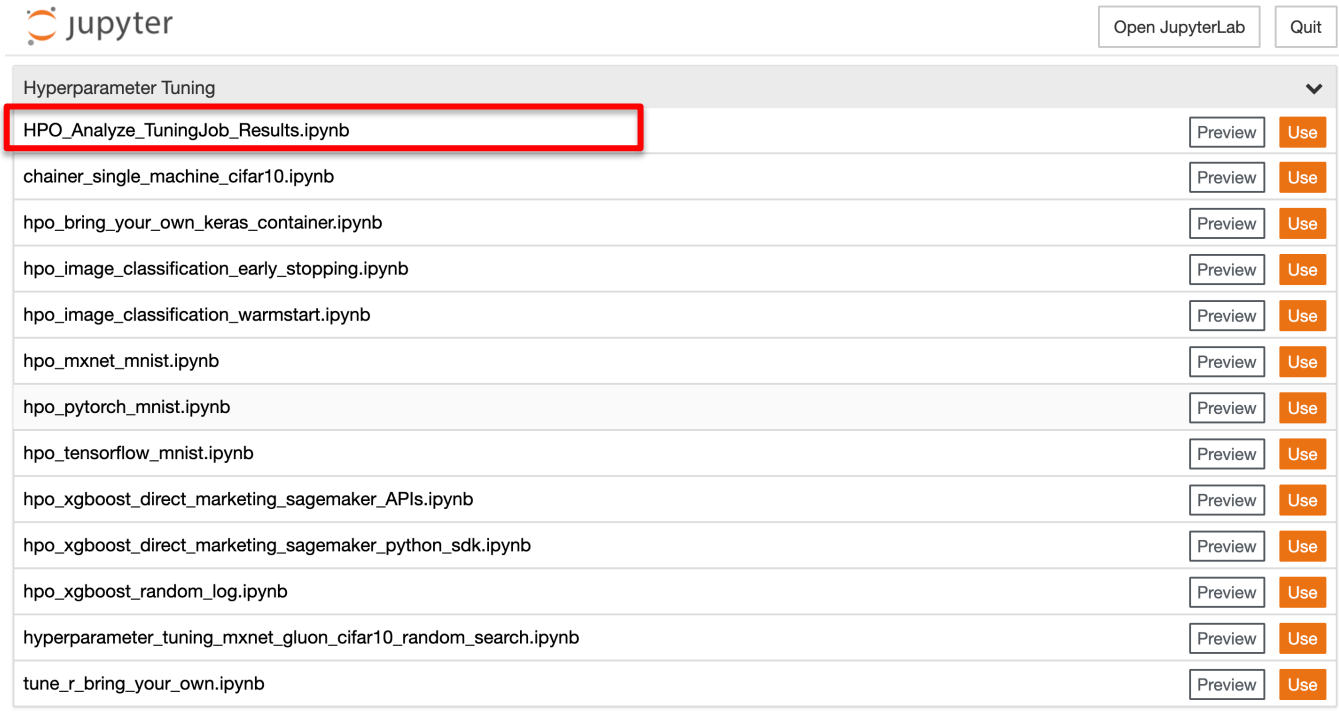
※ 이 모델을 훈련하는데는 약 20 분에서 25 분이 소요됩니다.

Module 4: 자동 모델 튜닝 결과 분석하기

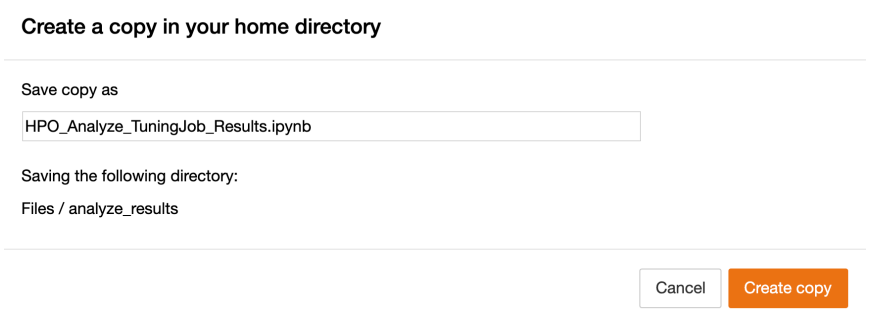
이 모듈에서는 앞에서 실행한 하이퍼파라미터 튜닝 작업의 결과를 해석하는 과정을 실습합니다. BokehJS 와 pandas 라이브러리를 사용해 튜닝 결과를 Jupyter 노트북에서 테이블과 그래프 형태로 시각화해볼 수 있습니다.

1) SageMaker 의 Jupyter 노트북 페이지 상단의 탭메뉴에서 "SageMaker Examples"를 클릭합니다.

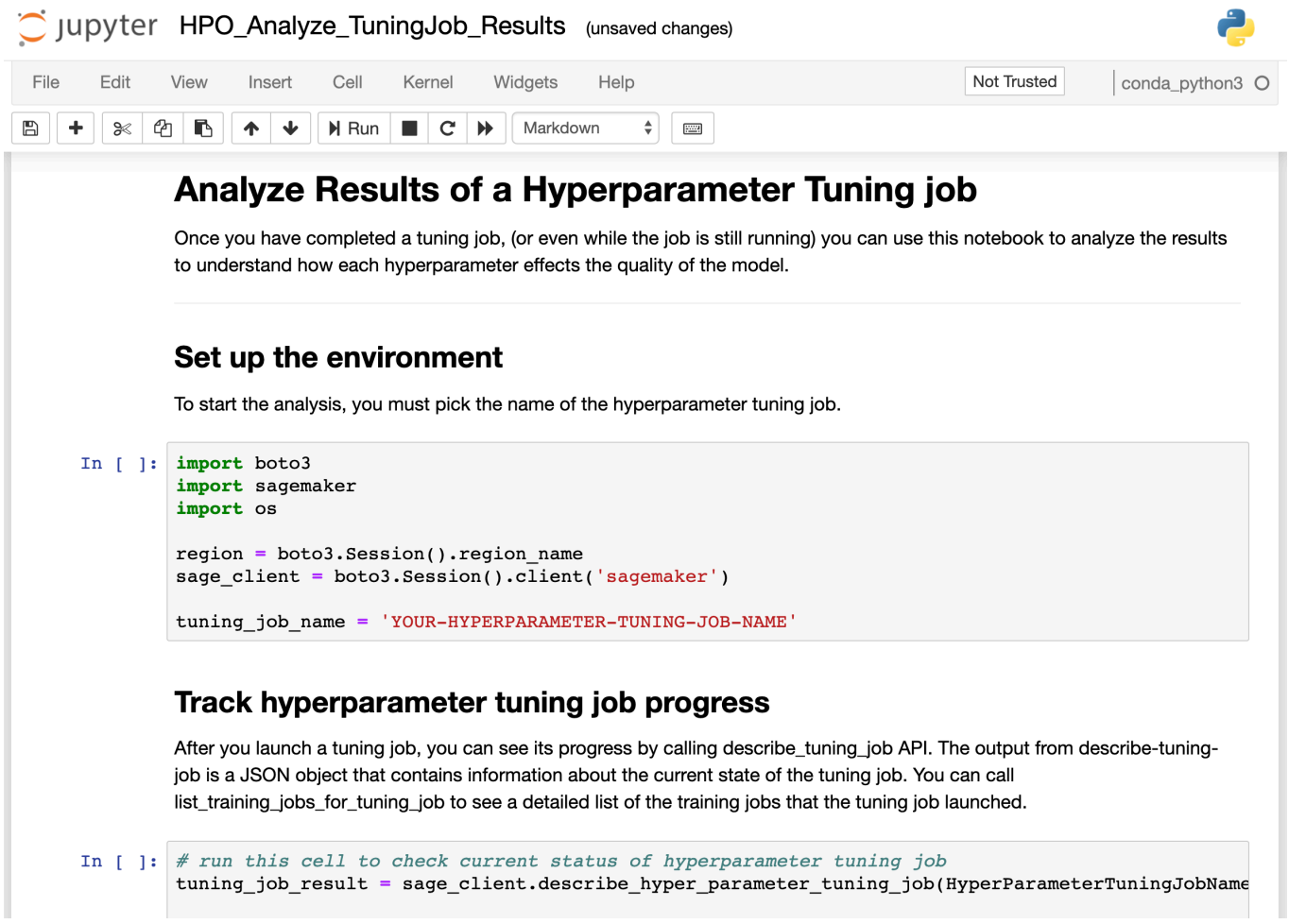
2) 샘플 노트북 목록에서 "Hyperparameter Tuning"을 선택합니다.



샘플 목록중 "HPO_Analyze_TuningJob_Results.ipynb" 를 찾아 우측의 Use 버튼을 클릭합니다. 다음과 같은 팝업창에서 **Create copy** 버튼을 클릭하여 관련 파일들을 사용자의 홈디렉토리로 복사를 진행합니다.



3) 새로운 브라우저 탭에서 노트북이 오픈되면 다음과 같이 준비가 완료됩니다.



4) 모듈의 첫 부분에서 아래의 코드를 만나면 앞 모듈에서 실행된 **하이퍼파라미터 튜닝 작업 (Hyperparameter tuning jobs)**의 이름을 따옴표 안에 넣습니다.

```
In [ ]: import boto3
        import sagemaker
        import os

        region = boto3.Session().region_name
        sage_client = boto3.Session().client('sagemaker')

        tuning_job_name = 'YOUR-HYPERPARAMETER-TUNING-JOB-NAME'
```

5) 실행 결과로 나오는 두 개의 그래프에서, 탐색된 하이퍼파라미터 값의 변화에 따른 loss 함수 값의 변화를 해석해 보시기 바랍니다. (그래프가 안나오면 해당 셀을 다시 한번 실행해 보세요.)

이 노트북의 소스 파일은 아래 GitHub 에 공개되어 있습니다.

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/hyperparameter_tuning/analyze_results

※ (참고) 본 노트북 한글버전

https://github.com/aws-samples/aws-ai-ml-workshop-kr/blob/master/contribution/seongshin/aws-ai-ml-immersionday-kr/analyze_results/HPO_Analyze_TuningJob_Results.ipynb

※ 이 모듈의 실습에는 약 5 분이 소요됩니다.

Module 5: Bring-your-own-container 기능 실습하기

Amazon SageMaker 는 머신 러닝 모델을 훈련하고 배포하기 위해 Docker container 를 사용합니다. SageMaker 에서 현재 지원하고 있지 않는 알고리즘이나 머신 러닝 프레임워크, 또는 여러분이 로컬 환경에서 개발한 모델이라도 Docker container 로 만들어 SageMaker 에서 훈련하고 배포할 수 있습니다. 이번 모듈에서는 SageMaker 환경에서 훈련과 배포를 할 수 있도록 Docker container 로 패키징 하는 방법에 대해 실습합니다.

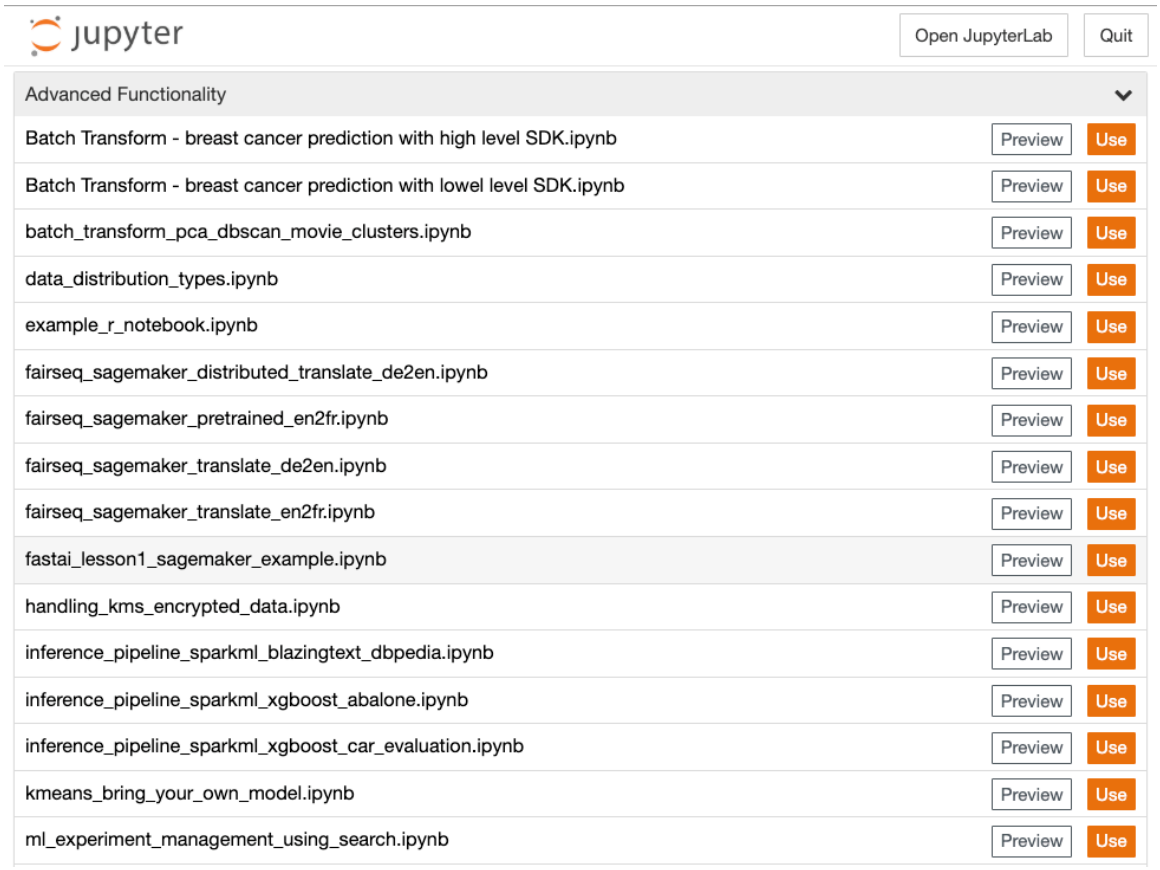
이번 모듈에서는 Scikit-Learn 으로 작성된 모델을 컨테이너로 패키징 해봅니다. 이 내용은 아래 AWS 블로그에 잘 설명되어 있습니다.

- https://aws.amazon.com/blogs/machine-learning/train-and-host-scikit-learn-models-in-amazon-sagemaker-by-building-a-scikit-docker-container/
- [한글버전] https://aws.amazon.com/ko/blogs/korea/train-and-host-scikit-learn-models-in-amazon-sagemaker-by-building-a-scikit-docker-container/

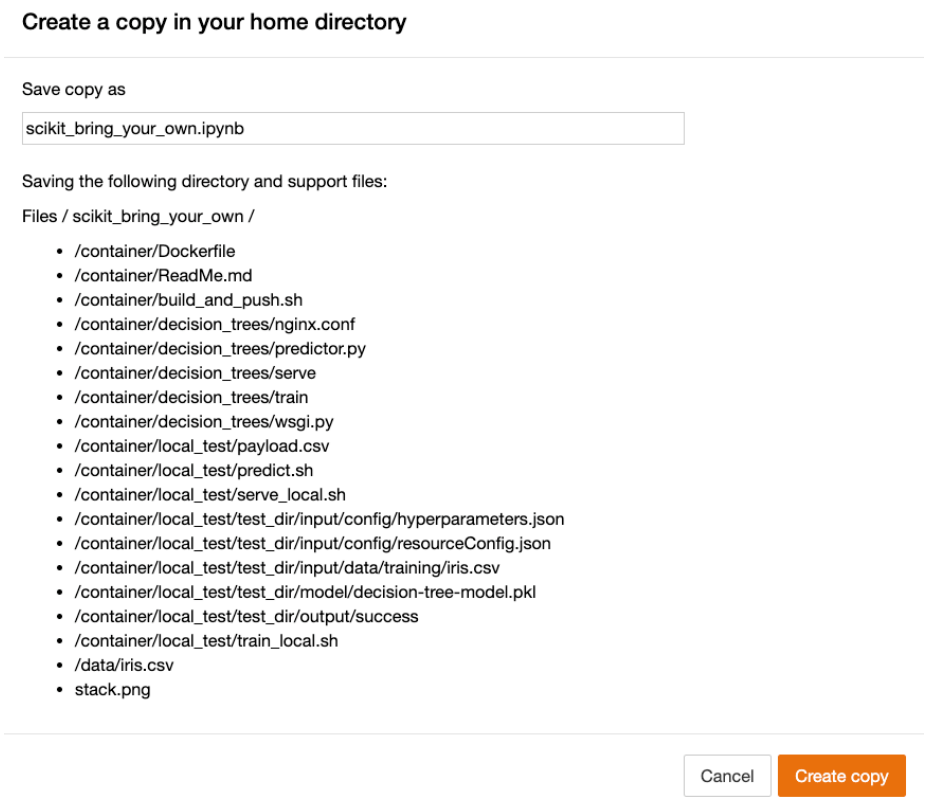
참고로 현재 SageMaker 는 Scikit-Learn 컨테이너를 제공하고 있기 때문에, Scikit-Learn 모델을 사용하기 위해 매번 이 모듈의 내용대로 새로운 컨테이너를 만들 필요는 없습니다. Scikit-Learn 을 사용하는 예제에 대해서는 아래의 노트북을 참고하시기 바랍니다.

