



# Amazon SageMaker

## Training & Processing

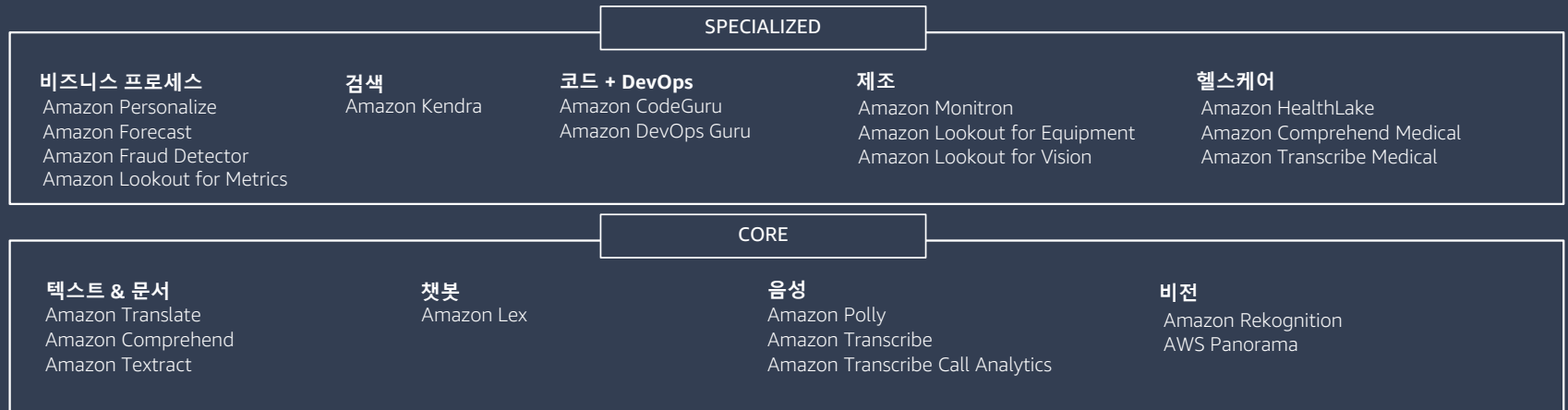
장동진 (dongjinj@amazon.com)

AIML Specialist SA

AWS

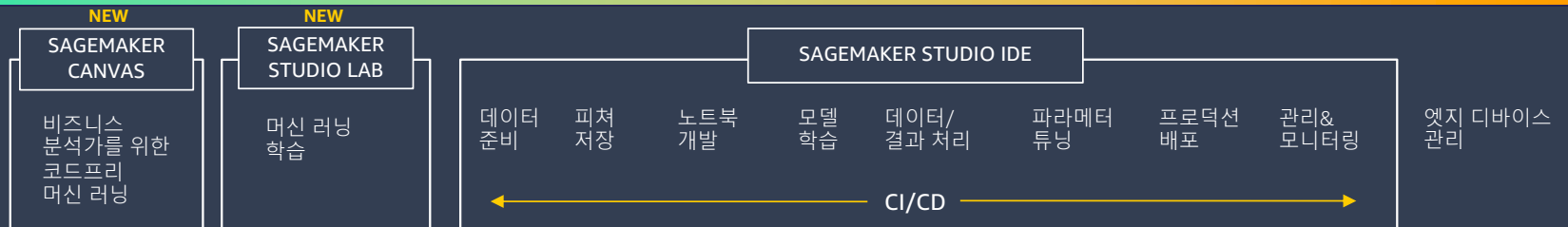
# The AWS ML Stack

## AI 서비스



## AMAZON SAGEMAKER

데이터 레이블링



## ML 프레임워크 & 인프라

TensorFlow, PyTorch, Apache MXNet, **Hugging Face** | Deep learning AMIs & containers | CPUs | GPUs | AWS Inferentia | **NEW** AWS Trainium | Habana Gaudi | Elastic inference | FPGA



# SageMaker Training - 모델 학습



## SageMaker Training은 무엇일까요?

- 완전 관리형 머신 러닝 학습 서비스
- 데이터 과학자가 빠르고 쉽게 모델 개발 및 학습을 할 수 있도록 지원



# SageMaker Training 학습 환경

## 1. 기본 Package들이 설치된 AWS가 제공하는 컨테이너 이미지 사용





## 2. 고객이 추가로 필요한 Package들이 설치된 컨테이너 이미지 생성 후 사용



# SageMaker Training 학습 환경

3. 알고리즘 + 기본 Package들이 설치된 AWS가 제공하는 컨테이너 이미지 사용

 고객 준비 영역  
 AWS 관리 영역



# Amazon SageMaker built-in algorithms

## Classification

- Linear Learner \*
- XGBoost
- KNN
- Factorization Machines

## Working with Text

- Blazing Text
  - Supervised
  - Unsupervised \*

## Sequence Translation

- Seq2Seq \*

## Computer Vision

- Image Classification <>
- Object Detection <>
- Semantic Segmentation

## Recommendation

- Factorization Machines \* (+ KNN)

## Anomaly Detection

- Random Cut Forests \*
- IP Insights \*

## Regression

- Linear Learner
- XGBoost
- KNN

## Topic Modeling

- LDA
- NTM

## Forecasting

- DeepAR \*

## Clustering

- Kmeans \*
- KNN

## Feature Reduction

- PCA
- Object2Vec



# 모델 학습 환경의 구성

1. **SageMaker 노트북 생성**
2. 학습 코드 내 경로 수정
3. 학습 작업의 실행 노트북 작성



# SageMaker Training - 노트북 생성

SageMaker 노트북은 클라우드에서 제공되는 Jupyter notebook/Lab UI

Data  
Scientists

aws AWS 고객 환경

노트북

노트북  
인스턴스

볼륨

SageMaker 노트북

```
• Have the predictor variable in the first column
• Not have a header row

But first, let's convert our categorical features into numeric features.

[ ]: model_data = pd.get_dummies(churn)
     model_data = pd.concat([model_data['Churn? True'], model_data.drop(['Churn? True'], axis=1)], axis=1)
     ...

And now let's split the data into training, validation, and test sets. This will help prevent us from overfitting the model, and allow us to test the models accuracy on data it hasn't already seen.

[ ]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=42), [int(len(model_data)*0.33), int(len(model_data)*0.66)])
     train_data.to_csv('train.csv', header=False, index=False)
     validation_data.to_csv('validation.csv', header=False, index=False)
     ...

Now we'll upload these files to S3.

[ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train.csv')).upload_file(train_data.to_csv('train.csv', header=False, index=False), bucket)
     boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'validation.csv')).upload_file(validation_data.to_csv('validation.csv', header=False, index=False), bucket)
     ...
```

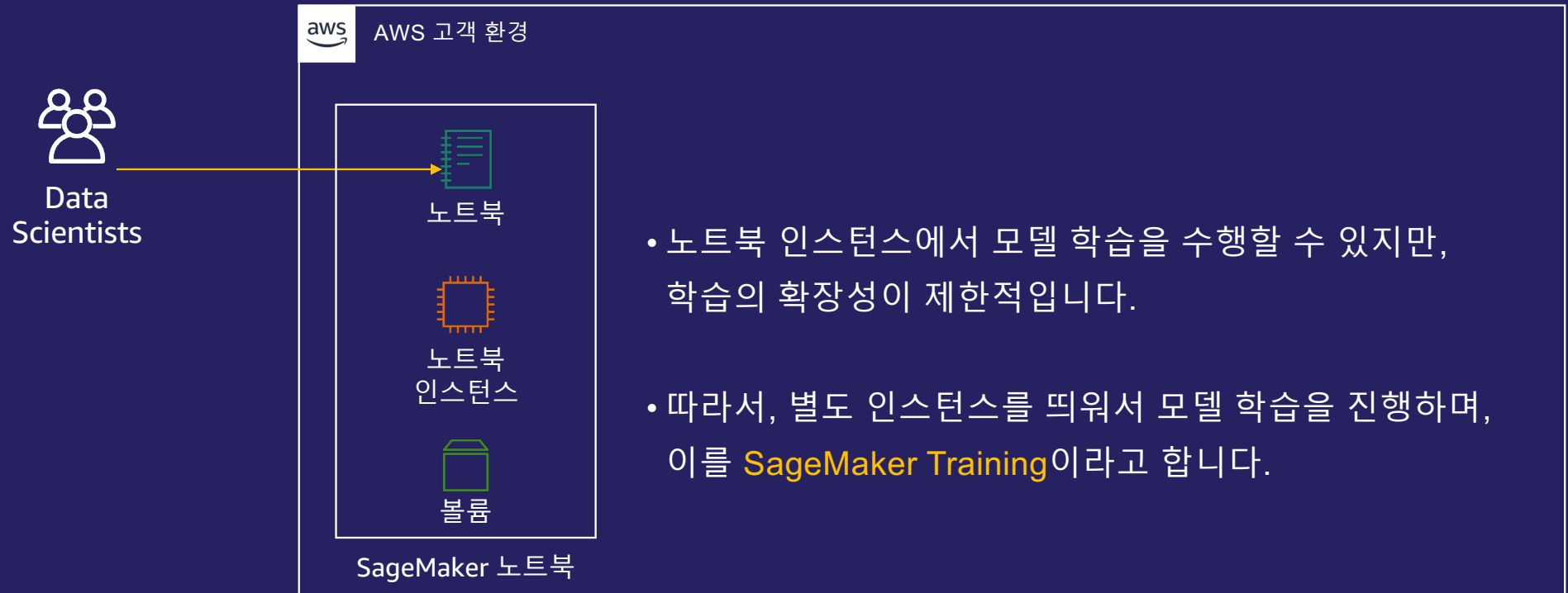
Trial Component Chart

Trial Component List

Status	Experiment	Type	Trial	Trial ID
Completed	customer-churn-predi...	Training job	Trial-3	Trial-3
Completed	customer-churn-predi...	Training job	Trial-2	Trial-2
Completed	customer-churn-predi...	Training job	Trial-1	Trial-1
Completed	customer-churn-predi...	Training job	Trial-0	Trial-0