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/sagemaker-p ython-sdk/scikit_learn_iris

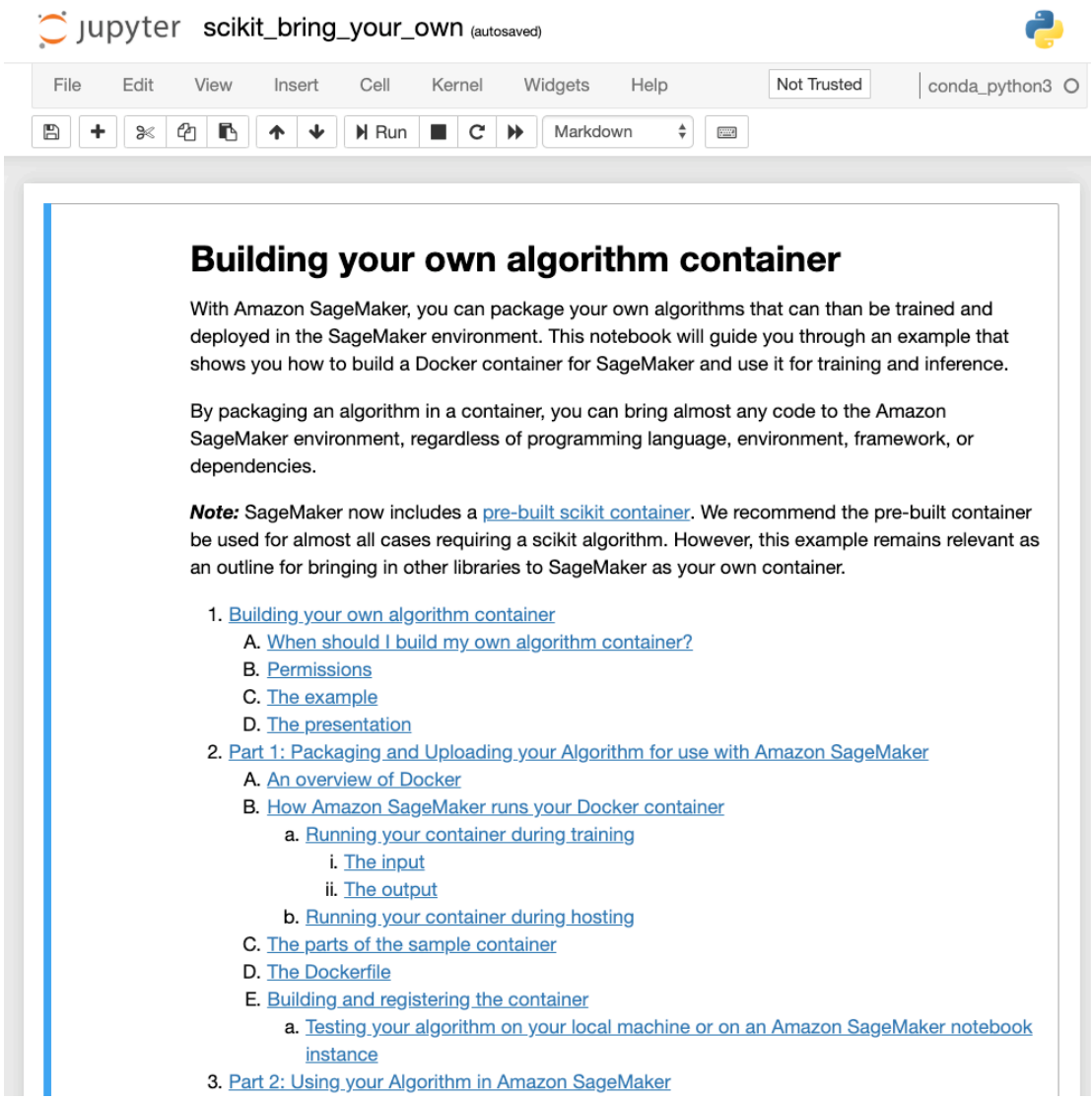
1) 샘플 노트북 목록에서 "Advanced Functionality"을 선택합니다.



샘플 목록중 "scikit_bring_your_own.ipynb" 를 찾아 우측의 Use 버튼을 클릭합니다. 다음과 같은 팝업창에서 **Create copy** 버튼을 클릭하여 관련 파일들을 사용자의 홈디렉토리로 복사를 진행합니다.



2) 새로운 브라우저 탭에서 노트북이 오픈되면 준비가 완료됩니다.



3) 모듈 실행중 아래 코드를 만나면 **<your_s3_bucket_name_here>** 부분에 모듈 1 에서 생성한 s3 버킷의 이름(예: sagemaker-xxxxx)을 넣고 실행합니다. 부등호 부호('<', '>')는 넣지 않습니다.

```
In [ ]: bucket = '<your_s3_bucket_name_here>'
        prefix = 'sagemaker/DEMO-linear-time-series-forecast'

        # Define IAM role
        import boto3
        import re
        from sagemaker import get_execution_role

        role = get_execution_role()
```

이 노트북의 소스 파일은 아래 GitHub 에 공개되어 있습니다.

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/advanced_functionality/scikit_bring_your_own

※ (참고) 본 노트북 한글버전

https://github.com/aws-samples/aws-ai-ml-workshop-kr/blob/master/contribution/seongshin/aws-ai-ml-immersionday-kr/scikit_bring_your_own/scikit_bring_your_own.ipynb

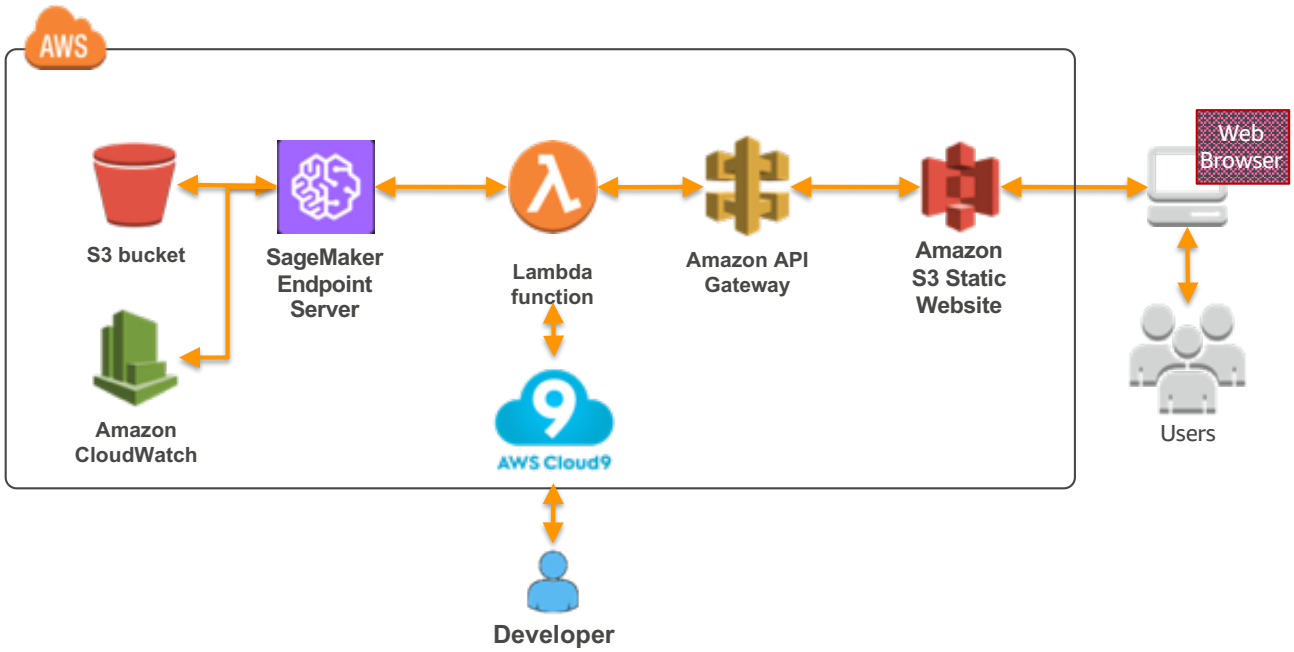
※ 이 모듈의 실습에는 약 30 분이 소요됩니다.

[Option] Module 6: Internet-facing 앱 개발

Amazon SageMaker 는 데이터 사이언티스트와 개발자들이 쉽고 빠르게 구성, 학습하고 어떤 규모 로든 기계 학습된 모델을 배포할 수 있도록 해주는 관리형 서비스 입니다. 이 워크샵을 통해 Sagemaker notebook instance 를 생성하고 샘플 Jupyter notebook 을 실습하면서 SageMaker 의 일부 기능을 알아보도록 합니다 .

이 모듈에서는 영어를 독일어로 변환하는 SageMaker 의 Sequence-to-Sequence 알고리즘을 이용한 언어번역기를 학습해보고 이 서비스를 인터넷을 통해 활용할 수 있는 방법에 대해 실습해 보겠습니다.

본 Hands-on 에서는 SageMaker 에서 생성한 Endpoint inference service 를 웹 상에서 호출하기 위해 AWS Lambda 와 AWS API Gateway 를 사용하여 아래와 같은 데모를 만들어 보겠습니다.



SageMaker Internet-facing App Data Flow.

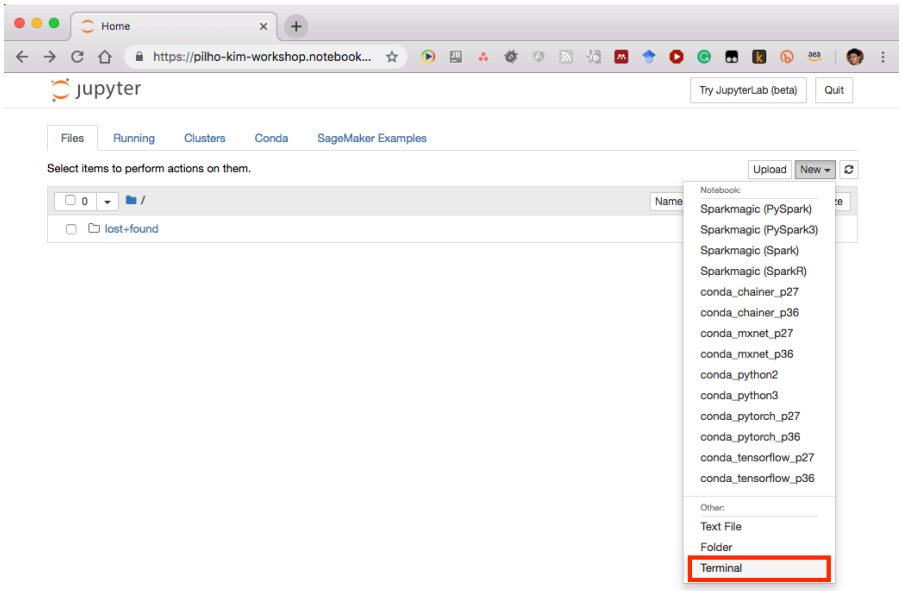
SageMaker 의 기능 데모를 위해 가장 간략한 구조를 채택하고 있습니다. 예를 들어 Amazon S3 의 Static Website 에 다른 도메인 이름을 지정하기 위한 Route 53 서비스나 캐싱 서비스를 위한 CloudFront 등의 서비스는 실제 비즈니스 적용 시에는 고려 되어야할 서비스입니다.

전제 Lab 시간은 일반 사용자의 경우 한시간에서 한시간 30 분정도 소요 예상 됩니다.

Module 6-1: 영어-독어 번역 ML 모델 학습

Sequence-to-Sequence 알고리즘 노트북 다운로드 받기

SageMaker 의 Jupyter 노트북도 Linux 기반의 서버입니다. Jupyter 노트북에서 서버의 Terminal 을 바로 실행하는 기능을 제공하고 있습니다.아래와 같이 Terminal 을 선택합니다.



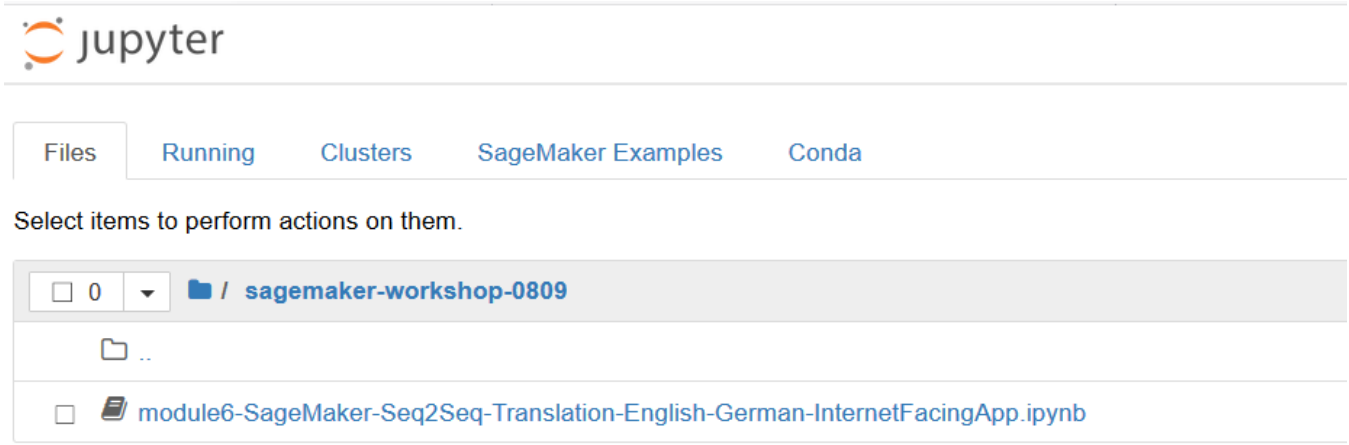
터미널이 실행되면 아래의 명령어들을 입력해서 실행합니다.

```
cd SageMaker/
git clone https://github.com/jihys/sagemaker-workshop-0809.git
```

SageMaker 가 지원하는 Seq2Seq 알고리즘은 MXNet 기반으로 개발된 Sockeye 알고리즘을 기반으로 개발된 최신의 Encoder-decoder 구조를 구현한 것으로 문서자동요약이나 언어 번역 서비스에 적용할 수 있습니다.

실습을 위해서 현재 설치되어 있는 SageMaker 의 Jupyter 노트북의 예제들 중 아래의 디렉토리에 위한 Jupyter 노트북을 실행하시면 됩니다

/ sagemaker-workshop /module6-SageMaker-Seq2Seq-Translation-English-German-InternetFacingApp.ipynb

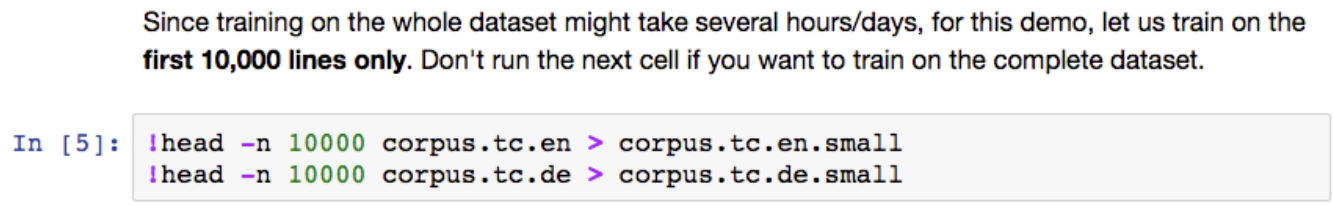


노트북에 대한 설명

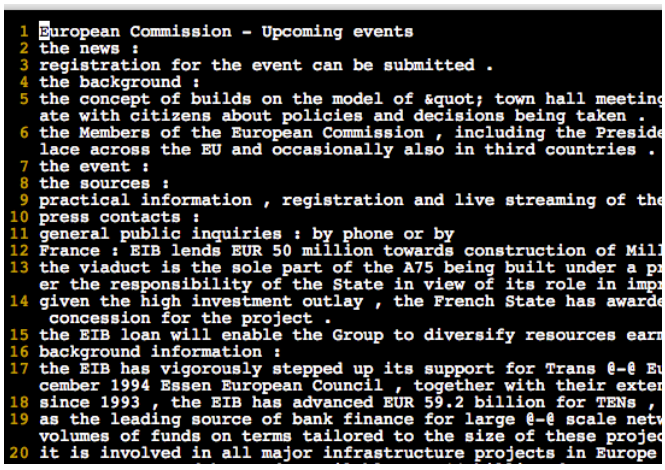
본 노트북은 아래에 위치한 예제 노트북의 수정된 버전으로 미리 학습된 머신 러닝 모델을 사용하도록 바뀌었습니다.

/sample-notebooks/introduction_to_amazon_algorithms/seq2seq_translation_en-de/SageMaker-Seq2Seq-Translation-English-German.ipynb

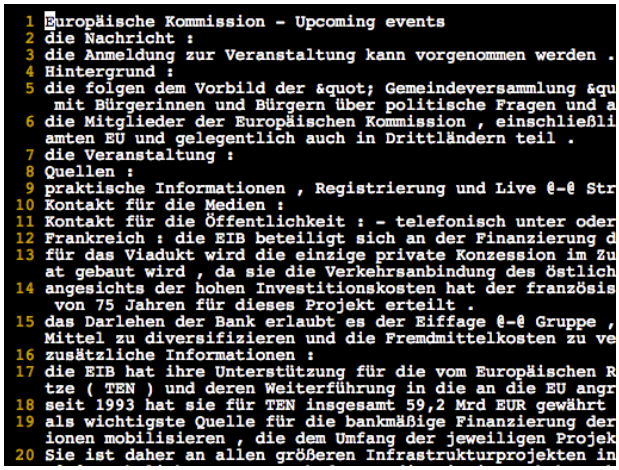
상기 노트북은 빠른 학습 시간을 위해 전체 데이터 중 첫번째 10000 개의 데이터의 대해서만 학습을 해서 Seq2Seq 알고리즘의 사용방법을 소개하고 있습니다.