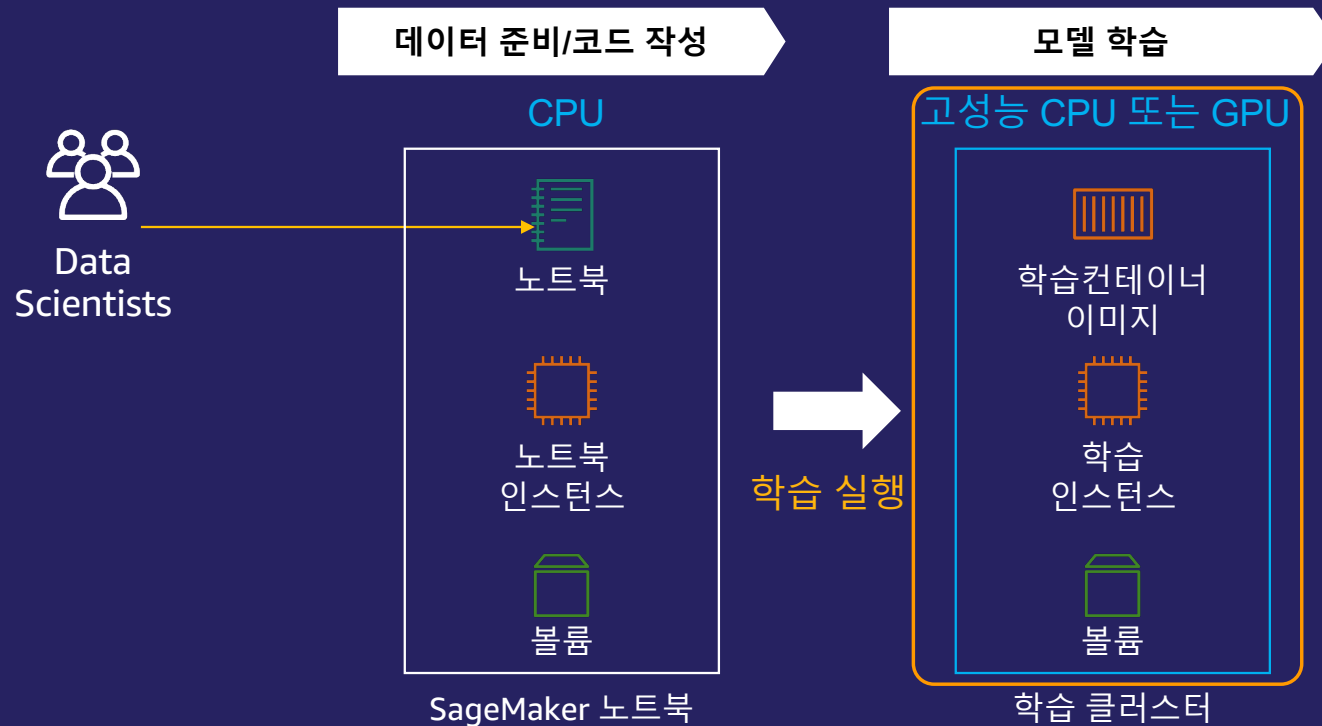
# 노트북 인스턴스에서 학습을 진행하나요

노트북은 데이터 준비, 학습 코드 수정, 학습 작업 실행 노트북 파일을 작성



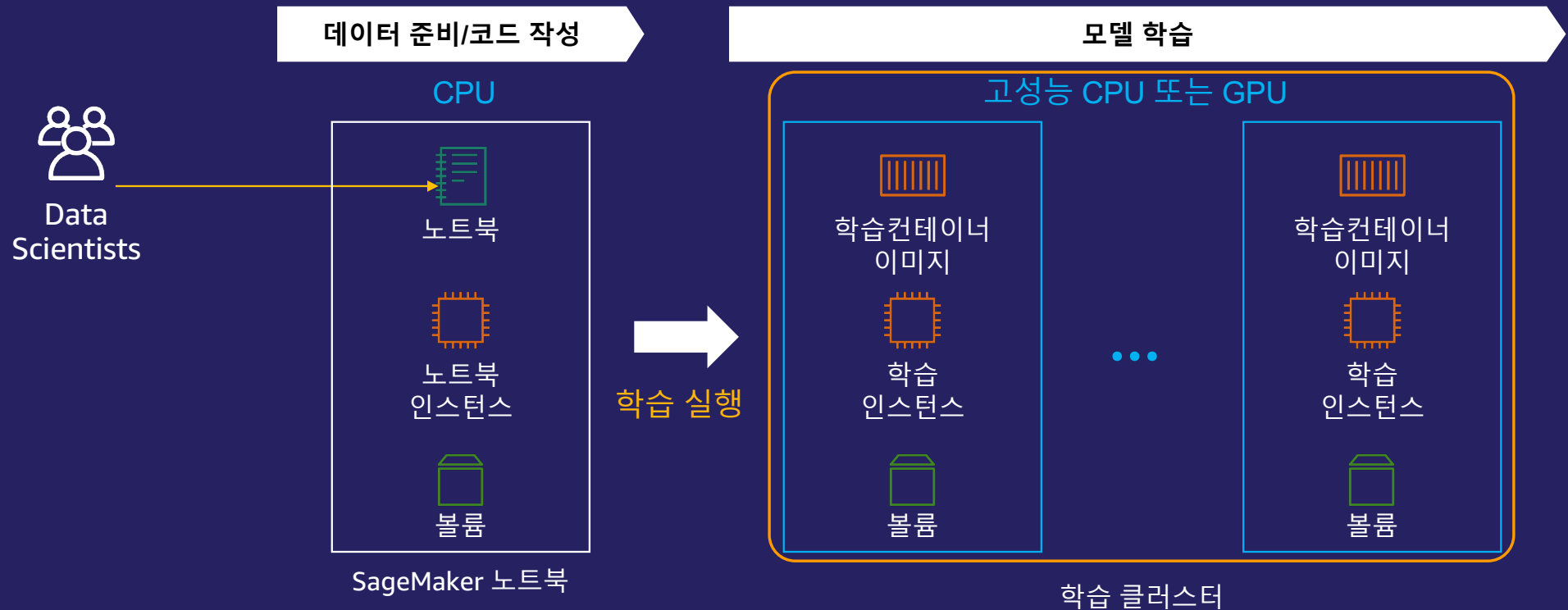
# SageMaker Training

학습은 노트북 인스턴스 대신 고성능 CPU 또는 GPU 인스턴스에서 수행하여 효율적으로 활용

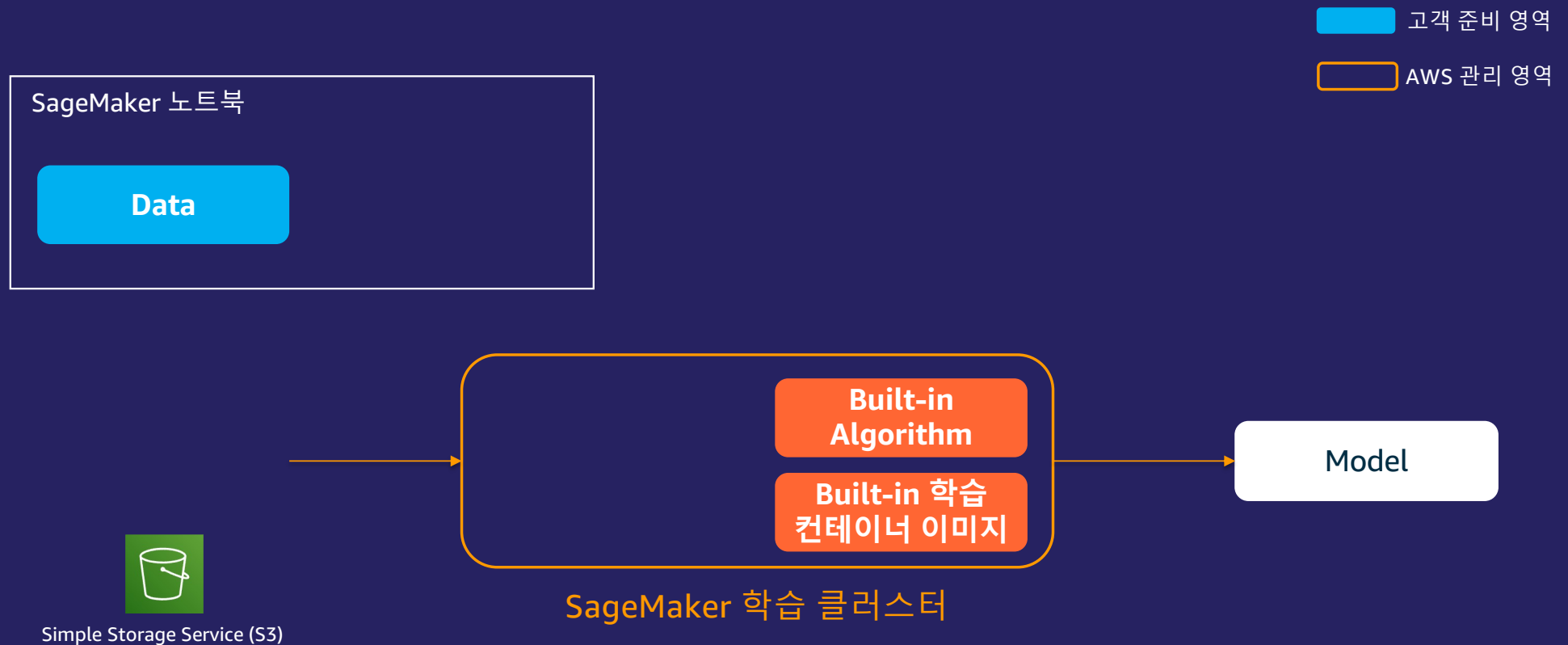


# SageMaker Training

다수의 인스턴스로 확장하여 학습이 가능 (단, 학습 코드는 분산학습 라이브러리 적용 필요)