다운받은 corpus 의 실제 데이터 내용으로 영어 및 독일어 데이터가 어떻게 문장 대 문장으로 매핑 되고 있는지를 보여주고 있습니다.



영문 데이터 (corpuc.tc.en.small 내용)

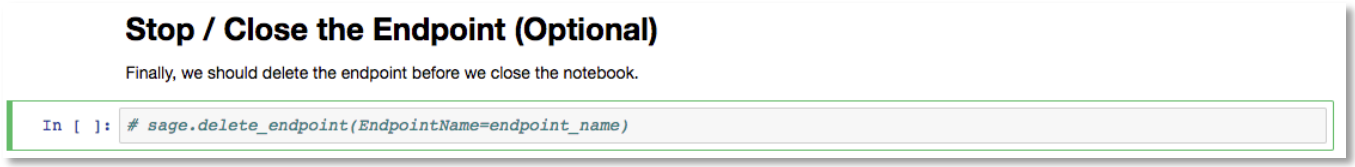


독일어 데이터(corpuc.tc.de.small 내용)

번역기 학습을 위한 영문 자료와 독일어 자료 비교 화면.

실제로는 10000 개의 샘플 문장으로 훈련한 번역기는 좋은 결과를 보여줄 수 없습니다. 그렇지만 전체 데이터 학습을 위해서는 선택하시는 SageMaker 의 서버 Instance Type 에 따라 다르지만 수시간에서 수일의 장시간이 소요될 수 있습니다. 따라서 이 노트북의 개발자들은 좀더 나은 품질의 번역 결과 체험을 원하시는 사용자들 위해 전체 데이터에 이미 훈련이 된 모델을 공유하고 있습니다.

이 Pre-trained model 을 사용하기 위해서는 노트북의 코드 중 Endpoint Configuration 직전의 코드를 아래와 같이 수정해서 이미 훈련된 모델을 다운로드 한 다음 본인의 S3 버켓으로 업로드 하시면 됩니다. 이때 Jupyter 노트북 마지막 줄의 *sage.delete_endpoint* 는 데모를 계속 진행하기 위해 실행하지 않습니다. 이를 위해 이번에는 가장 마지막 줄에 있는 코드를 주석 처리하겠습니다.



***delete_endpoint* 함수 콜 코멘트 처리 화면.**

Pre-trained 모델을 사용 하기 위한 노트북 수정

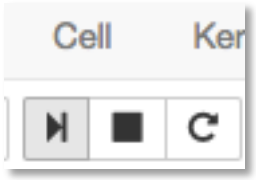
노트북에서 하단의 S3 bucket 이름에 상기 생성한 S3 이름을 입력하시고 우측의 예와 비슷한 형식으로 prefix 를 입력하시면 됩니다

```
# S3 bucket and prefix
bucket = '<your_s3_bucket_name_here>'
prefix = 'sagemaker/<your_s3_prefix_here>'  # E.g.'sagemaker/seq2seq/eng-german'
```

```
In [1]: # S3 bucket and prefix
        bucket = 'sagemaker-pilho-hands-on'
        prefix = 'sagemaker/seq2seq/eng-german'  # E.g.'sagemaker/seq2seq/eng-german'
```

노트북 실행 방법

이제 노트북 전체를 실행할 준비가 되었습니다. Jupyter 노트북을 실행하는 방법은 코드가 있는 셀을 클릭으로 선택하신 후 Shift-enter 키를 누르시거나 또는 Jupyter 노트북 상단의 툴바에서 "Run cell, select below" 버튼을 클릭하셔도 됩니다.



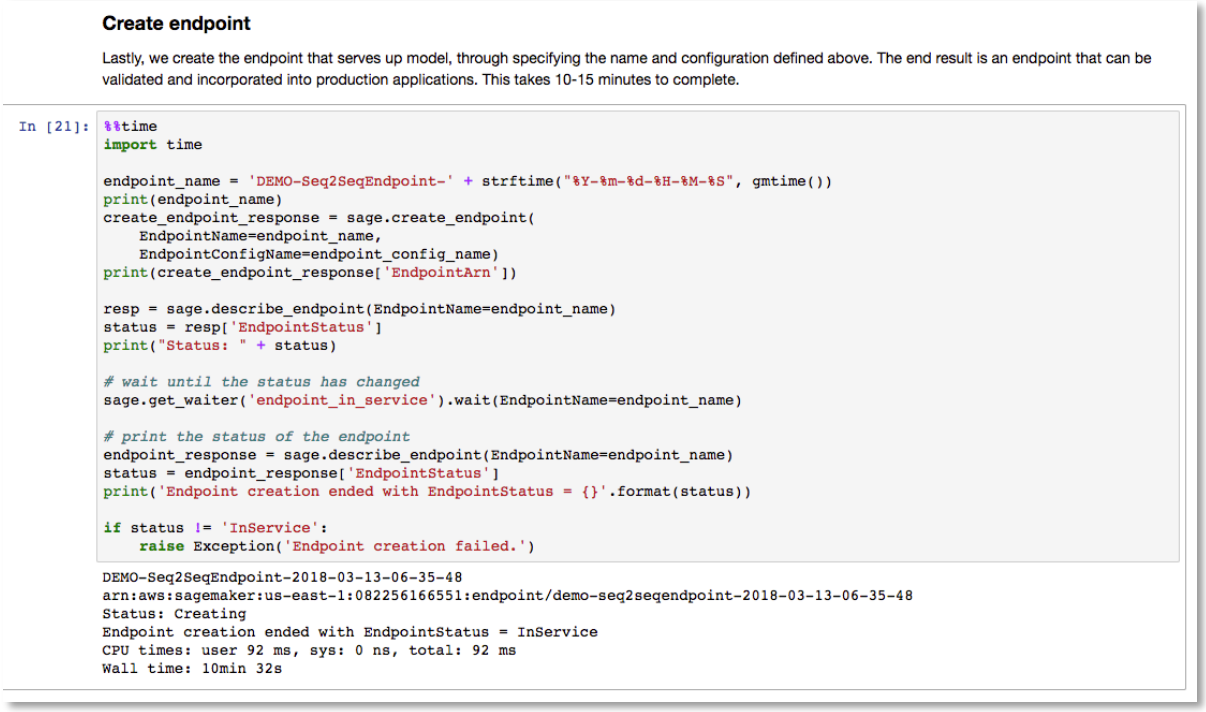
전체 실행 과정은 약 12 분에서 15 분 정도 소요 됩니다. 각각의 셀을 실행시키면서 셀 하단에 표시되는 처리결과들을 확인해 보시기 바랍니다.

노트북 코드 중 "Create endpoint configuration" 셀에서 현재 *InstanceType* 이 '*ml.m4.large*' 로 되어 있습니다. Seq2Seq 알고리즘은 Neural network 기반이기 때문에 *ml.p2.xlarge* (GPU) instance 를 사용하실 수 있지만 본 실습에서는 Free tier 가 지원되는 *ml.m4.xlarge* 를 사용하고 있습니다. ml.t2.* instance 는 time-out 문제가 발생할 수 있으므로 본 실습에서는 사용하지 않습니다.

```
In [12]: from time import gmtime, strftime

         endpoint_config_name = 'Seq2SeqEndpointConfig-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
         print(endpoint_config_name)
         create_endpoint_config_response = sage.create_endpoint_config(
             EndpointConfigName = endpoint_config_name,
             ProductionVariants=[{
                 'InstanceType':'ml.m4.xlarge',
                 'InitialInstanceCount':1,
                 'ModelName':model_name,
                 'VariantName':'AllTraffic'}])

         print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])

         Seq2SeqEndpointConfig-2018-03-24-08-35-50
         Endpoint Config Arn: arn:aws:sagemaker:us-east-1:082256166551:endpoint-config/seq2seqendpoint
         config-2018-03-24-08-35-50
```

노트북 코드 중 "Create endpoint" 셀은 새로운 서버를 설치하고 실행 코드를 설치하는 과정이므로 본 노트북에서는 가장 많은 시간 (약 10~11 여분)이 소요 되는데 아래와 같은 메세지를 확인하시면 다음 모듈로 진행하시면 됩니다.

Endpoint creation ended with EndpointStatus = InService



Create endpoint

Lastly, we create the endpoint that serves up model, through specifying the name and configuration defined above. The end result is an endpoint that can be validated and incorporated into production applications. This takes 10-15 minutes to complete.

In [21]:
```
%%time
import time

endpoint_name = 'DEMO-Seq2SeqEndpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = sage.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name)
print(create_endpoint_response['EndpointArn'])

resp = sage.describe_endpoint(EndpointName=endpoint_name)
status = resp['EndpointStatus']
print("Status: " + status)

# wait until the status has changed
sage.get_waiter('endpoint_in_service').wait(EndpointName=endpoint_name)

# print the status of the endpoint
endpoint_response = sage.describe_endpoint(EndpointName=endpoint_name)
status = endpoint_response['EndpointStatus']
print('Endpoint creation ended with EndpointStatus = {}'.format(status))

if status != 'InService':
    raise Exception('Endpoint creation failed.')
```

```
DEMO-Seq2SeqEndpoint-2018-03-13-06-35-48
arn:aws:sagemaker:us-east-1:082256166551:endpoint/demo-seq2seqendpoint-2018-03-13-06-35-48
Status: Creating
Endpoint creation ended with EndpointStatus = InService
CPU times: user 92 ms, sys: 0 ns, total: 92 ms
Wall time: 10min 32s
```

SageMaker Endpoint 생성 결과 화면.

노트북 가장 하단의 *delete_endpoint* 는 주석 처리 되어 있어야 endpoint 서버가 다음 실습을 위해 계속 운용될 수 있습니다. 만약에 실행 전에 수정하셨다면 "Create endpoint" 부분의 코드를 다시 실행하시기 바랍니다.

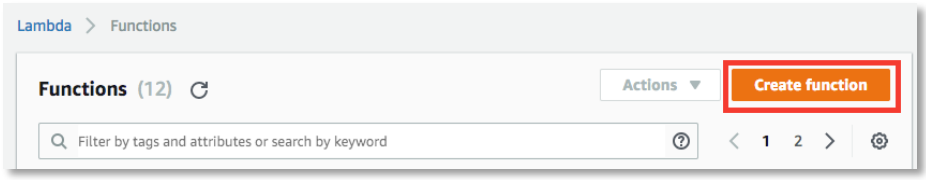
Module 6-2: SageMaker Endpoint 호출 Lambda 함수 개발하기

본 모듈에서는 방금 생성한 SageMaker 의 Inference service 를 호출하는 Lambda 함수를 개발해 보겠습니다.

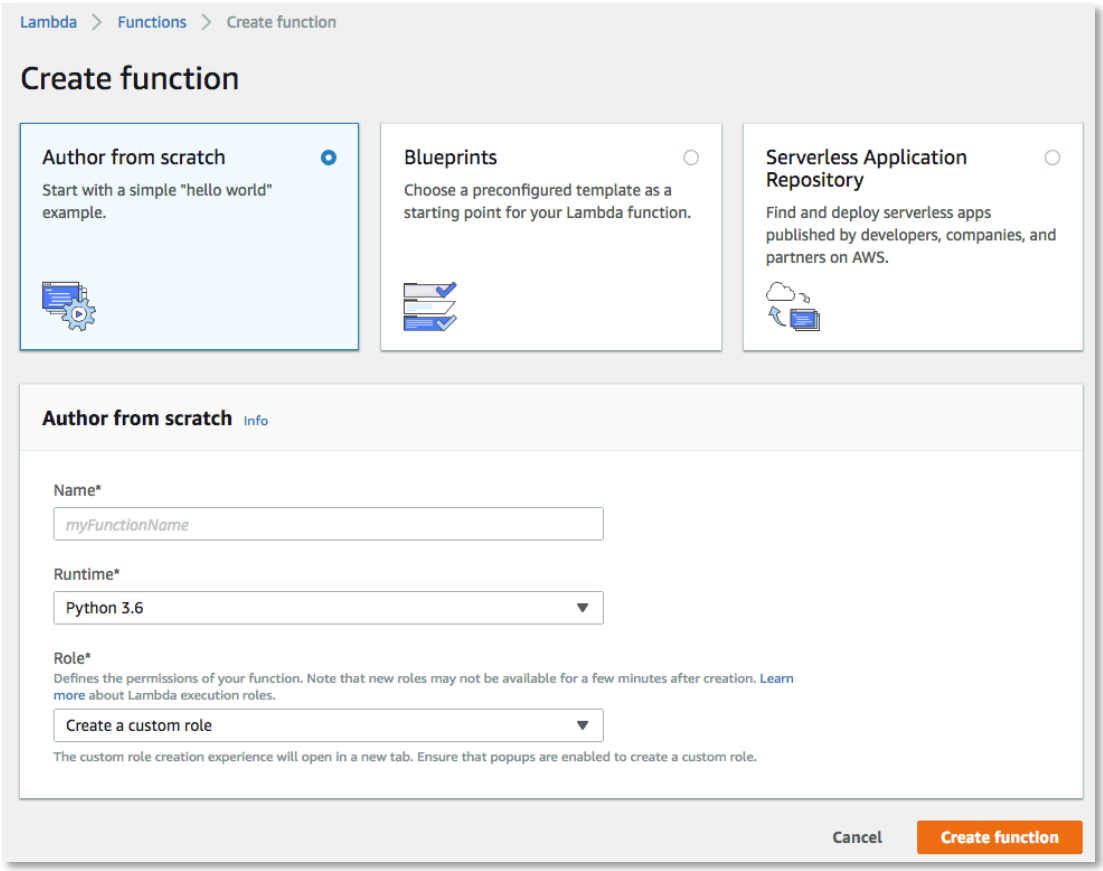
Lambda 함수 생성하기

1. AWS 콘솔에서 Lambda 를 선택 (https://console.aws.amazon.com/lambda)

2. "Create function" 선택



Lambda 함수 생성 화면.



Lambda 함수 생성 화면.

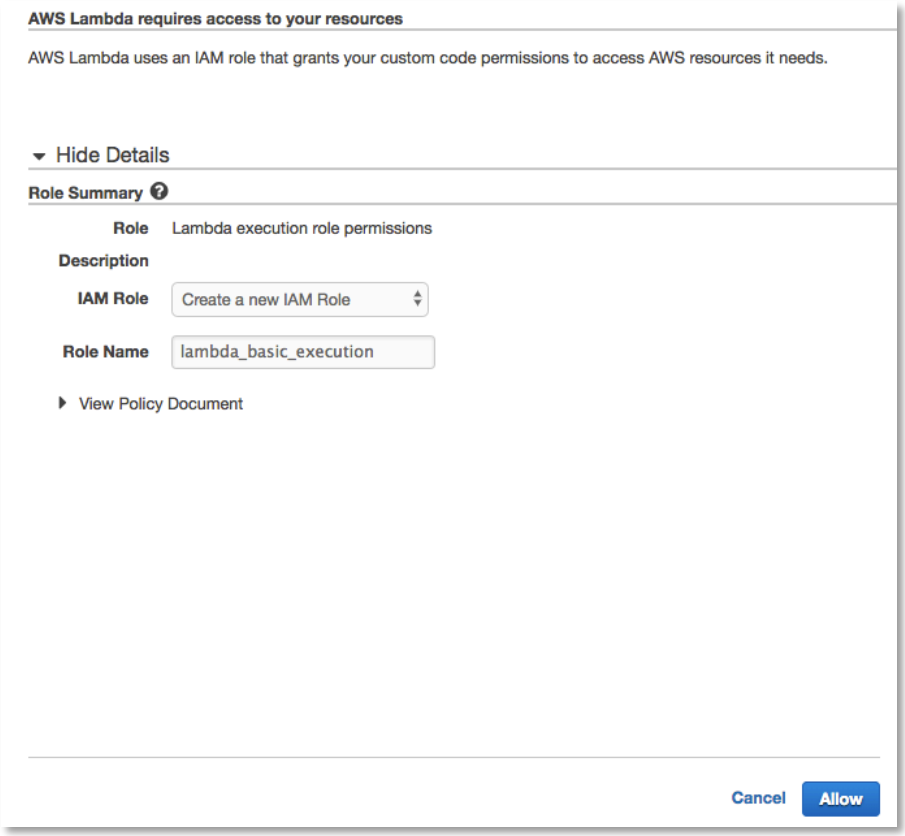
3. Lambda 생성화면에서 Lambda 함수 이름과 Runtime (Python 3.6) 그리고 Role 은 "Create a custom role"을 선택합니다.

a. Name : MySeq2SeqInference 으로 지정.

b. Runtime: Python 3.6 으로 지정

c. Role: Create a custom role 을 선택하면 [AWS Lambda required access to your resources]가 나옵니다. 여기서 [Allow]를 누릅니다.

d. Allow 클릭하면 창이 닫히고 Lambda Console 로 돌아가는 데 여기서 Create Function 을 선택하시면 됩니다.