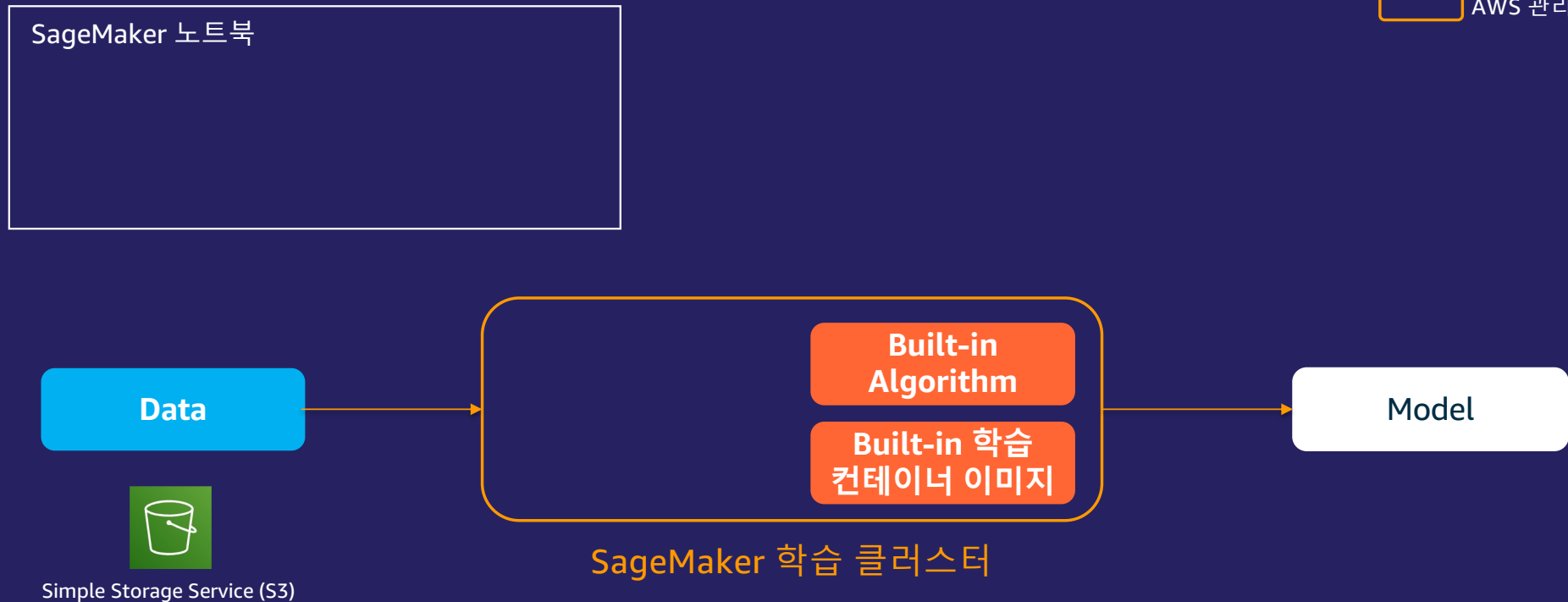


# SageMaker Training 동작 방식 (1)

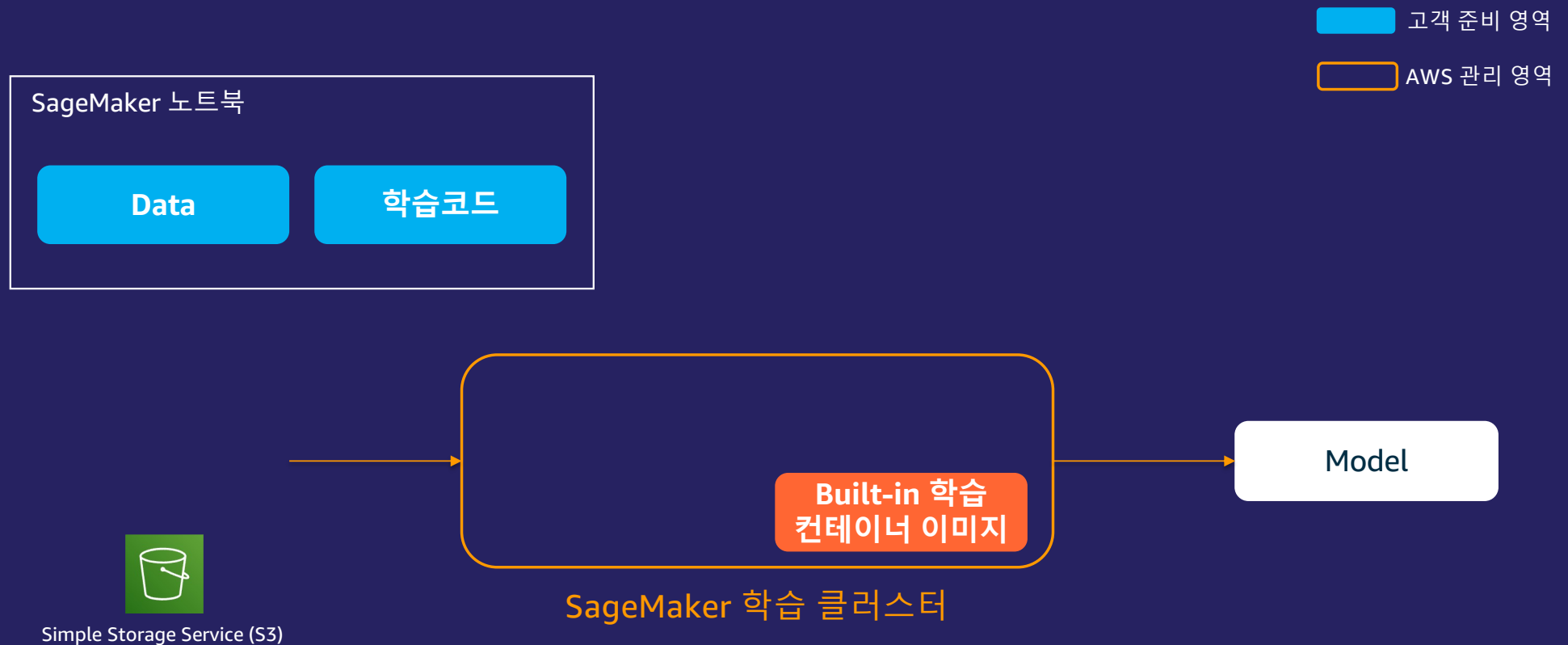


# SageMaker Training 동작 방식 (1)

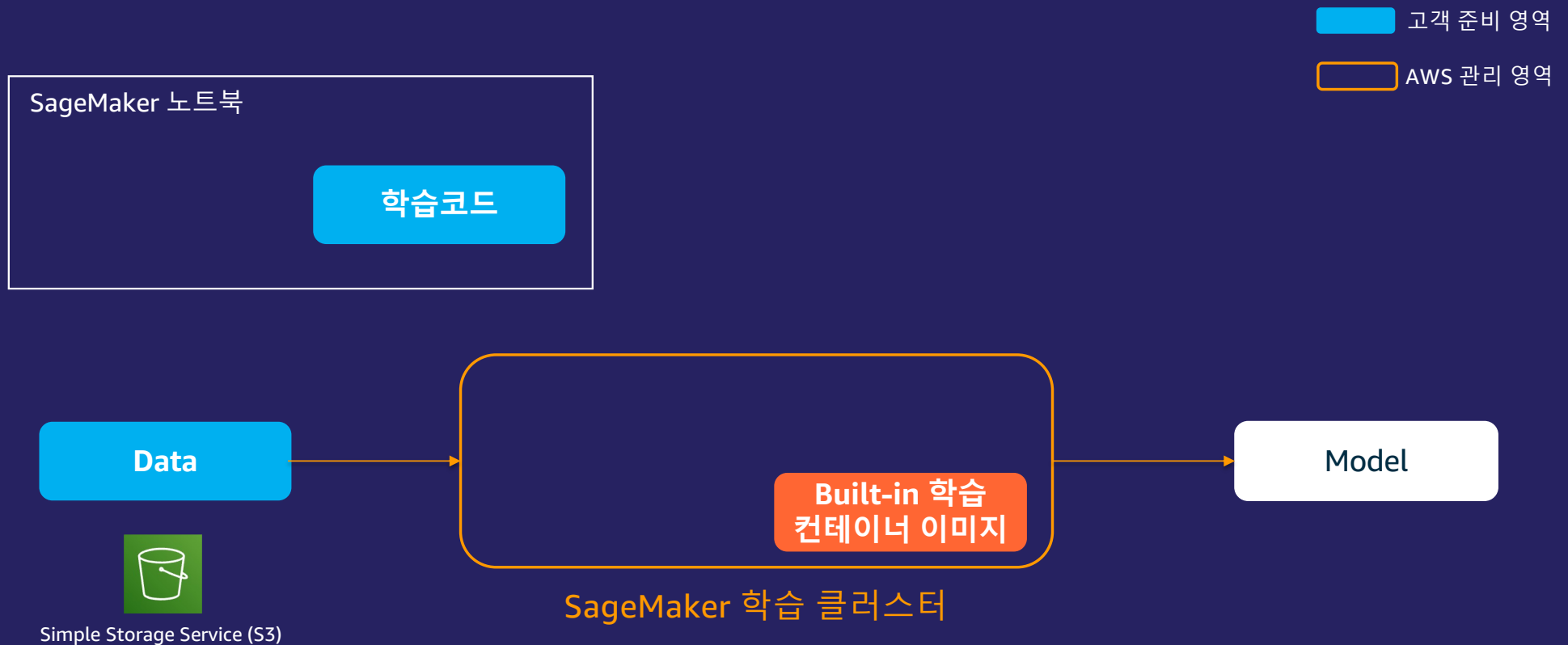
고객 준비 영역  
AWS 관리 영역



## SageMaker Training 동작 방식 (2)

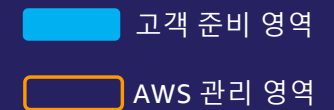
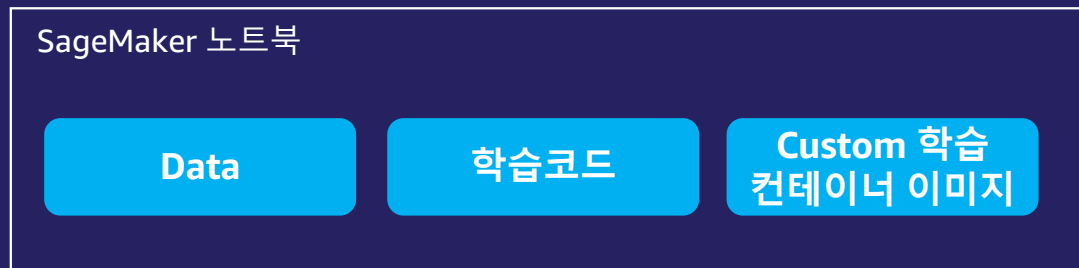