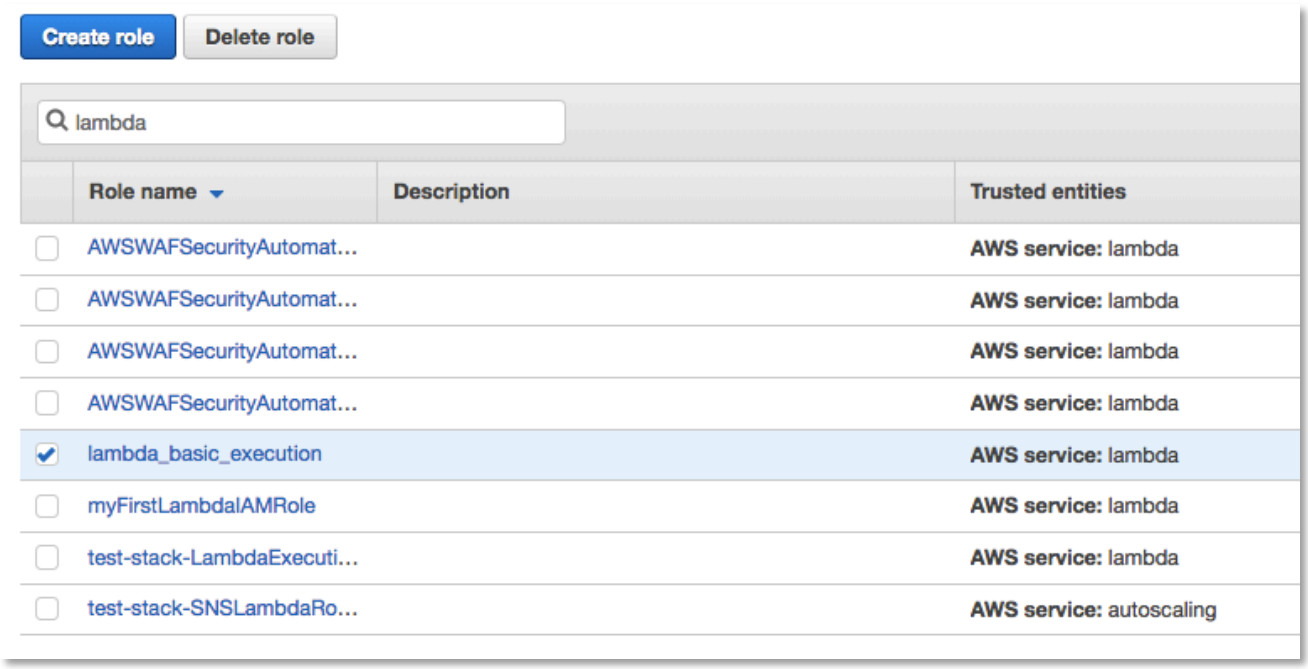


AWS Lambda 접근 허락 화면.

Lambda 함수에 Role 을 추가하기

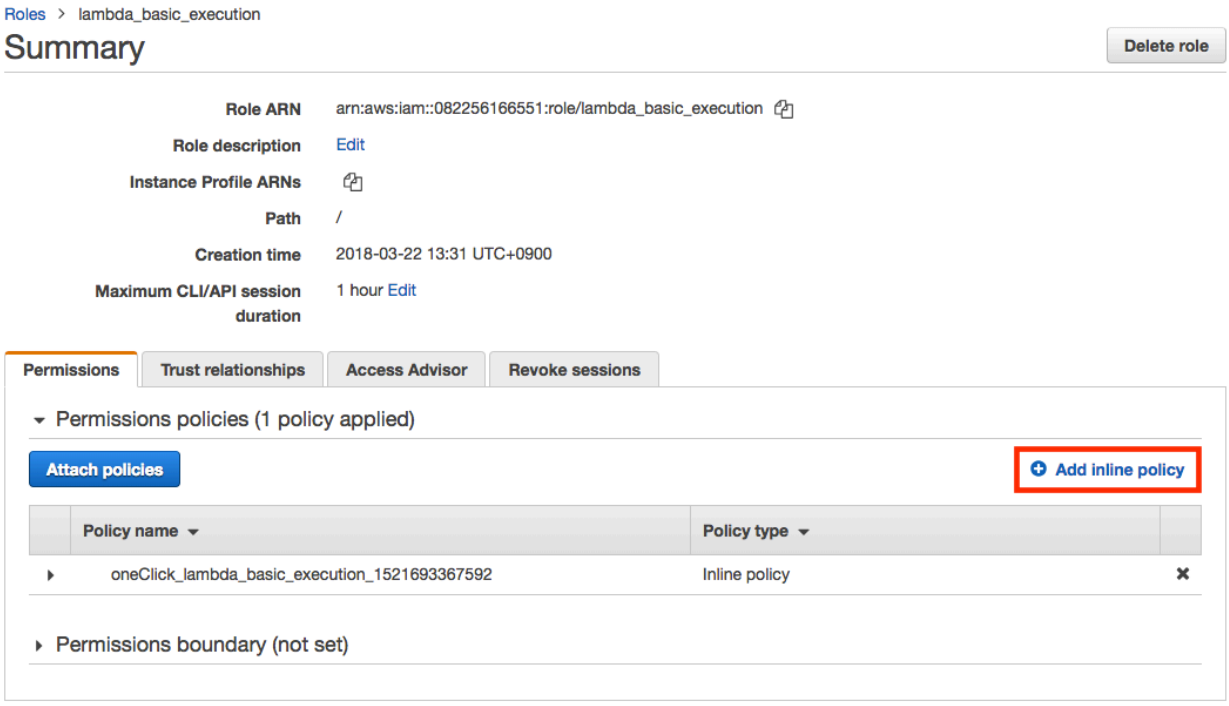
방금 생성한 Lambda 함수에 새롭게 추가된 Role 에 SageMaker 와 API Gateway 를 사용할 수 있는 정책(Policy)를 추가해보겠습니다.

1. AWS 콘솔에서 IAM 서비스를 선택하세요.
2. 왼편의 메뉴에서 "Roles"를 클릭하세요.
3. 방금 생성하신 Lambda 에 사용되는 Role 을 선택하세요



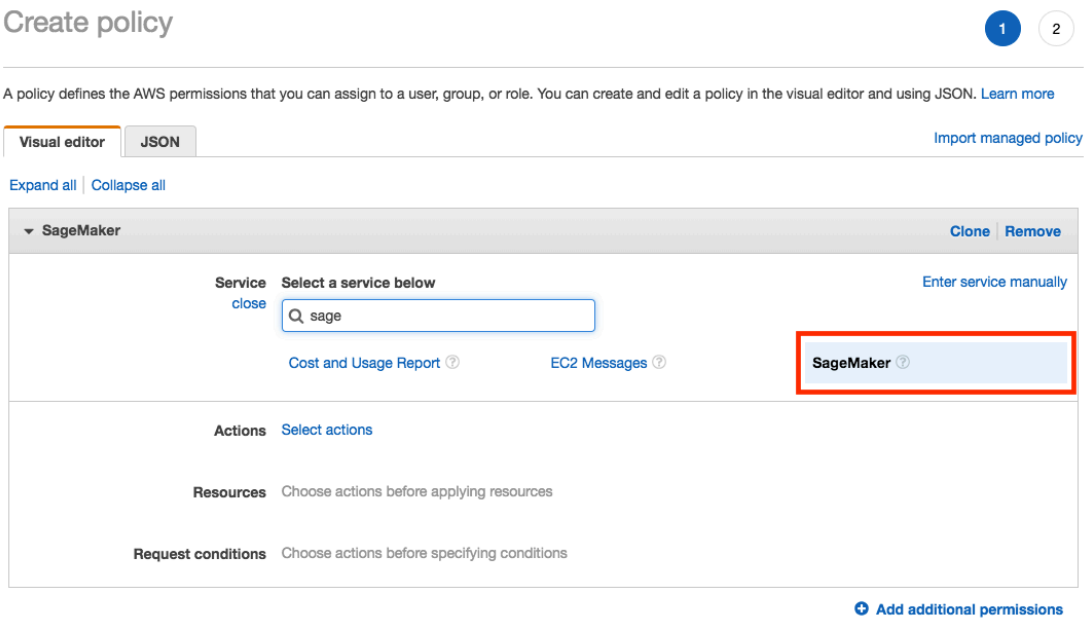
Lambda 함수 선택.

4. "Add inline policy"를 선택합니다.



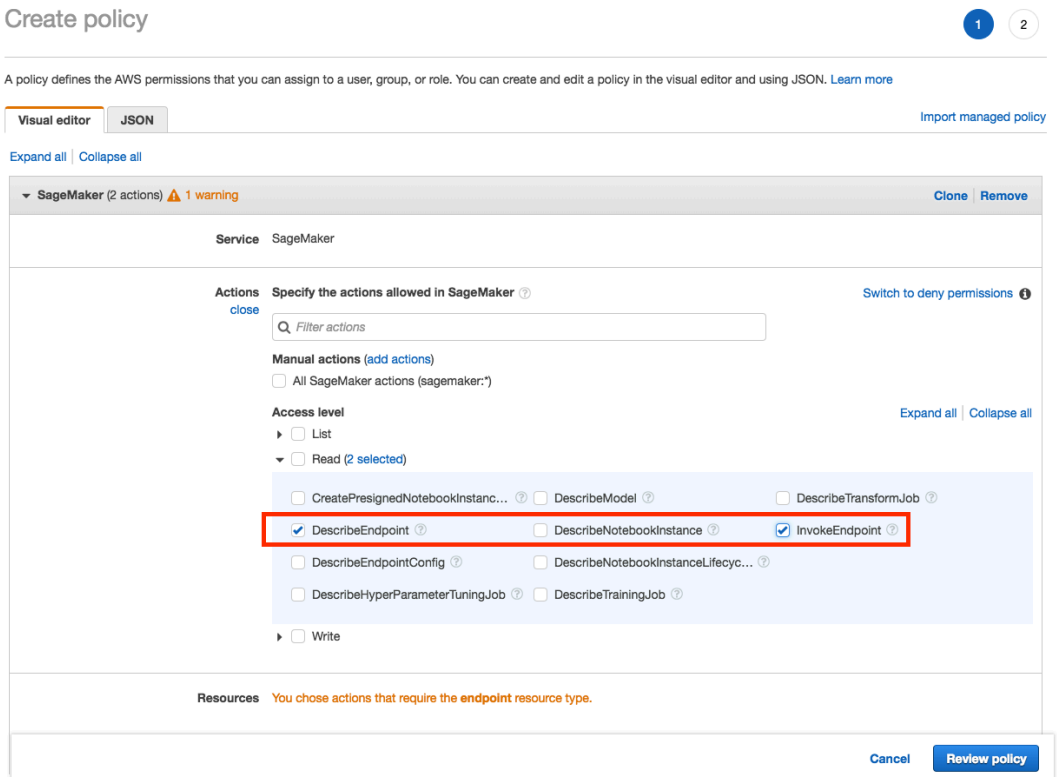
IAM Role 에 정책을 추가하는 화면.

5. 다음 화면의 검색창에 "SageMaker" 입력 합니다.



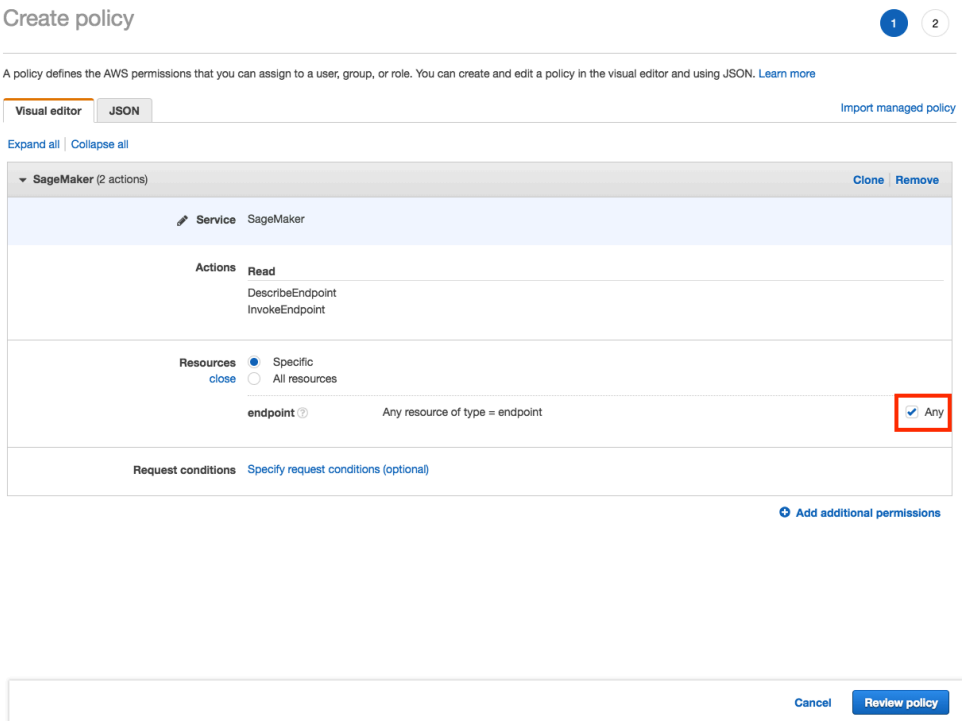
AmazonSageMakerFullAccess 정책 추가 화면.

6. Access level at Actions 에 있는 모든 "DescribeEndpoint" and "**InvokeEndpoint**" 를 선택합니다.



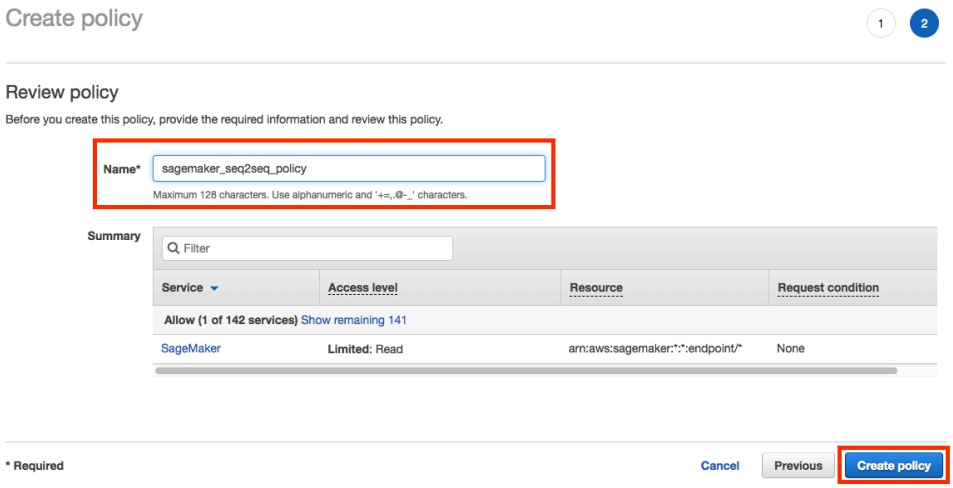
Select DescribeEndpoint and InvokeEndpoint in the Access level.

7. 하면 하단의 Resources 에 있는 노란색의 "You chose actions that require the endpoint-config resource type" 문장을 선택하신 후 Resources 섹션에 있는 "Any" 를 선택합니다. 이후 화면 하단에 있는 "Review policy"를 선택합니다.



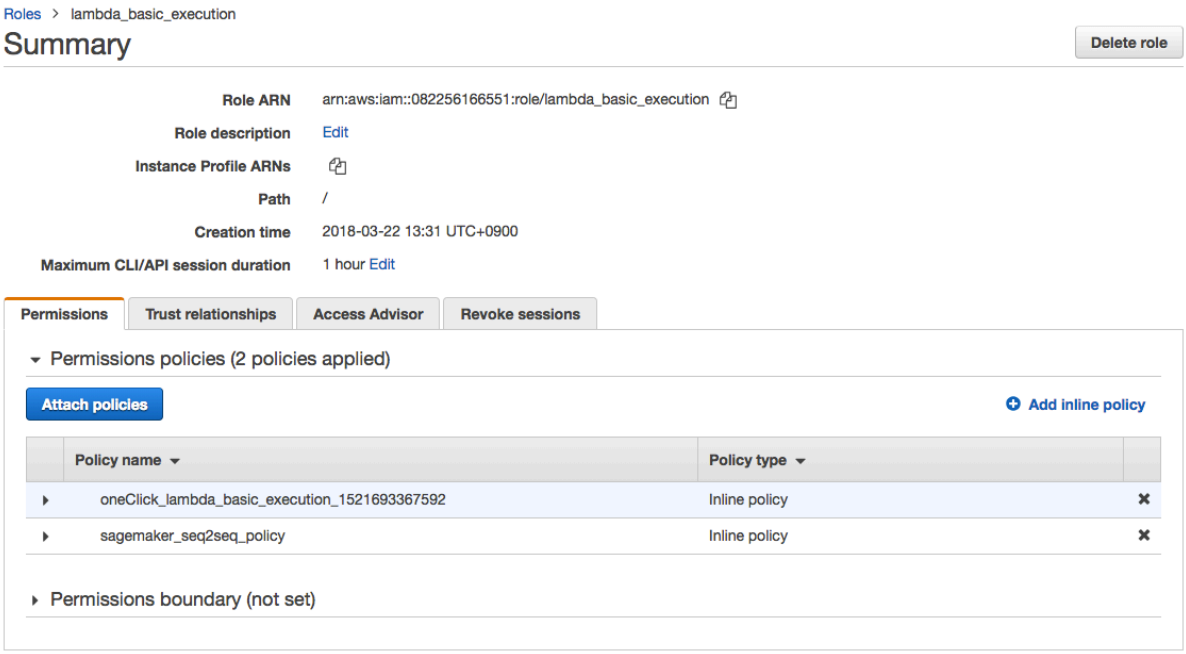
Select endpoint resource type.