## SageMaker Training 동작 방식 (2)

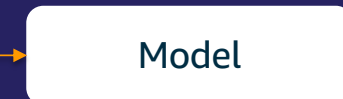




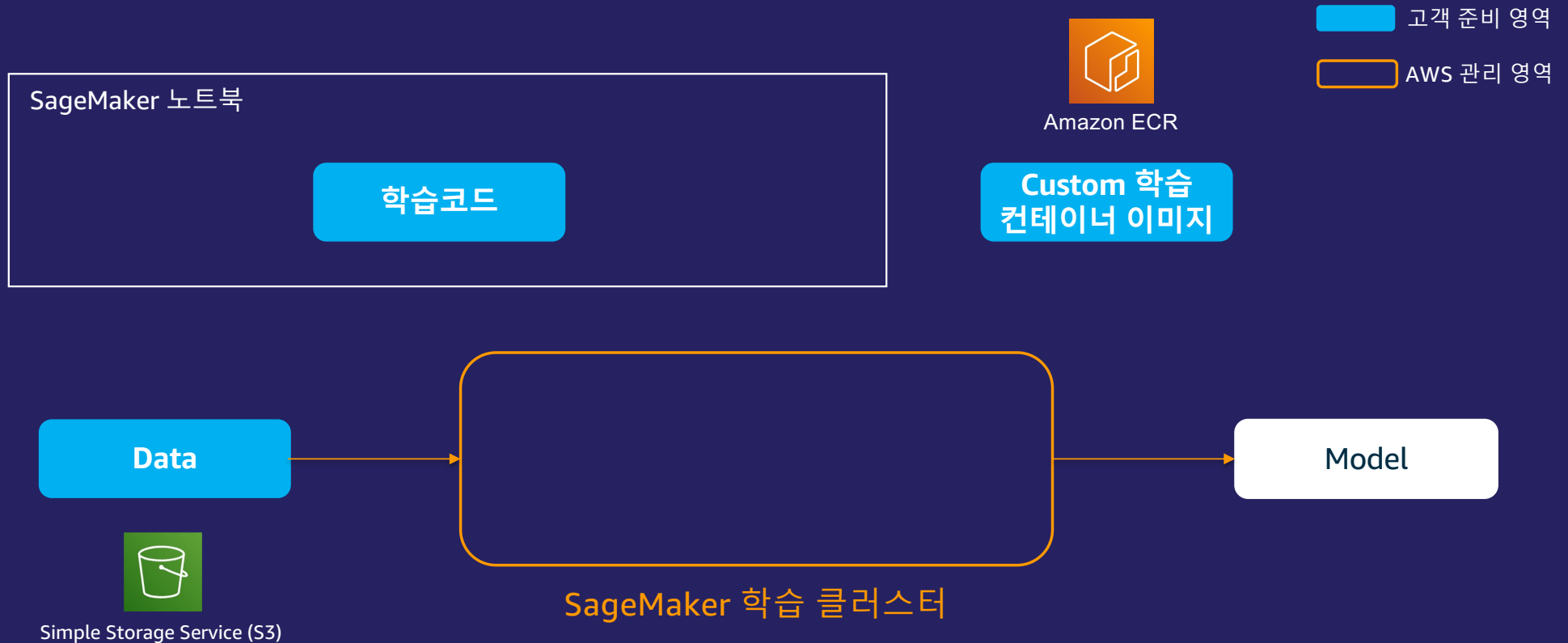
# SageMaker Training 동작 방식 (3)



SageMaker 학습 클러스터



# SageMaker Training 동작 방식 (3)



# SageMaker Training을 위한 작업

## 학습 코드 (Python 코드)

```
def main():
    parser = argparse.ArgumentParser()

    #####
    ## 커맨드 인자 처리
    #####

    # Hyperparameters are described here
    parser.add_argument('--scale_pos_weight', type=int, default=50)
    parser.add_argument('--num_round', type=int, default=999)
    parser.add_argument('--max_depth', type=int, default=3)
    parser.add_argument('--eta', type=float, default=0.2)
    parser.add_argument('--objective', type=str, default='binary:logistic')
    parser.add_argument('--nfold', type=int, default=5)
    parser.add_argument('--early_stopping_rounds', type=int, default=10)
    parser.add_argument('--train_data_path', type=str, default='./dataset')

    # SageMaker specific arguments. Defaults are set in the environment variables.
    parser.add_argument('--model-dir', type=str, default='./model')
    parser.add_argument('--output-data-dir', type=str, default='./output')

    args = parser.parse_args()

    ## Check Training Sagemaker
    args = train_sagemaker(args)

    #####
    ## 데이터 세트 로딩 및 변환
    #####
```

## 학습 작업의 실행 노트북 (ipynb)

### SageMaker 세션과 Role, 사용할 버킷 정의

```
[68]: sagemaker_session = sagemaker.session.Session()
      role = sagemaker.get_execution_role()

[69]: bucket = sagemaker_session.default_bucket()
      code_location = f's3://{bucket}/code'
      output_path = f's3://{bucket}/output'
```

### 하이퍼파라미터 정의

```
[80]: hyperparameters = {
      "scale_pos_weight": "29",
      "max_depth": "3",
      "eta": "0.2",
      "objective": "binary:logistic",
      "num_round": "100",
      }
```

### (1) 학습 실행 작업 정의

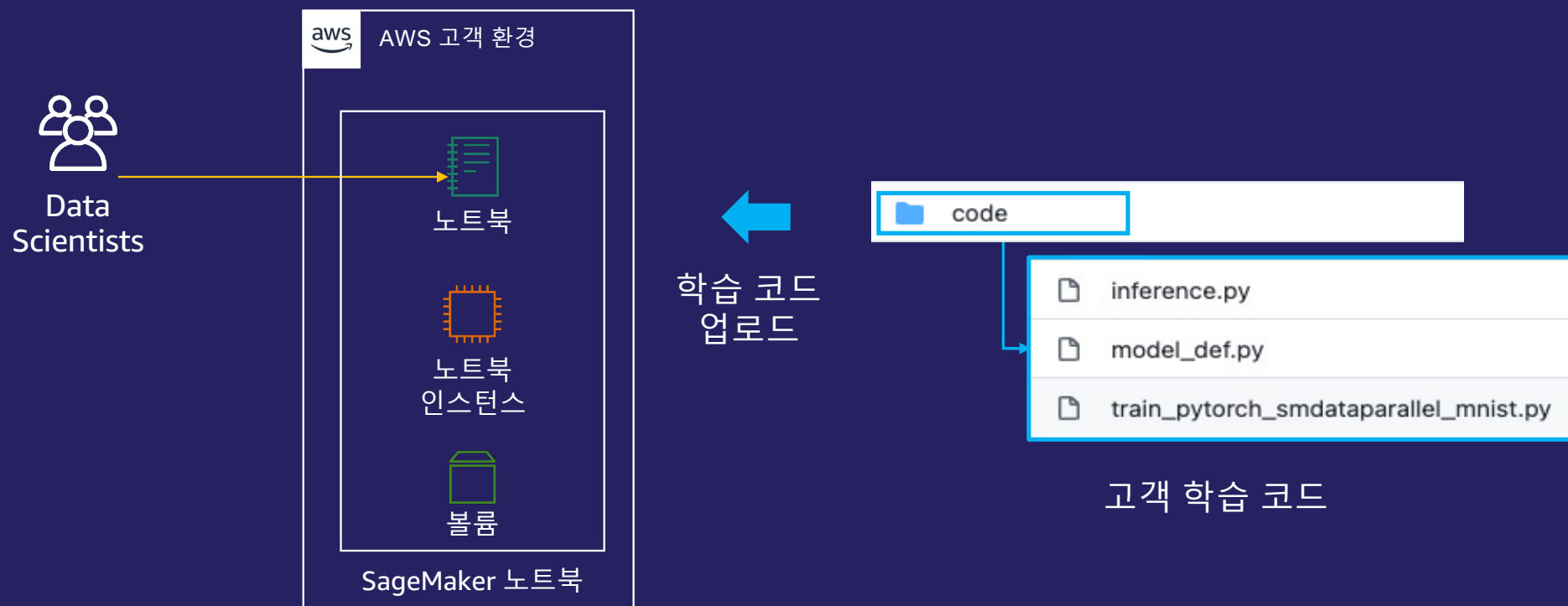
```
[81]: xgb_estimator = XGBoost(
      entry_point = "xgboost_starter_script.py",
      source_dir = "src",
      output_path = estimator_output_path,
      code_location = estimator_code_path,
      hyperparameters = hyperparameters,
      role = role,
      sagemaker_session = sagemaker_session,
      instance_count = 1,
      instance_type = "ml.m5.xlarge",
      framework_version = "1.3-1")
```

# 모델 학습 환경의 구성

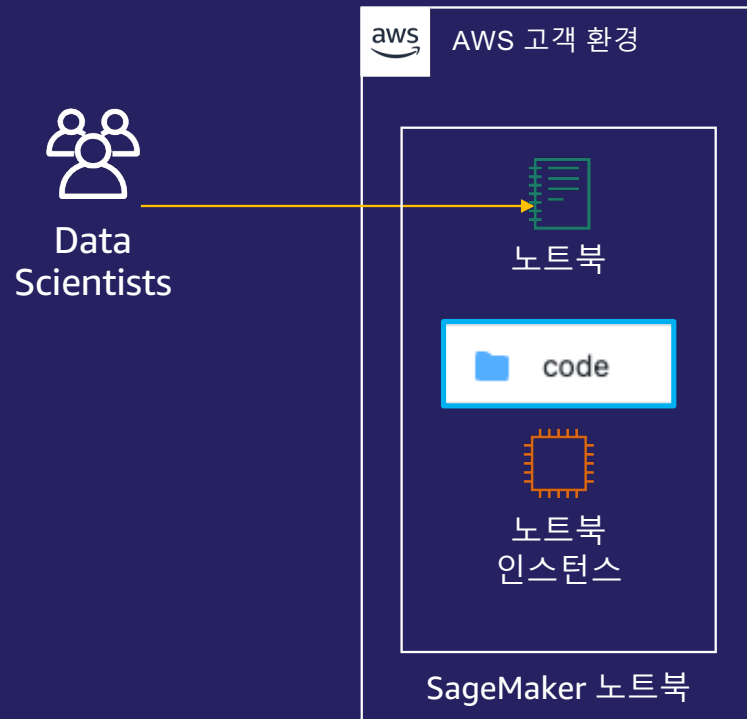
1. SageMaker 노트북 생성
2. 학습 코드 내 경로 수정
3. 학습 작업의 실행 노트북 작성

# SageMaker Training - 학습 코드 업로드

on-premise (로컬 환경)에서 학습이 되는 코드는 폴더 생성 후 폴더 안에 추가하여 업로드

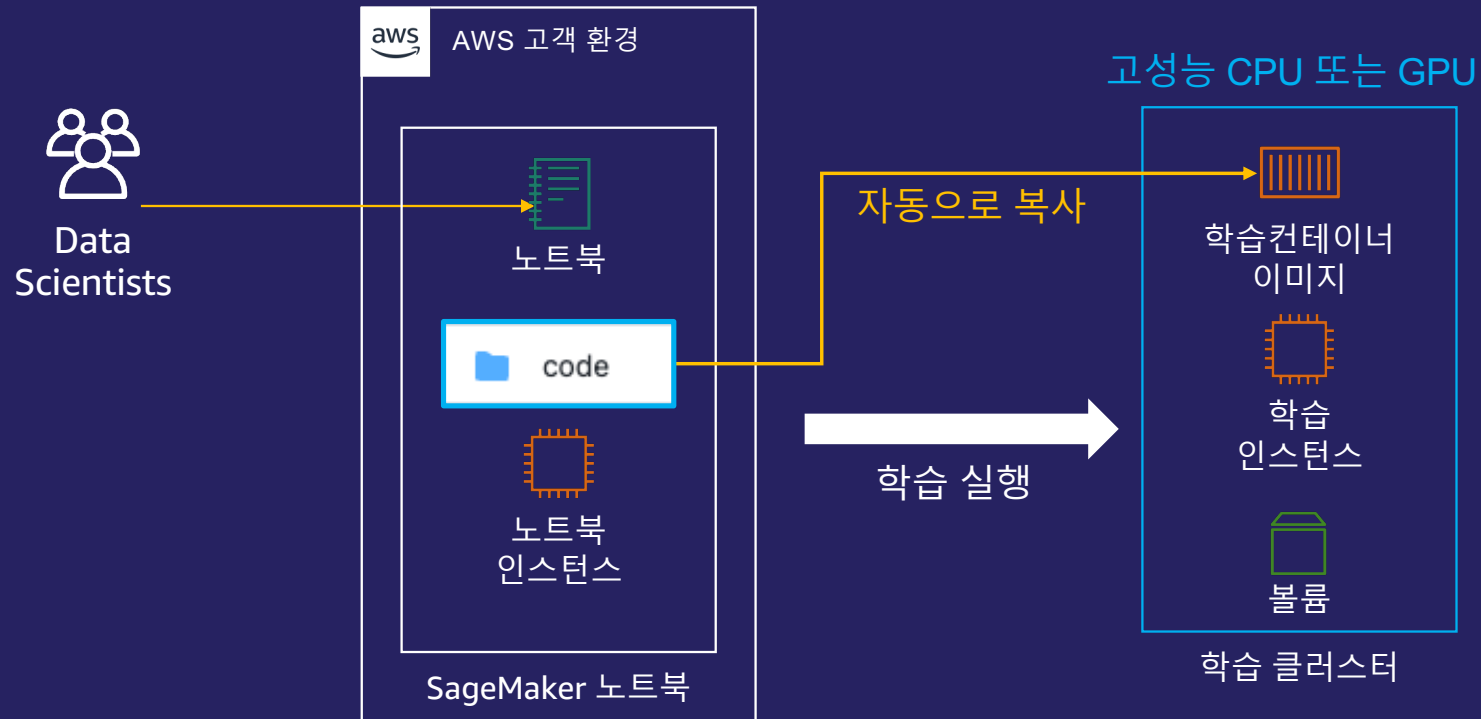


# SageMaker Training - 학습 코드 내 경로 수정



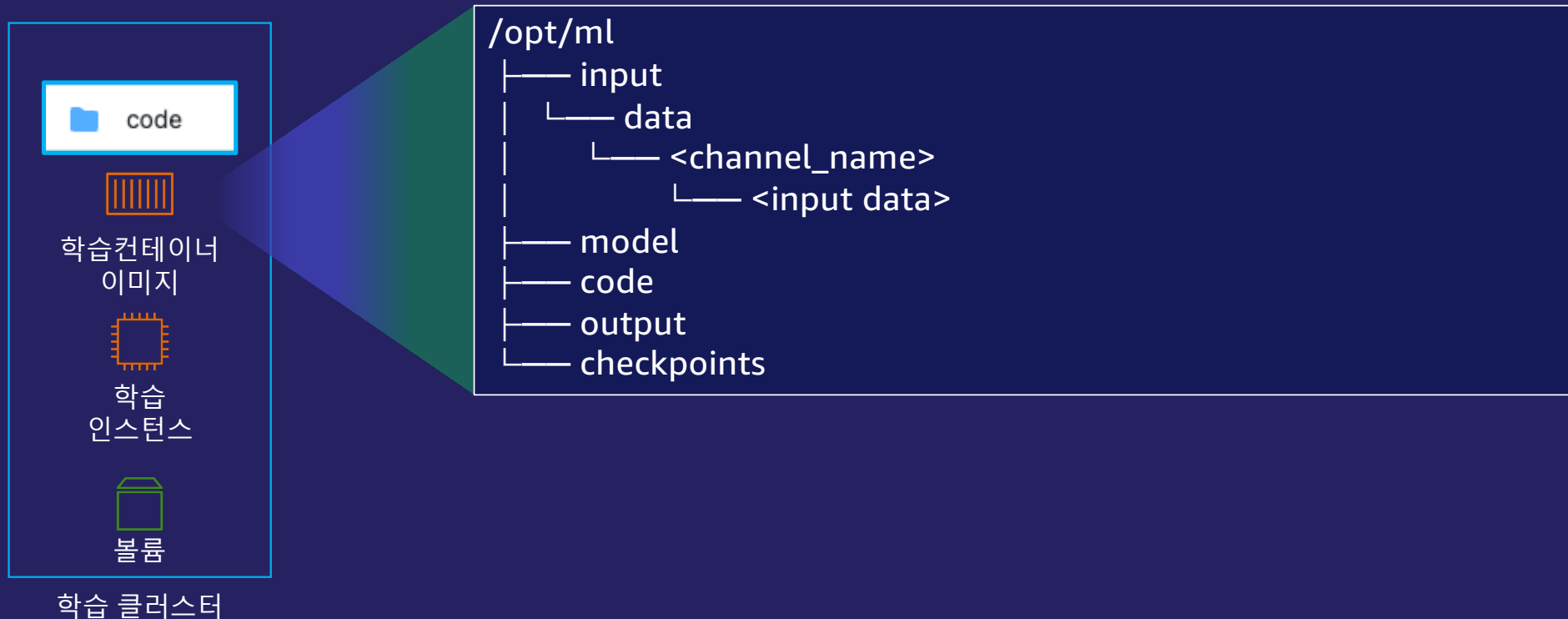
# SageMaker Training - 학습 코드 내 경로 수정

학습이 실행되면 학습 코드는 자동으로 학습 클러스터로 복사



# SageMaker Training - 학습 코드 내 경로 수정

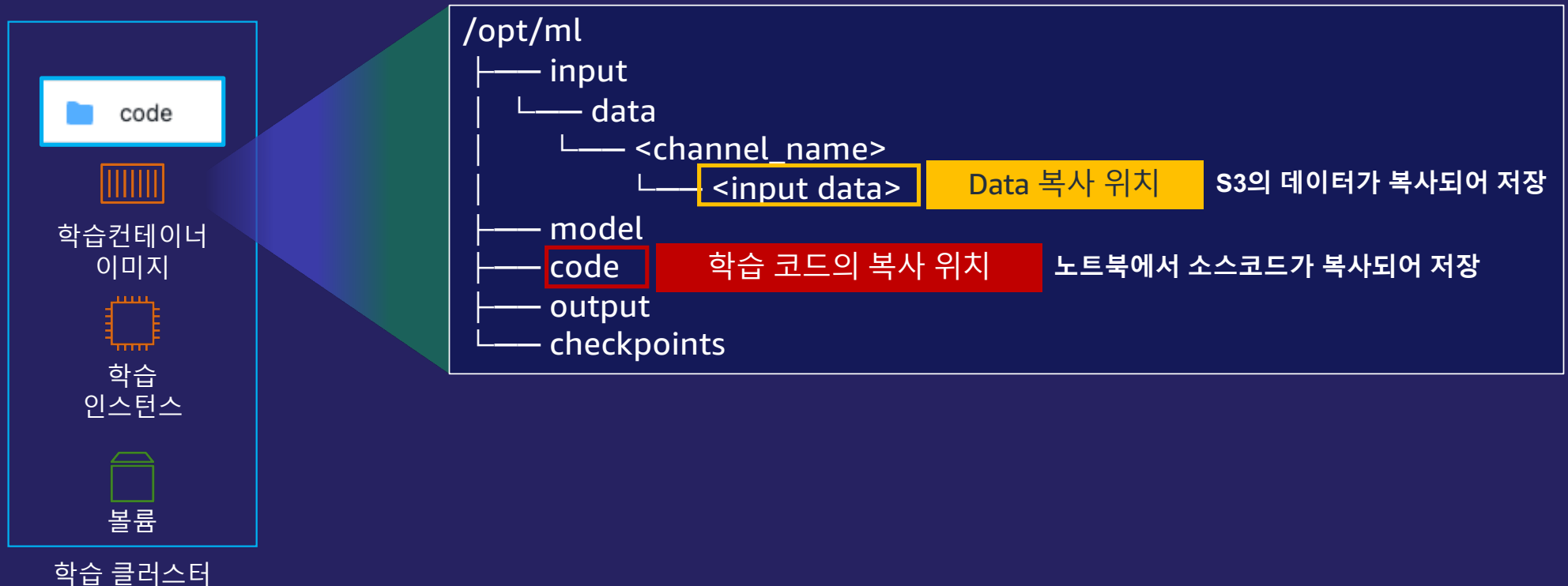
컨테이너 환경에 맞게 학습 코드 내의 경로 변경 필요





# SageMaker Training - 학습 코드 내 경로 수정

컨테이너 환경에 맞게 학습 코드 내의 경로 변경 필요

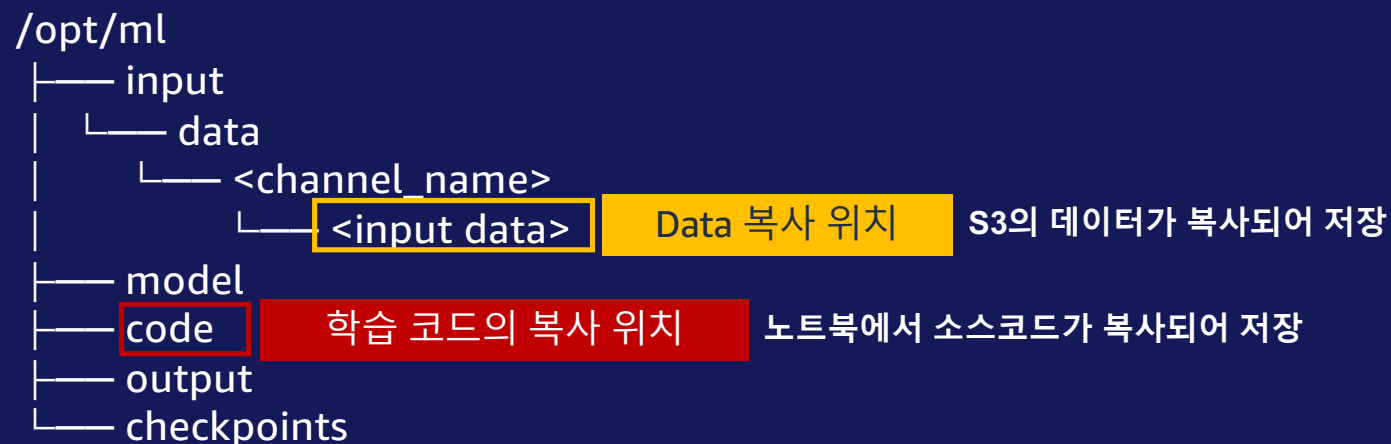


# SageMaker Training - 학습 코드 내 경로 수정

컨테이너 환경에 맞게 학습 코드 내의 경로 변경 필요



학습 클러스터

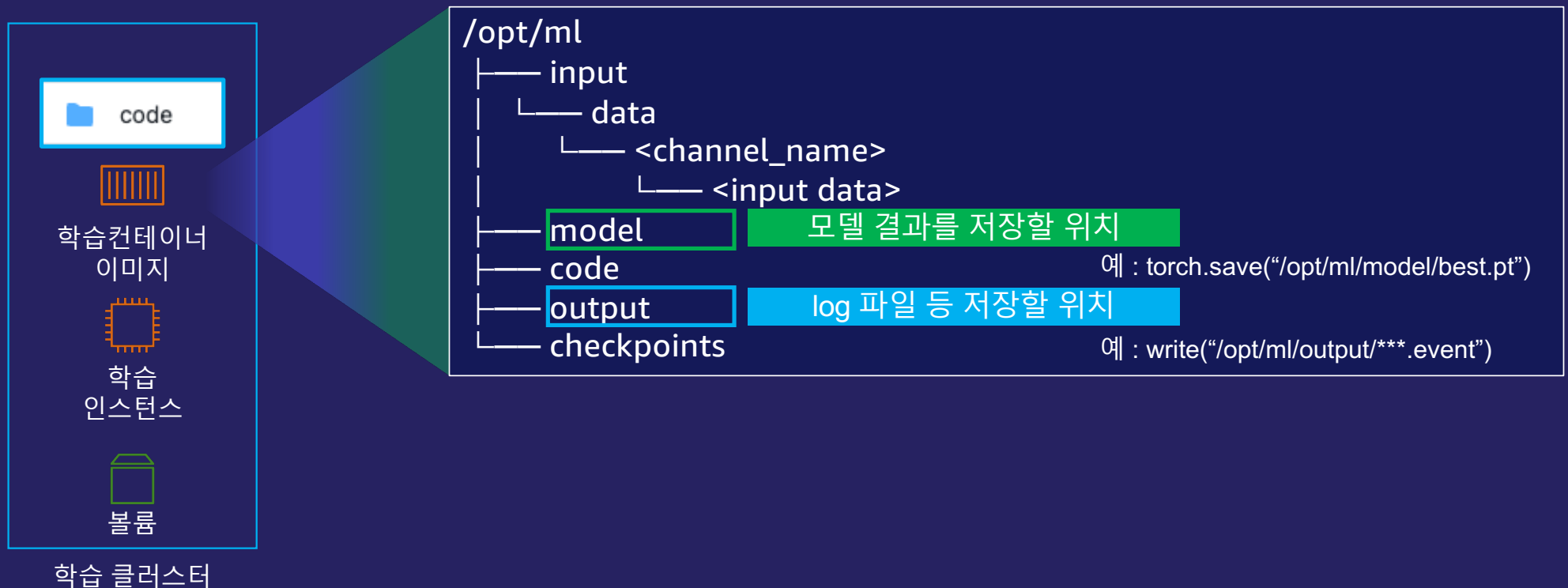


## Dataloader 경로

- `/opt/ml/input/data/{channel_name}` 로 수정
- 별도 환경변수로도 제공  
예) `os.environ.get('SM_CHANNEL_${channel_name}')`

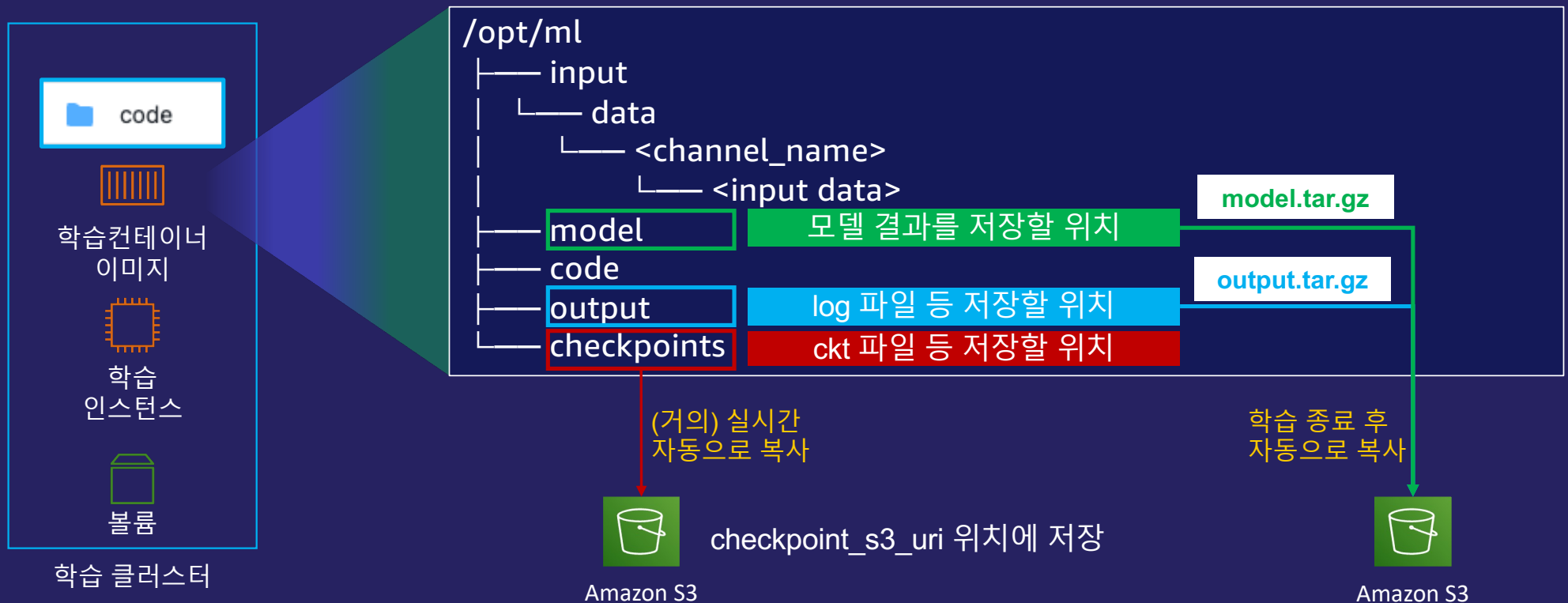
# SageMaker Training - 학습 코드 내 경로 수정

컨테이너 환경에 맞게 학습 코드 내의 경로 변경 필요



# SageMaker Training - 학습 코드 내 경로 수정

컨테이너 환경에 맞게 학습 코드 내의 경로 변경 필요



# SageMaker Training - 학습 코드 내 경로 수정

SageMaker 컨테이너는 주요 경로를 환경변수로 제공

```
# /opt/ml/model
parser.add_argument('--model_dir', type=str, default=os.environ.get('SM_MODEL_DIR'))

# /opt/ml/input/data/training
parser.add_argument('--dataset_dir', type=str, default=os.environ.get('SM_CHANNEL_TRAINING'))

# /opt/ml/output/data/algo-1
parser.add_argument('--output_data_dir', type=str, default=os.environ.get('SM_OUTPUT_DATA_DIR'))

# /opt/ml/output
parser.add_argument('--output-dir', type=str, default=os.environ.get('SM_OUTPUT_DIR'))
```

[https://github.com/aws/sagemaker-training-toolkit/blob/master/ENVIRONMENT\\_VARIABLES.md](https://github.com/aws/sagemaker-training-toolkit/blob/master/ENVIRONMENT_VARIABLES.md)