8. "Review policy" 다이얼로그에서 새로운 policy 이름을 입력하신 후 화면 하단의 Create policy 버튼을 선택합니다.



Create policy screen.

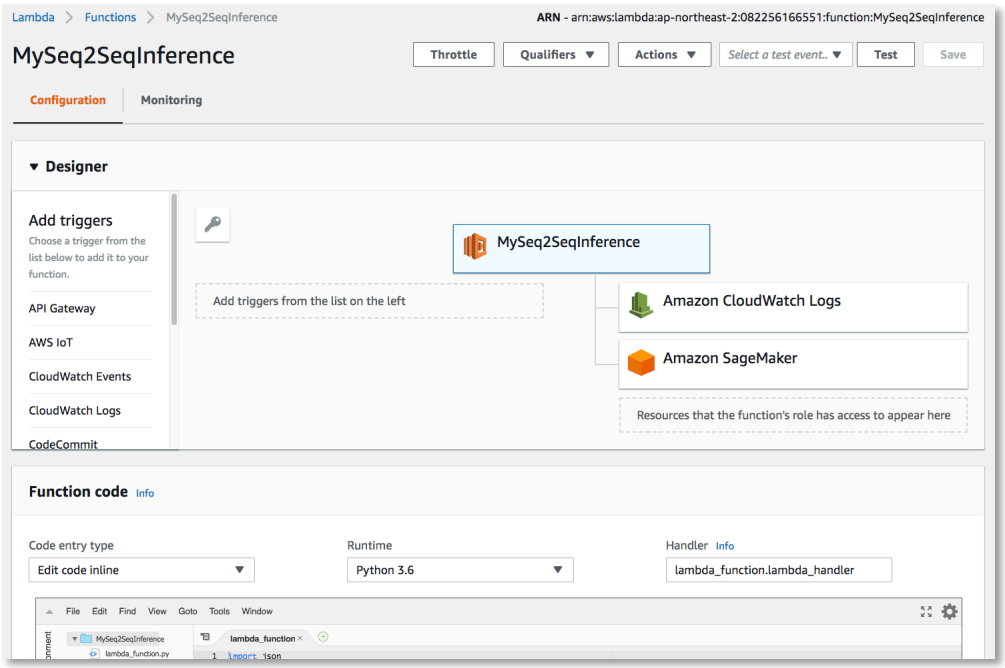
9. 최종 추가된 Policy 확인



최종 Role 의 정책들 화면.

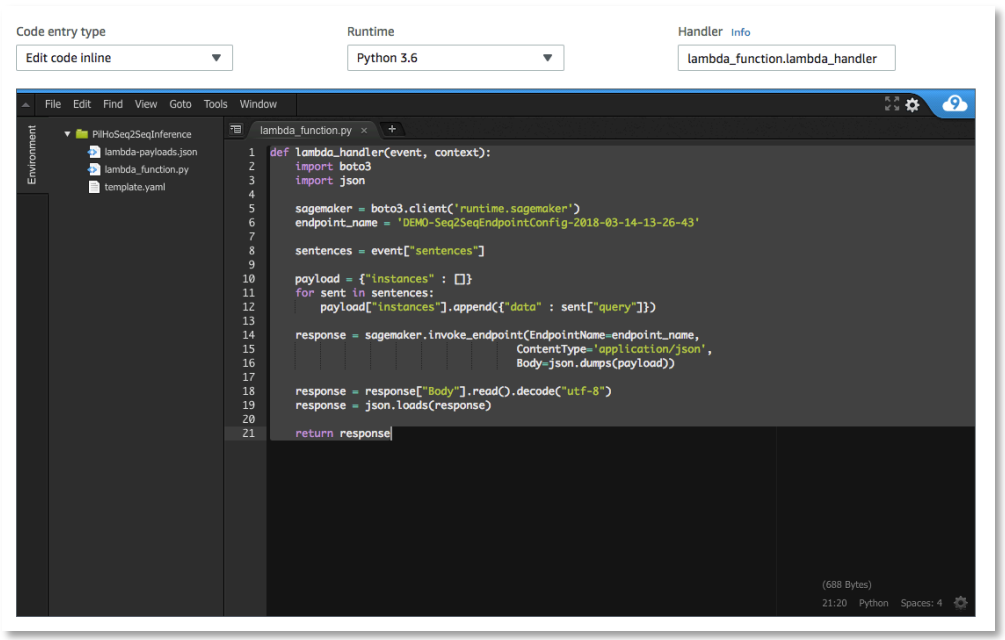
Lambda 함수 코딩하기

다시 AWS 콘솔의 Lambda 서비스 화면으로 이동하신 후 윗 단계에서 생성하신 Lambda 를 선택합니다. 아래와 같이 추가된 Role 의 Policy 들을 확인하실 수 있습니다.



Lambda 선택 화면.

현 페이지에서 마우스를 스크롤해서 하단으로 이동하면 Lambda 의 내장 코드들을 직접 수정할 수 있는 인터페이스가 제공이 됩니다.



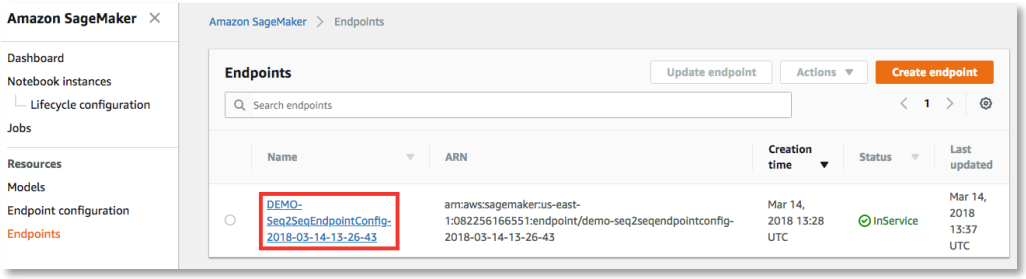
Lamba 코드 개발 화면.

AWS Lambda 는 AWS 콘솔 상에서 바로 코딩할 수 있게 Cloud9 에디터가 내장되어 있습니다. 아래의 순서에 따라 Lambda 함수를 만들어 보겠습니다.

1. 다음 페이지의 Python 샘플 코드를 Copy 후 Paste 로 Lambda 의 online editor 에 입력합니다. Python 코드를 복사 및 붙여 넣기를 할때는 원 코드의 indent 를 그대로 지키는 것이 중요합니다. 현재 보시고 있는 PDF 문서 상에서 복사가 제대로 되지 않는 경우 아래 온라인 주소에서 소스코드를 복사하 셔도 됩니다:

https://raw.githubusercontent.com/pilhokim/ai-ml-workshop/master/2018-09/lambda_function.py

2. 붙여넣기 하신 소스코드 상의 "endpoint_name" 을 본 실습 동안 생성한 Seq2Seq endpoint 서버 주소로 변경합니다.



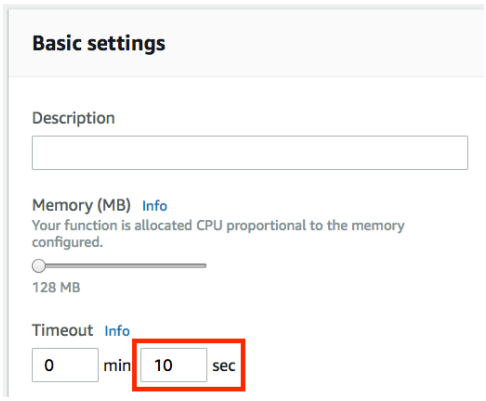
SageMaker EndPoint 이름 확인 방법.

Labmda Python sample Code

```
def lambda_handler(event, context):
    import boto3
    import json

    sagemaker = boto3.client('runtime.sagemaker')
    endpoint_name = 'YourSeq2SeqEndpointName'

    sentences = event["sentences"]

    payload = {"instances" : []}
    for sent in sentences:
        payload["instances"].append({"data" : sent["query"]})

    response = sagemaker.invoke_endpoint(EndpointName=endpoint_name,
                                         ContentType='application/json',
                                         Body=json.dumps(payload))

    response = response["Body"].read().decode("utf-8")
    response = json.loads(response)

    return response
```

3. Endpoint 용으로 선택하신 서버의 Instance Type 과 번역을 하기위한 text 의 크기에 따라 번역에 몇초 이상이 소요될 수도 있습니다. 이 시간동안 Lambda 함수 호출이 Timeout 되는 것을 방지하기 위해 Lambda 의 Timeout 시간을 10 초로 늘입니다.

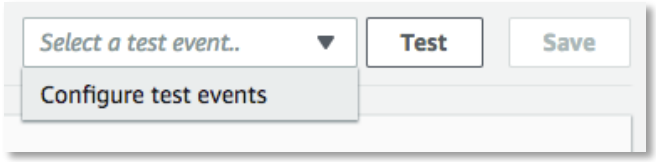
4. 상단의 "Save" 버튼을 눌러 저장합니다.



Lambda 함수 Timeout 값 조정.

새로 만든 Lambda 함수의 동작을 바로 확인할 수 있습니다.

1. 아래와 같이 "Configure test events"를 선택합니다.

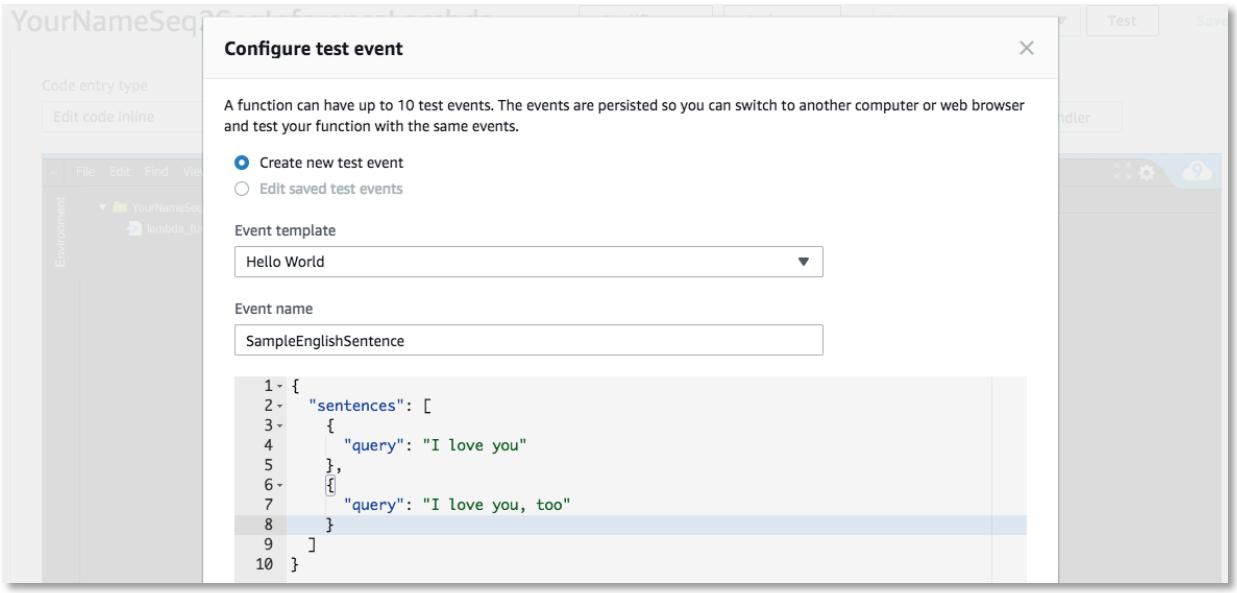


Lambda 테스트 데이터 구성 화면.

2. Event name 을 입력합니다 (예: SampleEnglishSentence)

3. 하단의 테스트 이벤트 입력화면에서 아래의 샘플 영어 문장을 입력합니다. 또는 https://raw.githubusercontent.com/pilhokim/ai-ml-workshop/master/2018-09/sample_query.json 에서 복사해서 사용하셔도 됩니다.

```
{
 "sentences": [
  {
   "query": "I love you"
  },
  {
   "query": "I love you, too"
  }
 ]
}
```

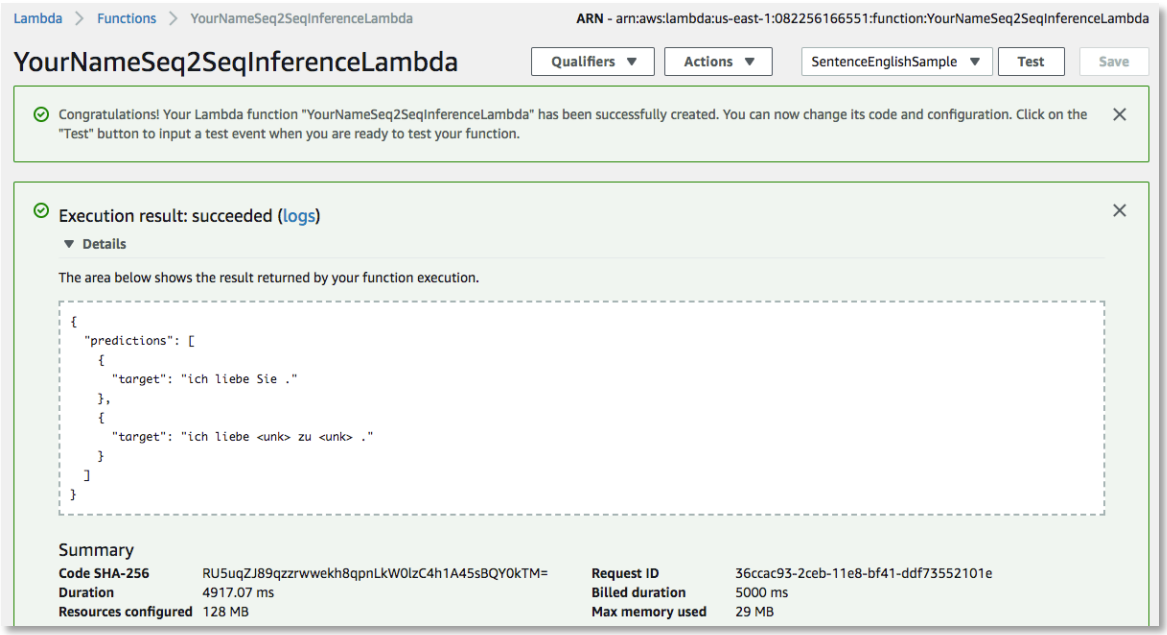


Test 이벤트 생성.

이때 주의하실점은 JSON 형식의 "sentences"와 "query"는 미리 약속된 key 값이므로 변경을 하시면 안됩니다.

4. Create 버튼을 선택합니다.

5. 입력이 완료 된 후 상단의 "Test" 버튼을 클릭하시면 아래와 같은 화면이 보이면 정상적으로 작동하는 것을 확인하실 수 있습니다. 하단의 Cloud9 에서도 결과를 확인하실 수 있습니다.

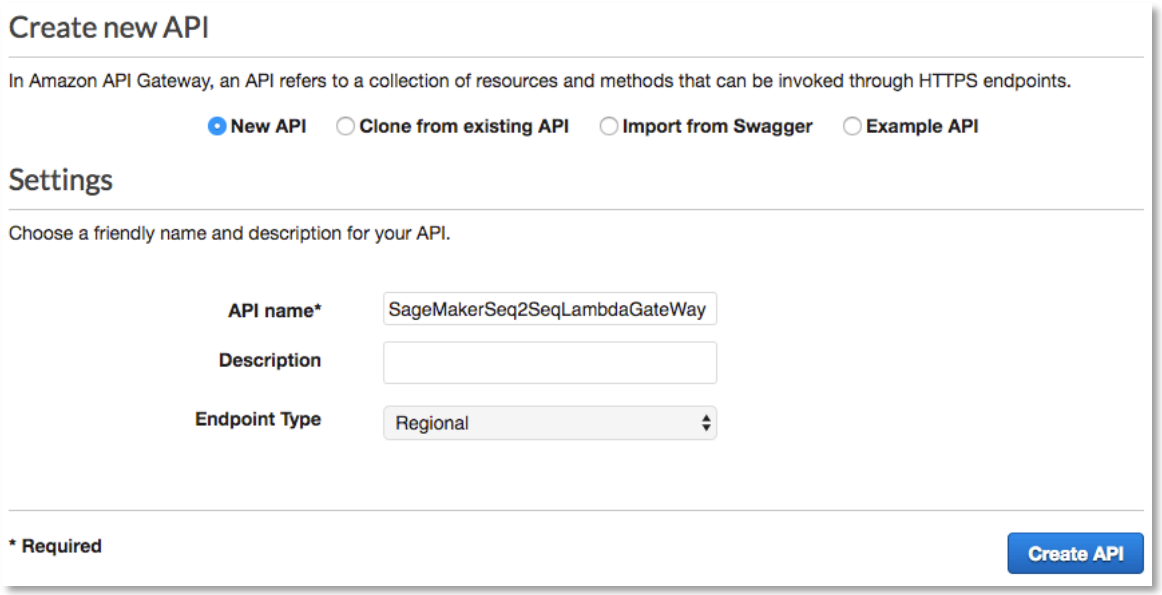


Lambda 함수 테스트 결과 화면.

Module 6-3: AWS API Gateway 와 S3 Static Web Server 를 이용한 웹서비스 연결하기

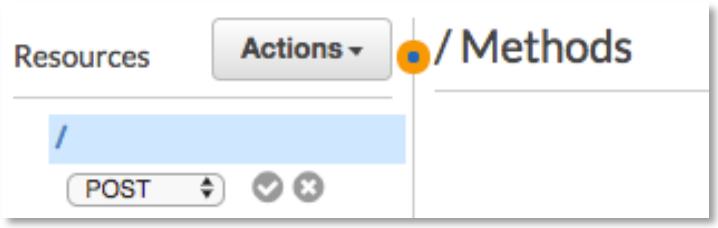
API Gateway 생성 및 Lambda 함수 연결하기

1. Amazon API Gateway 콘솔 접속 (https://console.aws.amazon.com/apigateway/)
2. "Create API" -> "New API" 선택
3. 셋팅에서 새로운 API name 입력 (ex. **SageMakerSeq2SeqLambdaGateWay**)후 Endpoint Type 을 Regional 로 선택



Amazon API Gateway 생성 화면.

4. 바뀐 화면에서 Actions -> Create Method 선택
5. 하단의 콤보 박스에서 POST 선택
6. 체크(V) 버튼 클릭해서 적용



POST method 추가 화면.

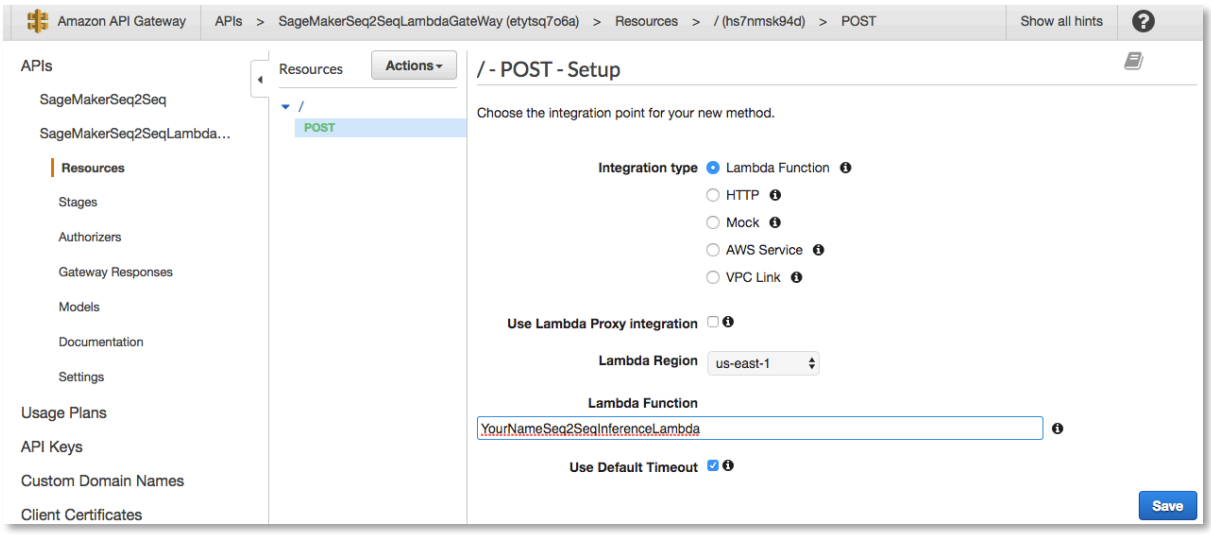
7. 오른편의 셋업에서 아래와 같이 입력 진행

   a. Integration type: Lambda function

   b. Lambda region: Labmda 를 생성하신 Region (us-east-1) 입력

   c. Lambda function: Lambda 함수 이름 입력

   d. "Save" 선택



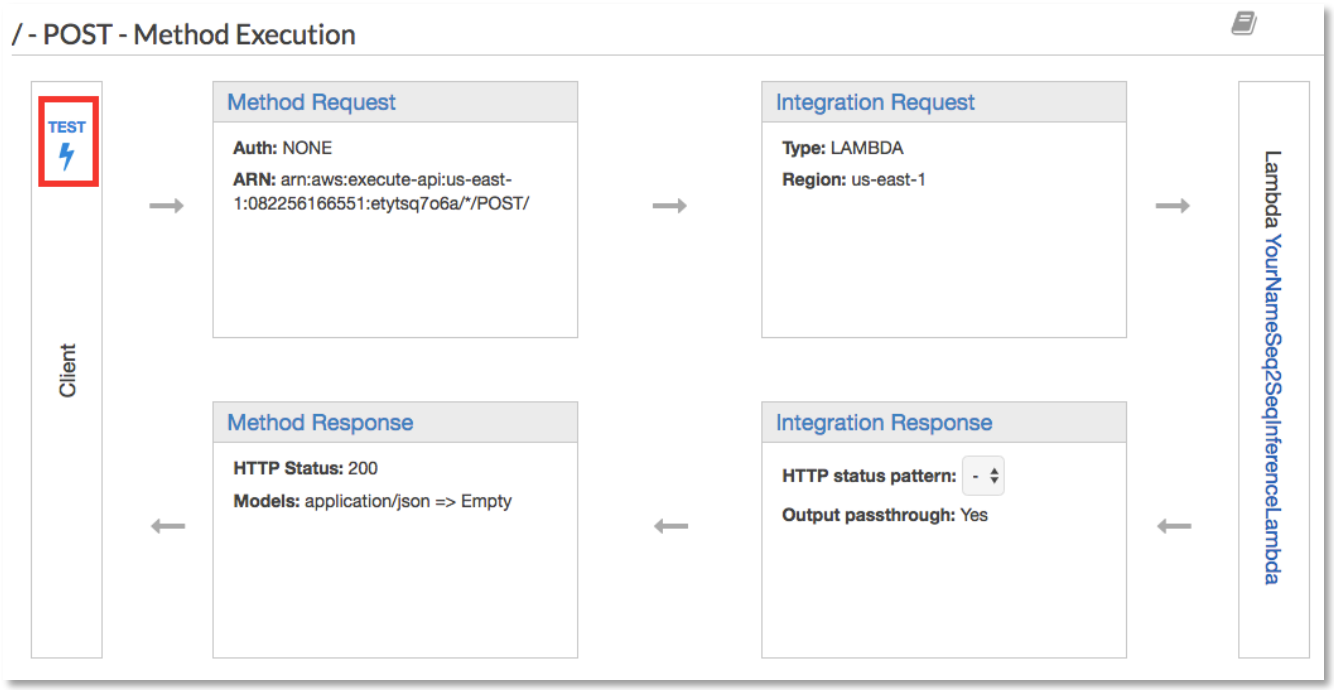
Lambda 함수를 호출하기 위한 Gateway POST method 셋팅 화면.

API Gateway 가 생성이 된 이후 Test 를 진행하여 제대로 Lambda 를 호출하는지 확인하실 수 있습니다.

a. "Test"를 선택하셔서 API Gateway 의 testing interface 를 확인합니다.

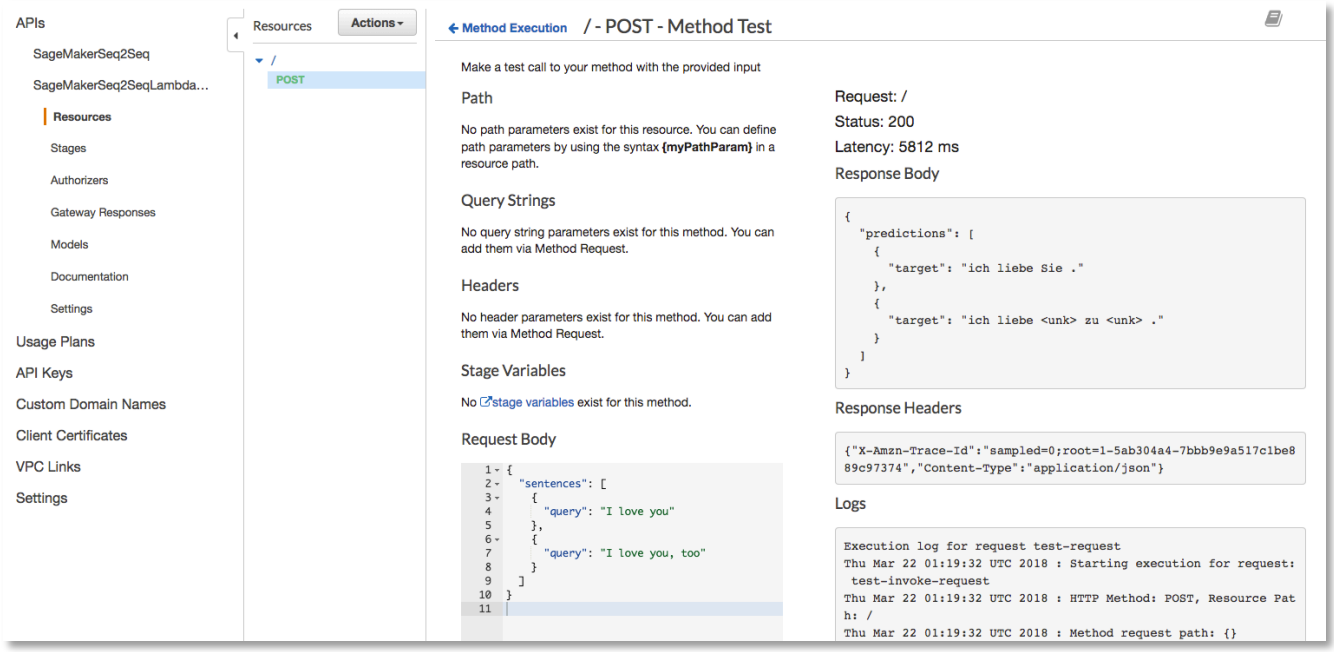
b. Request body 에 Lambda 호출에 사용되었던 아래의 예제 데이터를 입력하신 후 Test 를 선택합니다.

```
{
  "sentences": [
    {
      "query": "I love you"
    },
    {
      "query": "I love you, too"
    }
  ]
}
```



API Gateway Test 화면

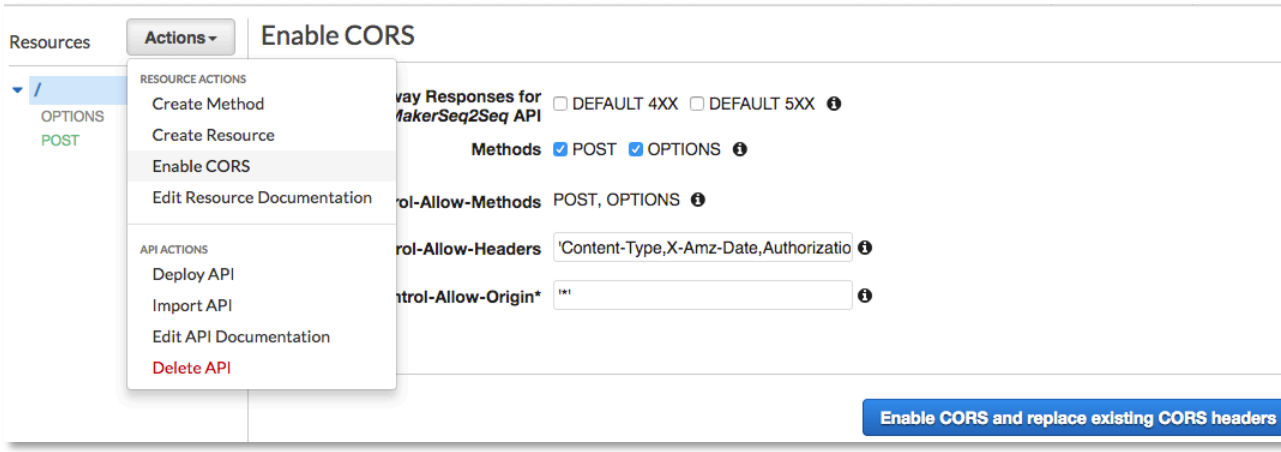
테스트 결과가 아래와 같이 보이면 정상적으로 동작하는 것으로 확인하실 수 있습니다.



API Gateway 테스트 결과.

8. Enable CORS: S3 Static Web Server 를 이용해서 API Gateway 를 호출하면 origin 이 다르기 때문에 반드시 CORS (Cross-Origin Resource Sharing)를 Enable 해야만 외부 싸이트에서 이 REST 서비스를 이용할 수 있게 됩니다.

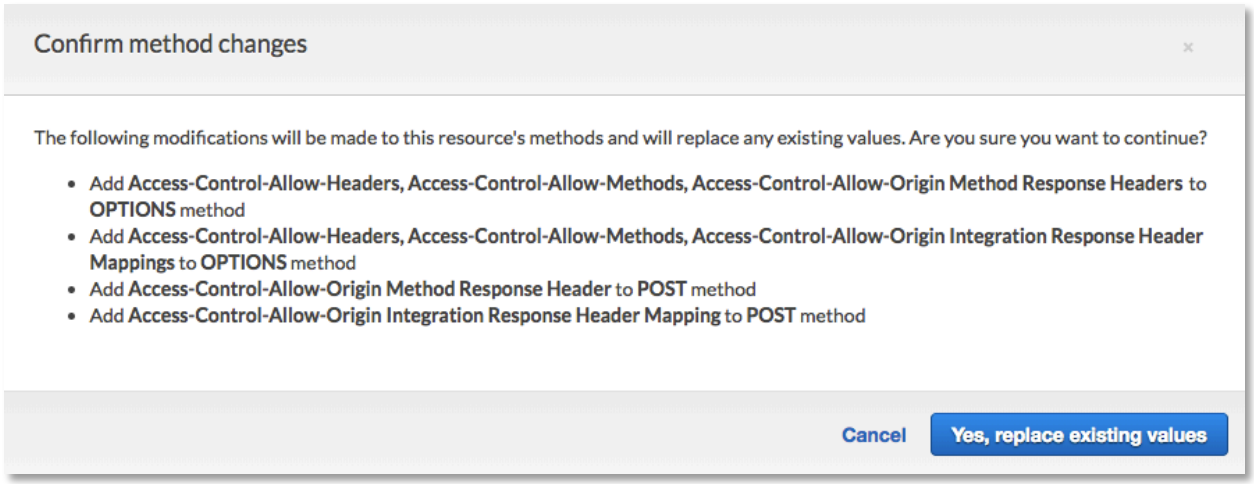
   a. Actions -> Enable CORS 선택



API Gateway API Enable CORS 화면.

   b. "Enable CORS and replace existing CORS headers" 선택

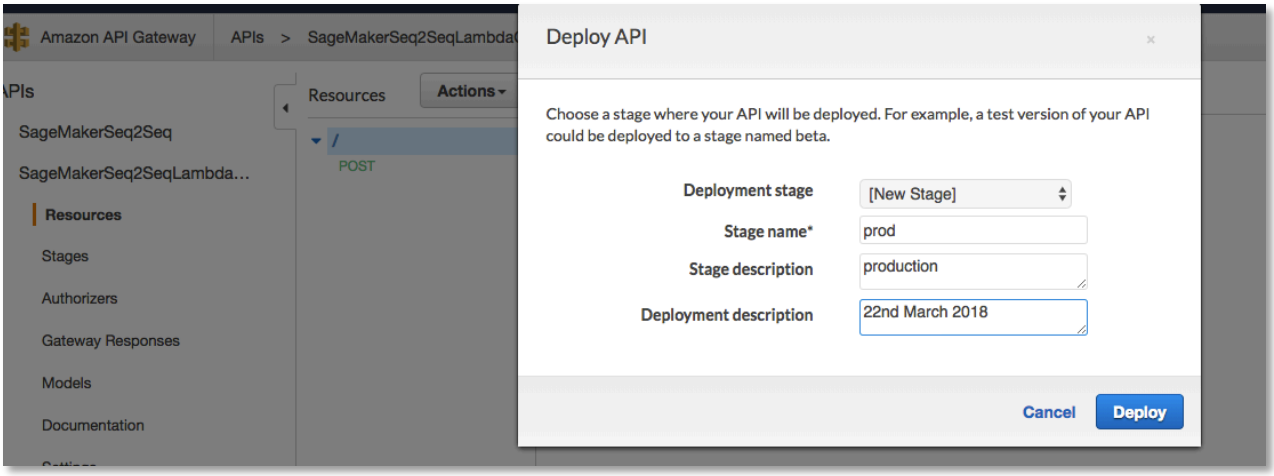
   c. "Yes, replace existing values" 선택



CORS replace existing values 화면.

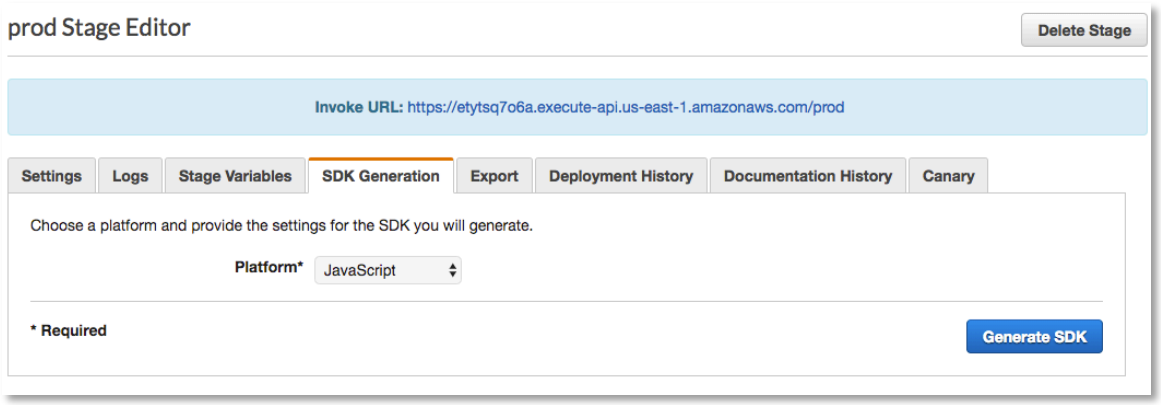
9. 정상적으로 동작이 되면 Actions->Deploy API 선택합니다. API Deploy 를 반드시 하셔야 실제 외부 (Public Internet)에서 호출을 할 수 있습니다.