# SageMaker Training - 학습 코드 수정 예제

## 학습 코드 수정 전

```
import torch
...
class Net(nn.Module):
    ...
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--model_dir', type=str, default="./result")
    ...
    with open(os.path.join(args.model_dir, 'model.pth'), 'rb') as f:
        model.load_state_dict(torch.load(f))

if __name__ == '__main__':
    main()
```



# SageMaker Training - 학습 코드 수정 예제

## 학습 코드 수정 후

```
import torch
...
class Net(nn.Module):
    ...
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--model_dir', type=str, default=os.environ.get('SM_MODEL_DIR'))
    ...
    with open(os.path.join(args.model_dir, 'model.pth'), 'rb') as f:
        model.load_state_dict(torch.load(f))

if __name__ == '__main__':
    main()
```



# SageMaker Training - 학습 코드 필수 수정 사항

- Dataloader 경로 : `os.environ['SM_CHANNEL_{channel_name}']` 로 수정  
- channel\_name은 학습 작업의 실행 노트북에서 정의

- Model 저장 경로 : `os.environ['SM_MODEL_DIR']` 로 수정

[optional] Output 저장 경로 : `os.environ['SM_OUTPUT_DATA_DIR']` 로 수정

[optional] Checkpoints 저장 경로 : `"/opt/ml/checkpoints"`로 수정





# 모델 학습 환경의 구성

1. SageMaker 노트북 생성
2. 학습 코드 내 경로 수정
3. **학습 작업의 실행 노트북 작성**

# SageMaker Training - 학습 작업의 실행 노트북 작성

학습 클러스터의 인스턴스 종류/수, 실행할 학습 코드, 학습 환경 컨테이너 등을 Estimator로 정의

```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(
    source_dir="code",
    entry_point="train_pytorch_smdataparallel_mnist.py",
    role=role,
    framework_version="1.10",
    py_version="py38",
    instance_count=1,
    instance_type="ml.p4d.24xlarge",
    sagemaker_session=sagemaker_session,
    hyperparameters=hyperparameters,
)

# 학습 코드 폴더 지정
# 실행 학습 스크립트 명
# 학습 클러스터에서 사용할 Role
# Pytorch 버전
# Python 버전
# 학습 인스턴스 수
# 학습 인스턴스 명
# SageMaker 세션
# 하이퍼파라미터 설정
```

# SageMaker Training - 학습 작업의 실행 노트북 작성

학습 클러스터에서 사용할 데이터 경로와 `channel_name`을 선언한 후 실행

```
channel_name = "training"

estimator.fit(
    inputs={channel_name : data_path},
    job_name=job_name
)
```

# SageMaker Training - 학습 작업의 실행 노트북 작성

Estimator에서 정의된 값들을 선언

```
import sagemaker

sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()

hyperparameters = {"batch_size" : 32 ,
                   "lr" : 1e-4 ,
                   "image_size" : 128 }
```

# SageMaker 세션 정의  
# SageMaker 노트북에서 사용하는 role 활용

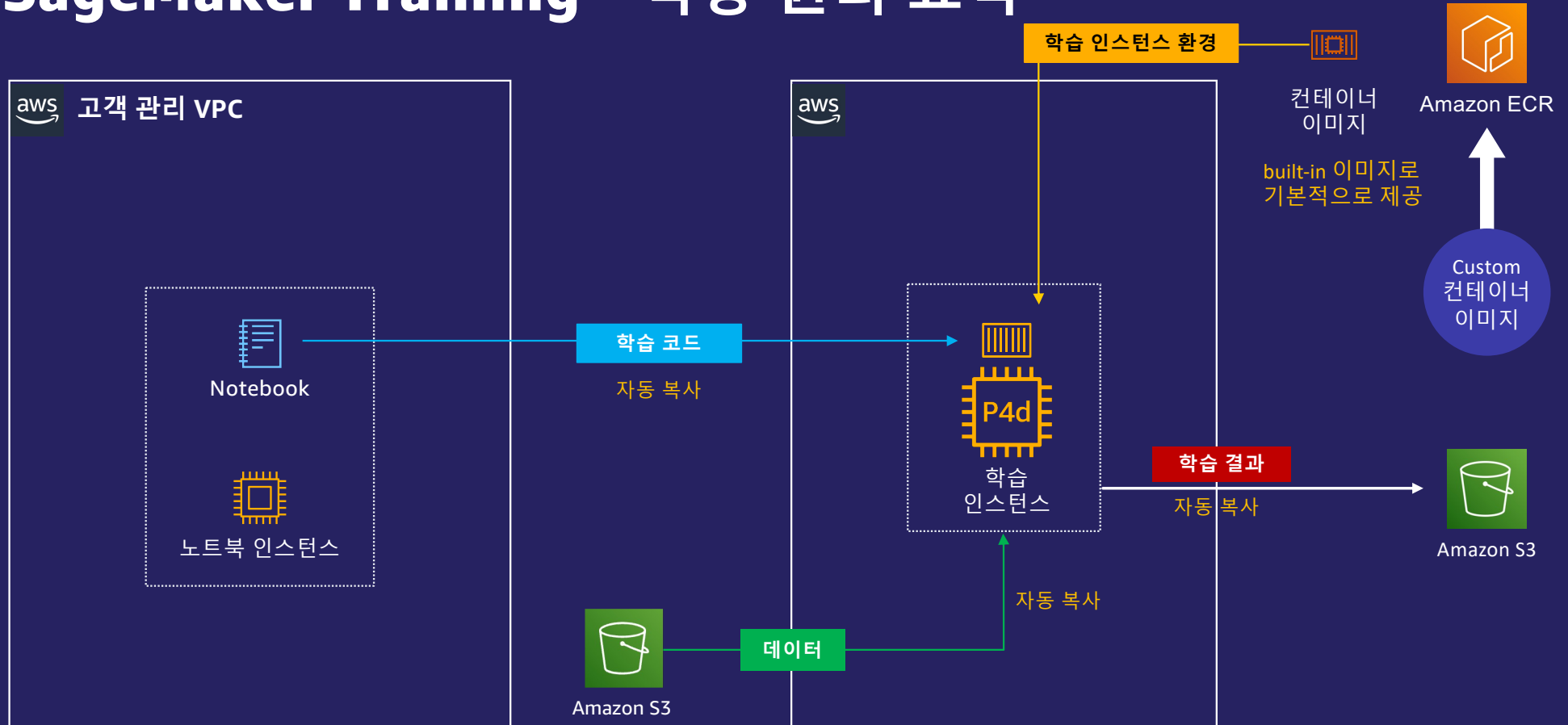
# 학습 코드의 arguments 값

# SageMaker Training – 작업 정의 시 추가 파라미터

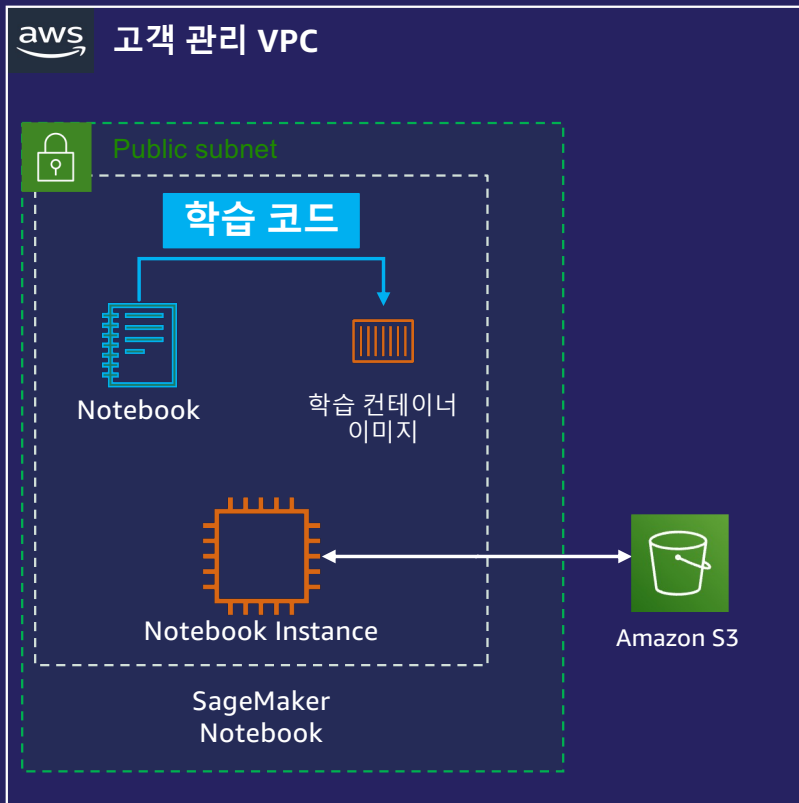
학습 작업과 관련된 다양한 추가 파라미터를 제공

```
estimator = PyTorch(
    ... ,
    max_run=5*24*60*60,           # 최대 학습 수행 시간 (초)
    use_spot_instances=True,      # spot 인스턴스 사용 여부
    max_wait=3*60*60,            # spot 사용 시 자원 재확보를 위한 대기 시간
    checkpoint_s3_uri= checkpoint_s3_uri, # checkpoints 저장 S3 위치
    ...
)
```

# SageMaker Training – 작동 원리 요약



# SageMaker Training의 디버깅



## Local Mode

- 생성한 SageMaker Notebook에서 학습 코드를 개발할 목적으로, 첫번째는 **Local Mode**로 수행
- 예) 딥러닝 분산학습의 경우 노트북 인스턴스를 GPU 유형으로 생성
  - ml.p3.16xlarge의 경우 SageMaker의 Data parallel과 Model parallel Library 테스트 가능
  - **임시적 사용이며 비용을 위해 테스트 후 CPU 유형으로 변경**

# Local Mode를 위한 학습 작업의 실행 노트북 추가

Local Mode

테스트 후  
실학습 시 변경

실제 학습

```
from sagemaker.local import LocalSession
from pathlib import Path

sagemaker_session = LocalSession()
sagemaker_session.config = {'local': {'local_code': True}}
data_path = f'file://{Path.cwd()}/dataset'
source_dir = f'{Path.cwd()}/{source_code_directory}'
checkpoint_s3_bucket = None

instance_type = 'local_gpu'
```

# Local 세션 설정  
# Notebook 인스턴스 내 데이터셋 경로  
# 학습 스크립트가 들어 있는 폴더 경로  
# 인스턴스 타입

```
import sagemaker

sagemaker_session = sagemaker.Session()
source_dir = f'{source_code_directory}'
checkpoint_s3_bucket = f's3://{result_bucket}/checkpoints'

instance_type = 'ml.p4d.24xlarge'
```

# SageMaker 세션 설정  
# 학습 스크립트가 들어 있는 폴더 명  
# Checkpoint 파일 저장 위치 (S3)  
# 인스턴스 타입



# SageMaker의 추가 기능

1. **SageMaker Experiments**
2. 학습 모니터링

# SageMaker의 추가 기능

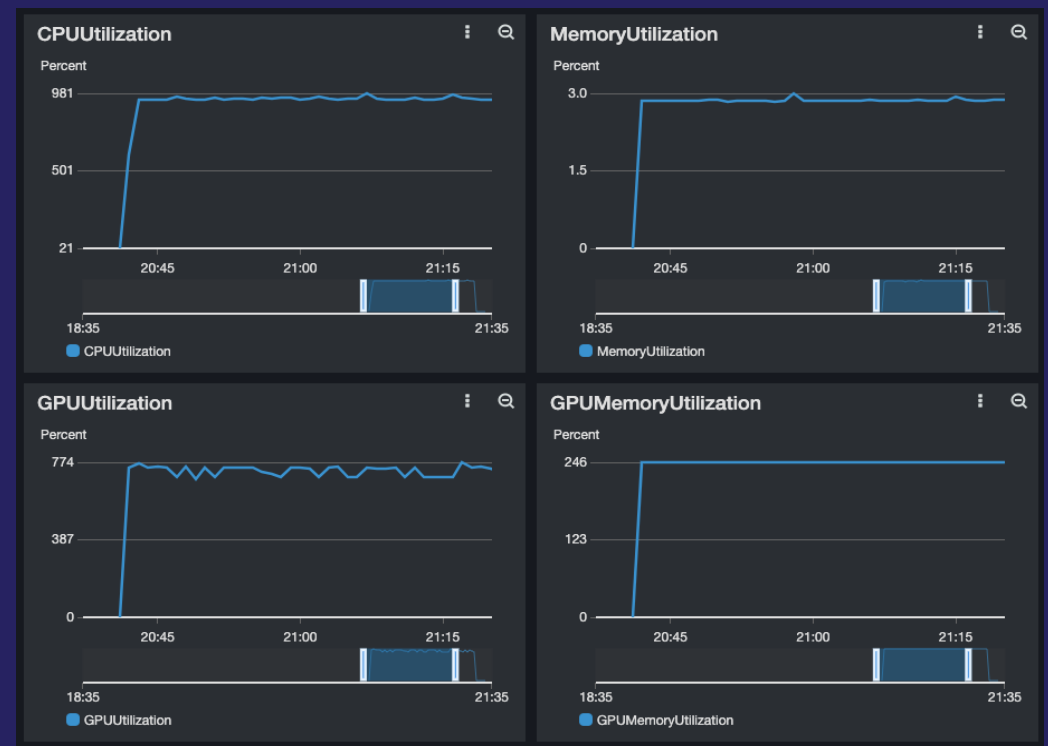
1. SageMaker Experiments
2. **학습 모니터링**

# SageMaker Training 모니터링 – CloudWatch

## 학습 코드의 출력 결과 확인

The screenshot displays the Amazon CloudWatch console interface. On the left, the navigation menu includes sections for Alarms, Logs, Metrics, X-Ray traces, Events, Application monitoring, and Insights. The 'Logs' section is selected, showing a list of log groups under the 'Log groups' heading. The main panel shows the 'Log events' view for a specific log group. It includes a search bar, a filter bar, and a list of log events. The events are displayed in a table format, showing the message content for each event. The messages include training progress information such as loss, lr, and step numbers.

## GPU/CPU 리소스 사용량 확인

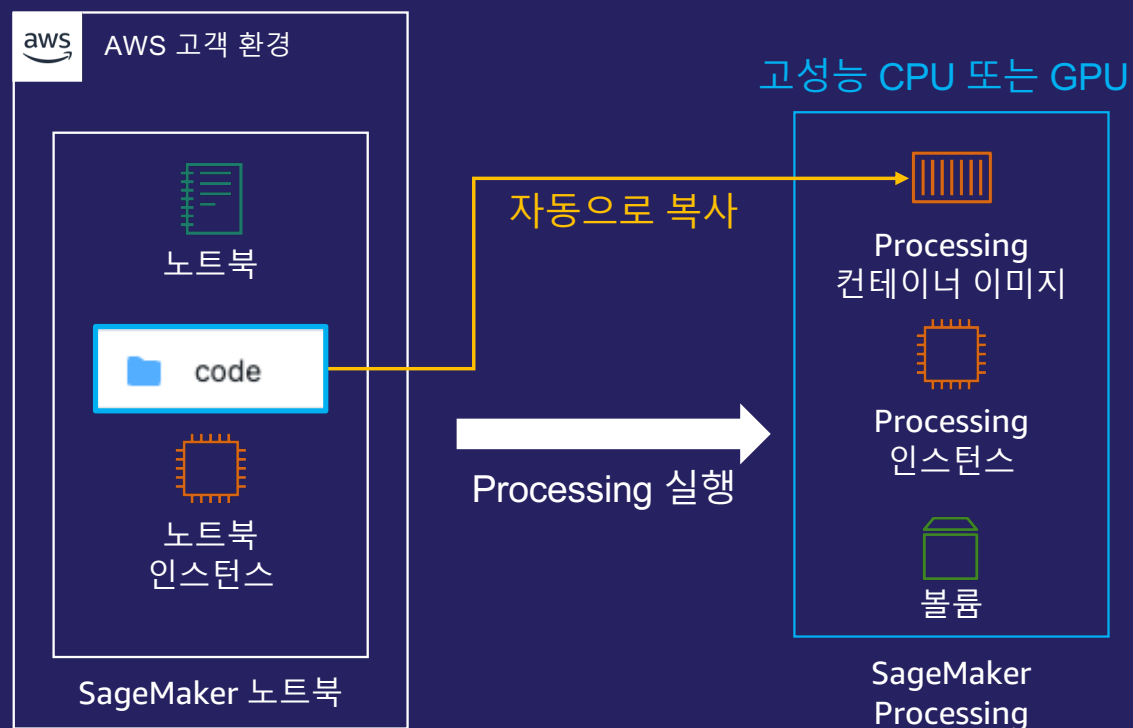


# SageMaker Processing - 데이터/결과 처리



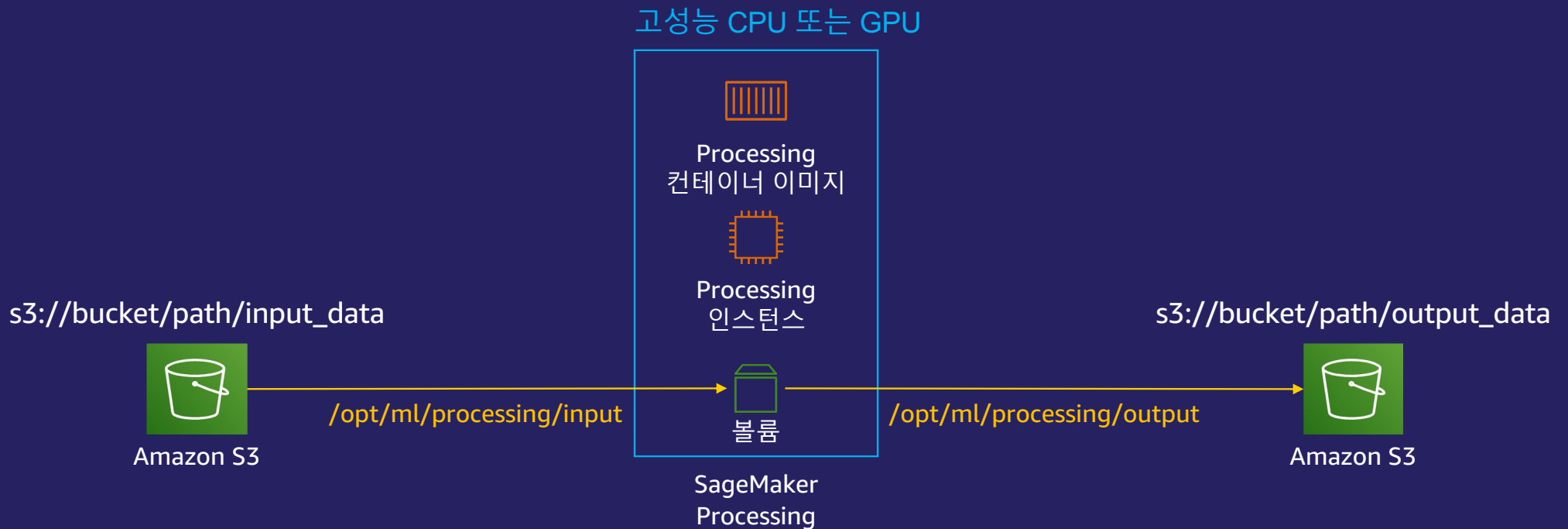
# SageMaker Processing 실행

사전 처리, 후 처리 및 모델 평가를 실행할 수 있는 환경 제공



# SageMaker Processing - S3와의 관계

S3의 데이터를 입력으로 받아 로직 처리 후 S3에 출력으로 저장



## SageMaker Processing 실행 코드

```
# Built-in Scikit Learn Container or FrameworkProcessor
from sagemaker.sklearn.processing import SKLearnProcessor
from sagemaker.processing import Processor, ScriptProcessor, FrameworkProcessor

processor= FrameworkProcessor(PyTorch, framework_version="1.10",
                             role=role, instance_type='ml.g5.xlarge',
                             instance_count=1)

from sagemaker.processing import ProcessingInput, ProcessingOutput

processor.run(
    code='preprocessing.py',
    inputs=[ProcessingInput(source=INPUT_S3_URI, destination='/opt/ml/processing/input')],
    outputs=[ProcessingOutput(source='/opt/ml/processing/output/train', destination=OUTPUT_S3_URI_1),
             ProcessingOutput(source='/opt/ml/processing/output/validation',
                               destination=OUTPUT_S3_URI_2)]
)
```



# Thank you!