10. 현재 생성한 Gateway 의 stage 이름을 부여합니다. 예제에서는 "prod"라는 약어로 stage 이름을 정의하였습니다. 개발 단계에 따라 "test" 나 "prod" 등 의미 있는 키워드를 부여하시면 됩니다.



API deploy 화면.

11. Deploy 가 된 이후 Stage Editor 에서 invoke URL 을 메모장에 따로 기록해 두시고 "SDK Generation" -> Platform (JavaScript) -> Generate SDK 선택. 이 JavaScript 라이브러리는 API Gateway 서비스에 대해 CORS (Cross-Origin Resource Sharing)을 지원해주는 기능을 포함하고 있습니다.



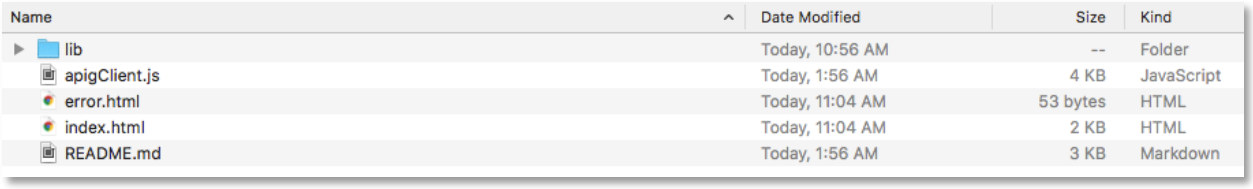
API Gateway 접속 SDK 다운로드 화면

이제 S3 를 이용해서 static web server 를 설정하기 위한 화일들을 준비하겠습니다.

a. 상기 API Gateway SDK 생성으로 다운 받은 압축 화일을 임의의 디렉토리에 푸세요 (unzip).

b. S3 Static 웹 서버에 사용될 index.html 과 error.html 파일을 다음의 S3 버켓에서 다운로드 하여 상기 단계에서 사용된 디렉토리에 동일하게 저장합니다: https://s3.amazonaws.com/pilho-sagemaker-ai-workshop-lambda/index_error_html.zip

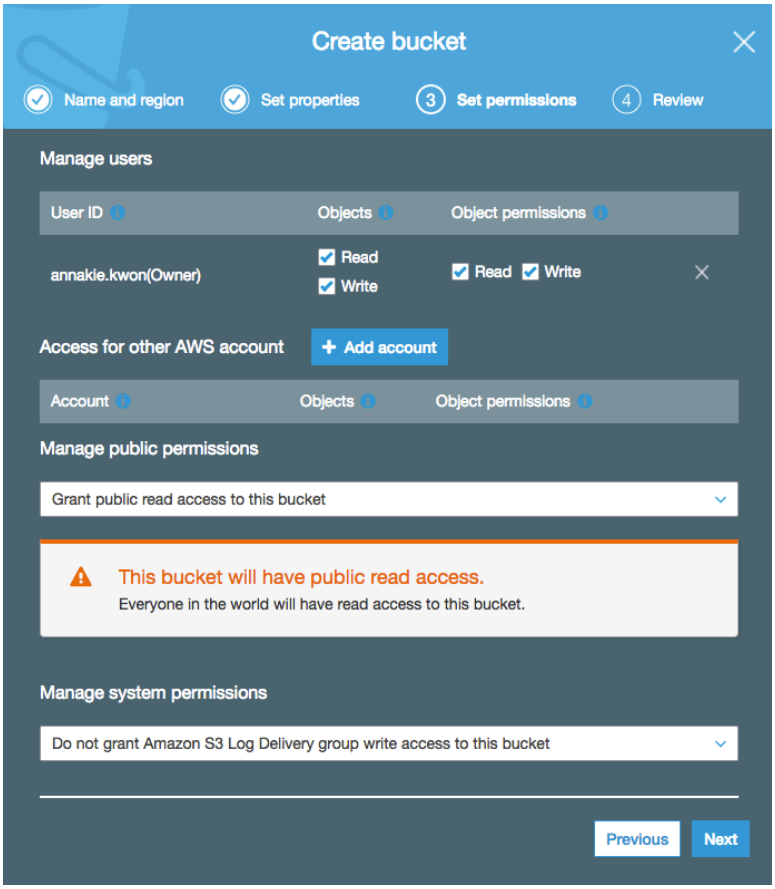
c. 최종 파일들이 아래와 같이 구성되어 있으면 됩니다. 이 파일들은 다음 단계에서 만들 S3 버킷에 업로드 되게 됩니다.



웹서버 구성 화일 리스트 화면.

S3 Static Web Server 생성하기

1. Amazon S3 콘솔 접속 (https://s3.console.aws.amazon.com)
2. "Create bucket" 선택
3. 새로운 버킷 이름 입력 (ex. "jihye-sagemaker-public-test") -> Next -> Next 선택
4. Set permissions 에서 Manage public permissons 를 "Grant public read access to this bucket" 으로 설정



S3 버킷 Public 접속 허용 화면.

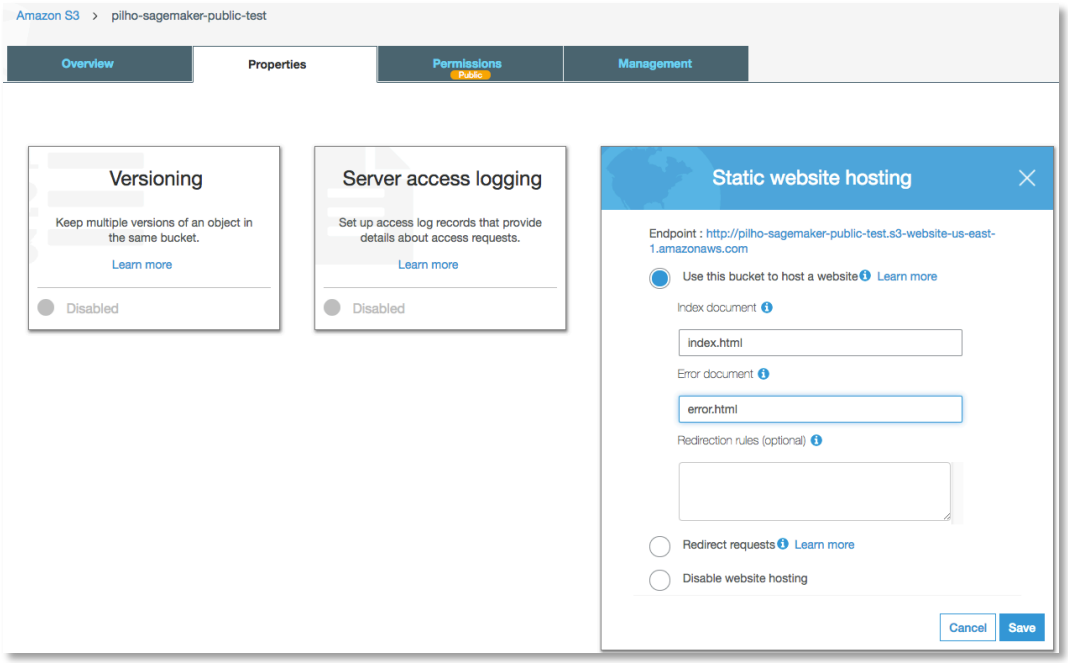
5. Next->Create bucket 선택

6. 생성된 S3 bucket 선택

7. "Properties" -> "Static website hosting" -> "Use this bucket to host a website" 선택 후 Index document : index.html, Error document : error.html 입력

8. "Save" 선택

9. 이 단계 까지 마치신 후 상단의 URL 형식의 Endpoint 주소를 기록해 둡니다. 이 URL 주소를 이용해서 S3 웹 서버에 접속하게 됩니다.

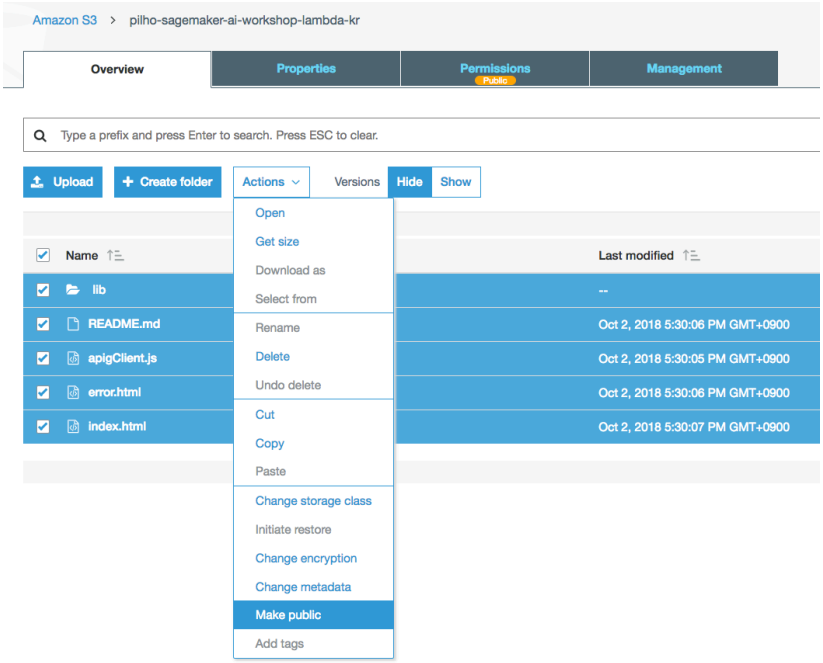


S3 static 웹서버 설정 화면.

10. "Overview" 탭 선택 -> "Upload" 선택

11. 생성된 S3 Bucket 에 이전 단계에서 생성된 파일들을 Drag & Drop 으로 업로드 합니다.

12. 이때 Set permissions 을 아래와 같이 반드시 Grant public read access to this object(s)로 설정해야 합니다.



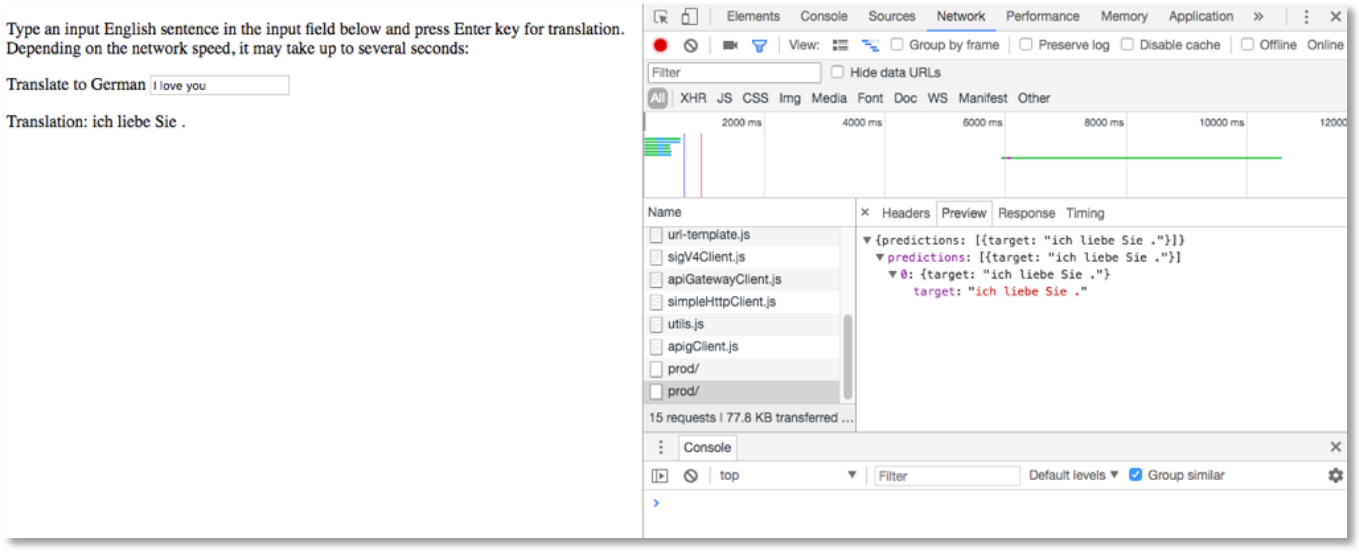
S3 파일들에 대한 Make public 설정 화면.

최종 서비스 테스트하기

1. 웹브라우즈를 구동하시고 S3 Endpoint URL 에 접속합니다.

2. Translate to German 오른편의 텍스트 입력 창에 영문 문장을 입력합니다. (Ex. "I love you")

3. 몇 초 정도 기다리시면 하단에 번역 결과가 보여집니다.



웹기반 번역 서비스 테스트 화면.

SageMaker Endpoint 서버 자동 확장 설정하기

본 섹션은 향후 실제 필요시에 대한 참조용으로 제공됩니다. 실제 Hands-on 을 하실 필요는 없습니다.

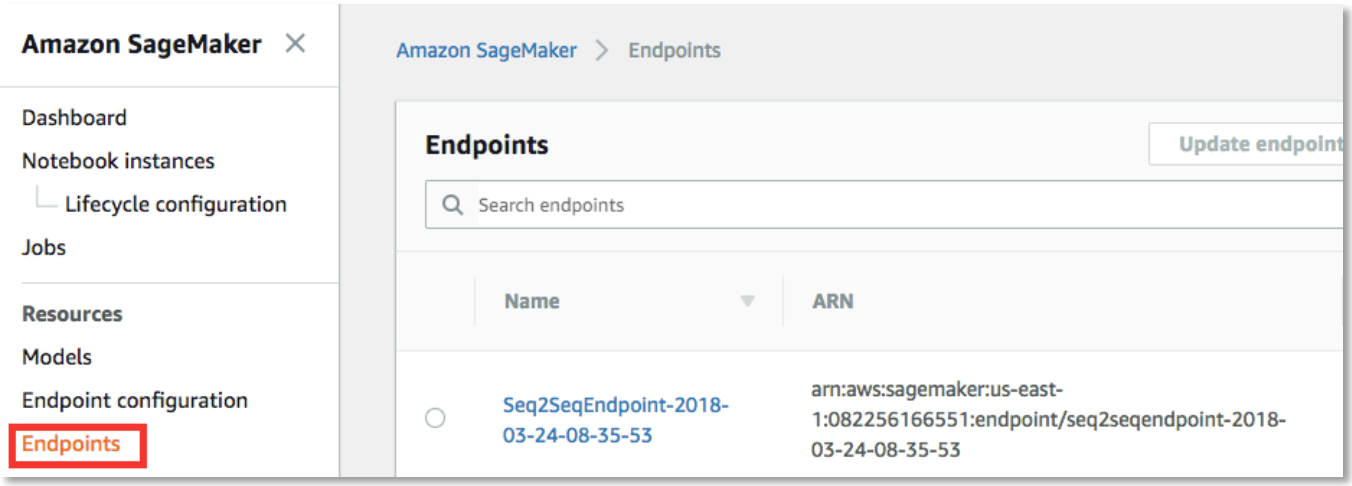
웹 기반 서비스를 제공하기 시작하고 사용자 수가 증가하기 시작하면 SageMaker 의 Inference 서버도 자동으로 확장되게 설정하실 수 있습니다.

```
In [12]: from time import gmtime, strftime

endpoint_config_name = 'Seq2SeqEndpointConfig-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = sage.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType':'ml.m4.xlarge',
        'InitialInstanceCount':1,
        'ModelName':model_name,
        'VariantName':'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])

Seq2SeqEndpointConfig-2018-03-24-08-35-50
Endpoint Config Arn: arn:aws:sagemaker:us-east-1:082256166551:endpoint-config/seq2seqendpoint
config-2018-03-24-08-35-50
```

Endpoint 설정에서 InitialInstanceCount 변수 화면.

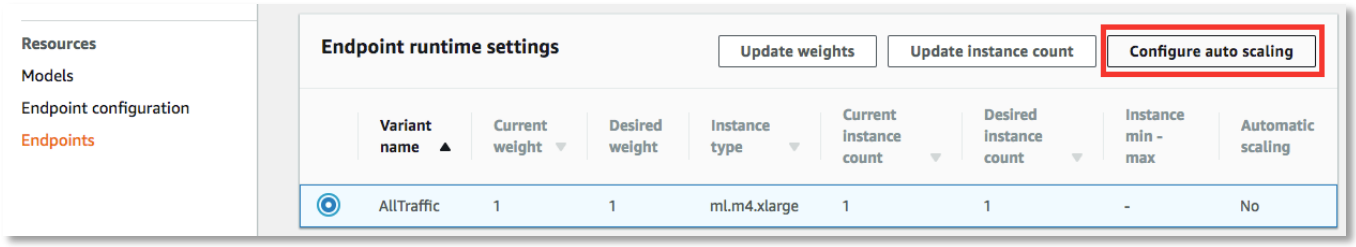
위와 같이 Endpoint 서버 설정에서의 Instance count 는 "InitialInstanceCount"로 설정이 됩니다. 즉 초기의 서버 갯수 만을 설정하는 것이고 사용자의 요청 부하에 따라 서버 설정이 바뀌게 할 수 있습니다. 아래에는 AWS SageMaker 콘솔을 이용해서 autoscaling 을 설정 하는 방법을 보겠습니다.

1. AWS SageMaker 콘솔에서 왼편의 Endpoints 를 선택하신 후 오른편 화면에서 생성하신 Endpoint 를 선택합니다



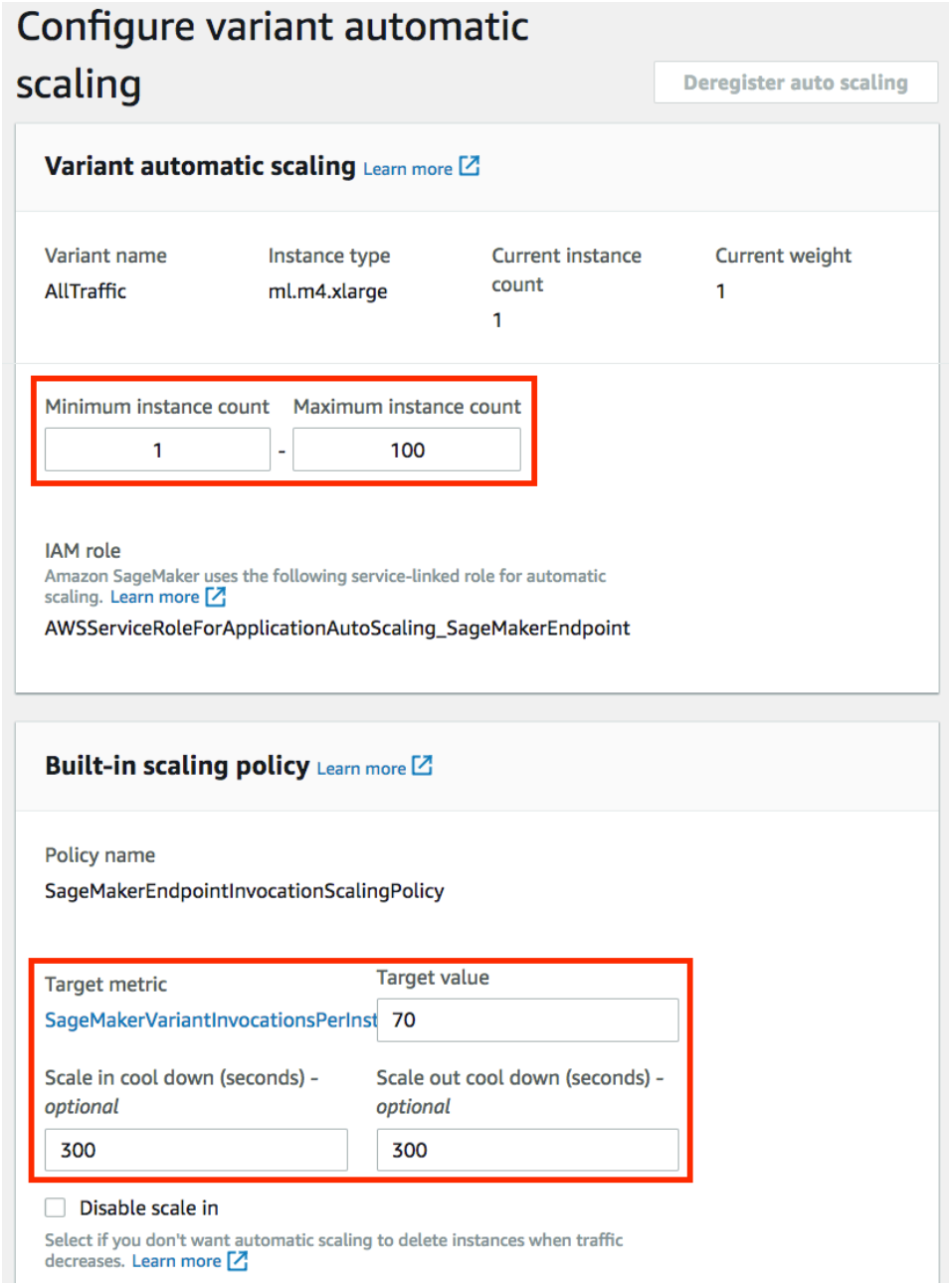
AWS 콘솔에서 SageMaker 의 Endpoints 선택 화면

2. 선택된 Endpoint 내용 화면에서 스크롤을 하셔서 Endpoint runtime settings 에서 AllTraffic 을 선택하신 후 오른편의 Configure auto scaling 버튼을 선택합니다. 참고로 이 화면에서 각 Variant 별 Weight 변경 (Update Weigths)와 평상시 서버 개수 (Update Instance count) 도 변경하실 수 있습니다.



Auto scaling 설정 화면.

3. Configure variant automatic scaling 화면에서는 Variant automatic scaling 과 Scaling policy 를 설정 하실 수 있습니다 (참조링크). Amazon SageMaker 는 target-tracking scaling 정책을 사용하고 있습니다. 즉 미리 정의된 metric 이나 custom metric 을 사용하셔서 target value 를 지정하실 수 있는데 CloudWatch 알람을 통해 scaling 정책을 구동 시키고 instance server scale 을 조정하실 수 있습니다. 본 핸즈온에서는 직접 다루지는 않지만 참조링크를 통해 좀 더 자세한 내용을 파악해 보시는 것도 좋을 것 같습니다.



Automatic scaling 정책 설정 화면.

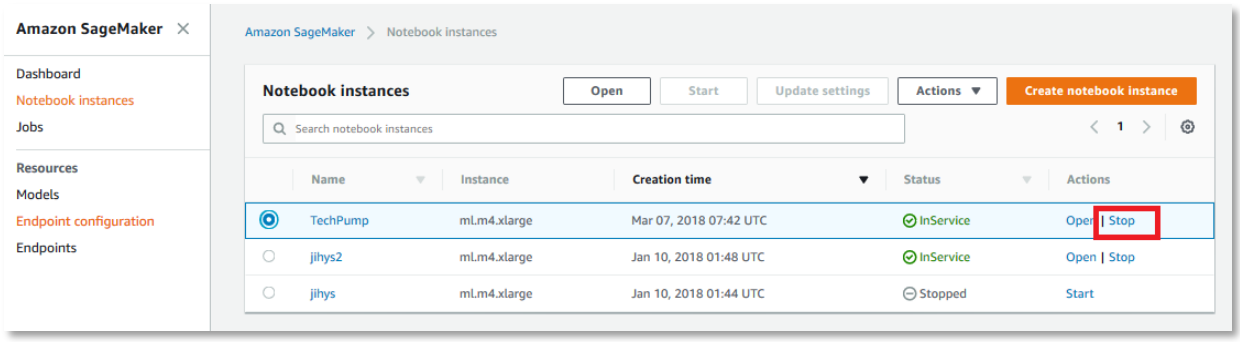
이상으로 모듈 6 의 실습 과정을 마무리 하셨습니다. 워크 샵 이후 발생되는 비용을 방지하기 위해 다음 페이지의 서비스 종료 가이드를 통해 사용하신 리소스들을 모두 종료/삭제 해주십시오.

서비스 종료 가이드

워크 샵 이후 발생 되는 비용을 방지하기 위해서 아래의 단계에 따라 모두 종료/삭제 해 주세요. 비용이 발생하더라도 실습하신 Internet-facing App 을 유지하고 싶으신 경우에는 아래의 Notebook instance 의 경우만 처리하시면 됩니다.

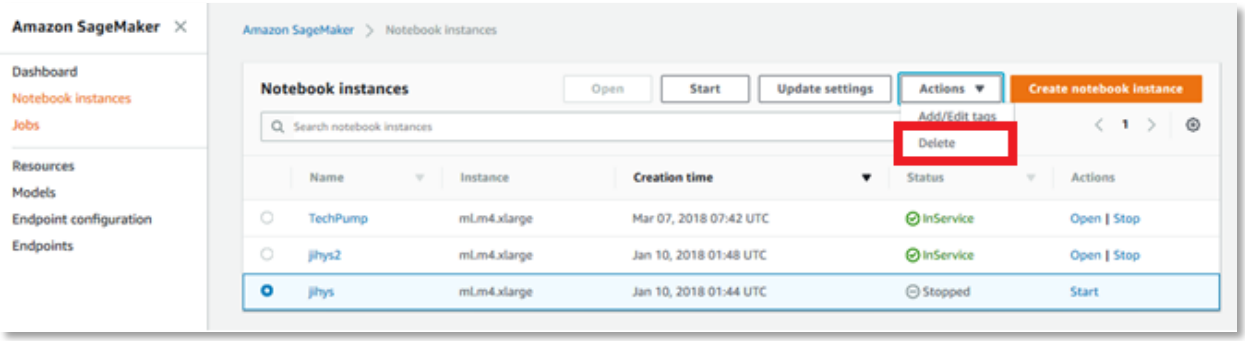
- **Notebook instance:**

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



SageMaker 노트북 인스턴스 중단 화면.

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

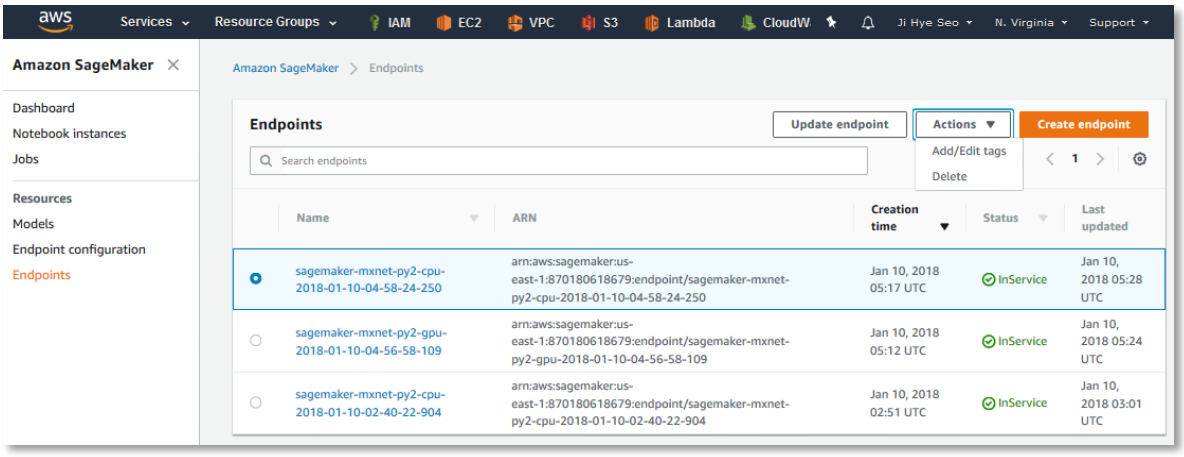


SageMaker 노트북 인스턴스 삭제 화면.

- **SageMaker Endpoints:**

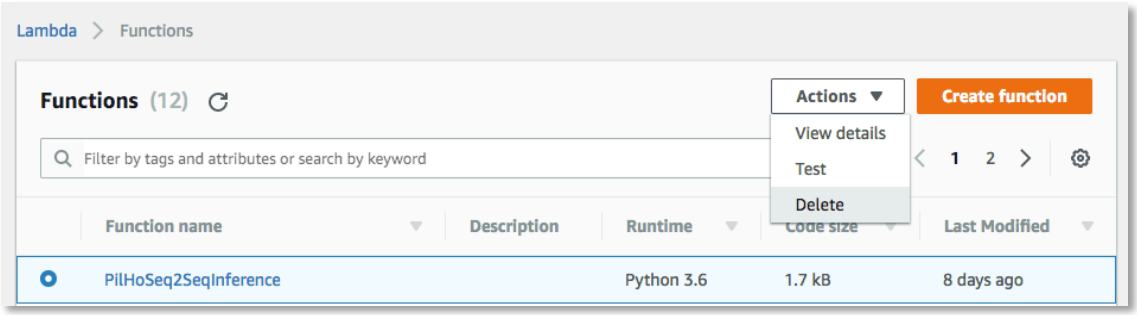
훈련된 모델을 실제 예측 업무를 위해 배포된 한대 이상으로 구성된 클러스터입니다. Notebook 안에서 명령어로 삭제하거나 SageMaker console 에서 삭제 하실 수 있습니다. 삭제 하시기 위해서는 왼쪽

패널의 Endpoints 를 선택 하신 후 해당 endpoints 들 옆에 radio button 을 클릭 하신 후 Action Dropdown 메뉴에서 Delete 선택 하시면 됩니다.



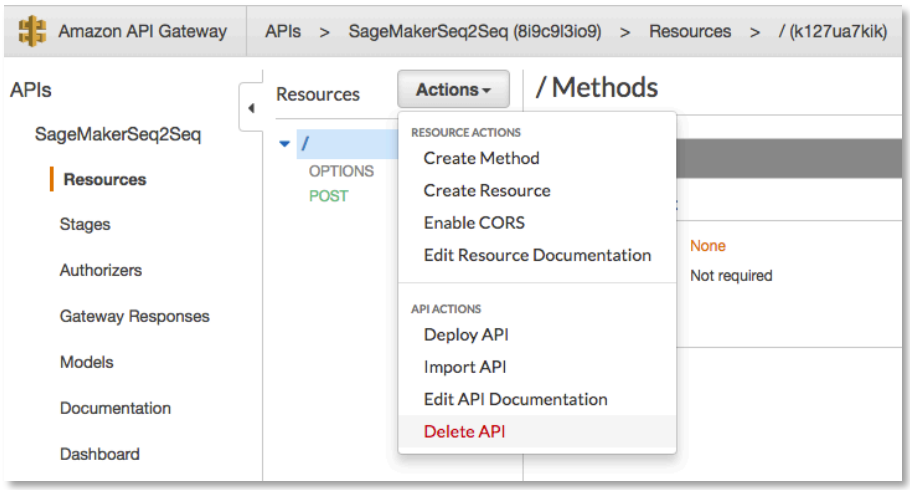
SageMaker Endpoint 삭제 화면.

- **Lambda instance: 생성하신 Lambda instance 를 삭제합니다.**



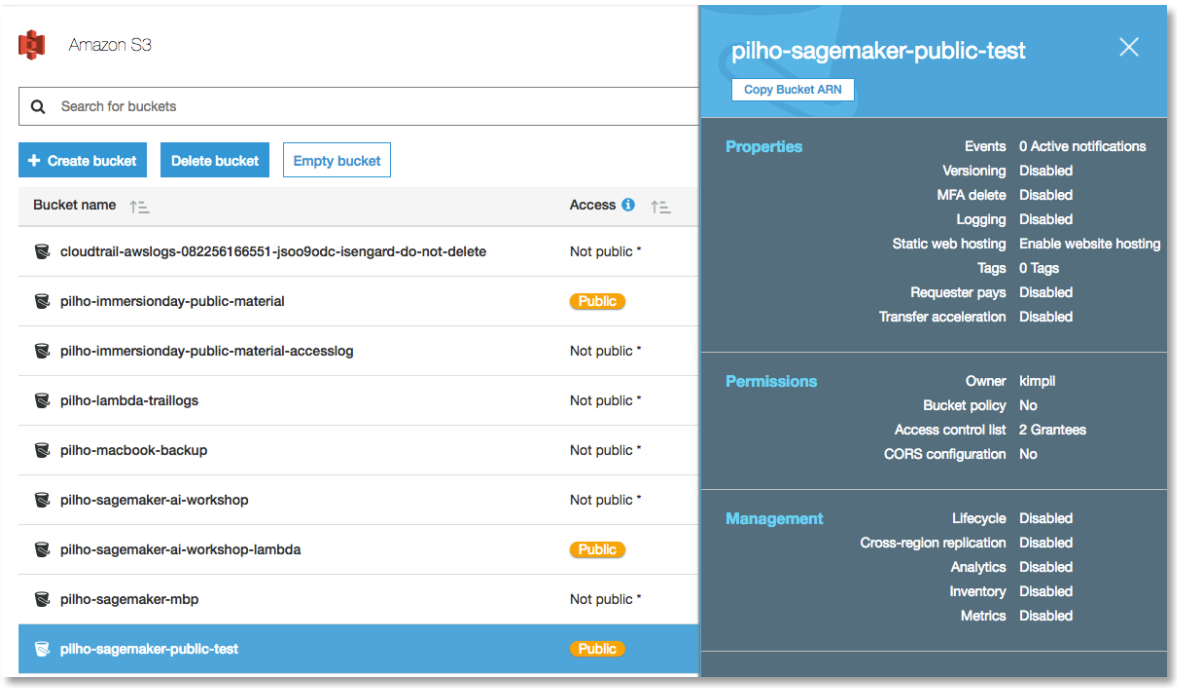
Lambda 인스턴스 삭제 화면.

- **Amazon API Gateway instance: 생성하신 Gateway instance 를 삭제합니다.**



API Gateway 삭제 화면.

- **Amazon S3 buckets: 생성하신 S3 Bucket (SageMaker 용, Public Internet 용)들을 모두 삭제합니다.**



S3 버킷 삭제 화면.

이상으로 본 핸즈온 세션의 모든 과정을 마무리 하셨습니다. 수고하셨습니다.