Amazon SageMaker Bring Your Own Script/Container

Henrique Fugita

Sr. Prototyping Architect

Prototyping and Cloud Engineering

Immersion Day

aws machine learning

Model Options

In this session: Bring Your Own Script/Container

Training code

AWS Marketplace for Machine Learning

Amazon SageMaker AutoPilot

- XGBoost - Gradient Boosted Trees
- Matrix Factorization
- Regression
- Principal Component Analysis
- K-Means Clustering
- And More!

Built-in Algorithms (17)
No ML coding required

Bring Your Own Script
Amazon SageMaker builds the container
Open source containers

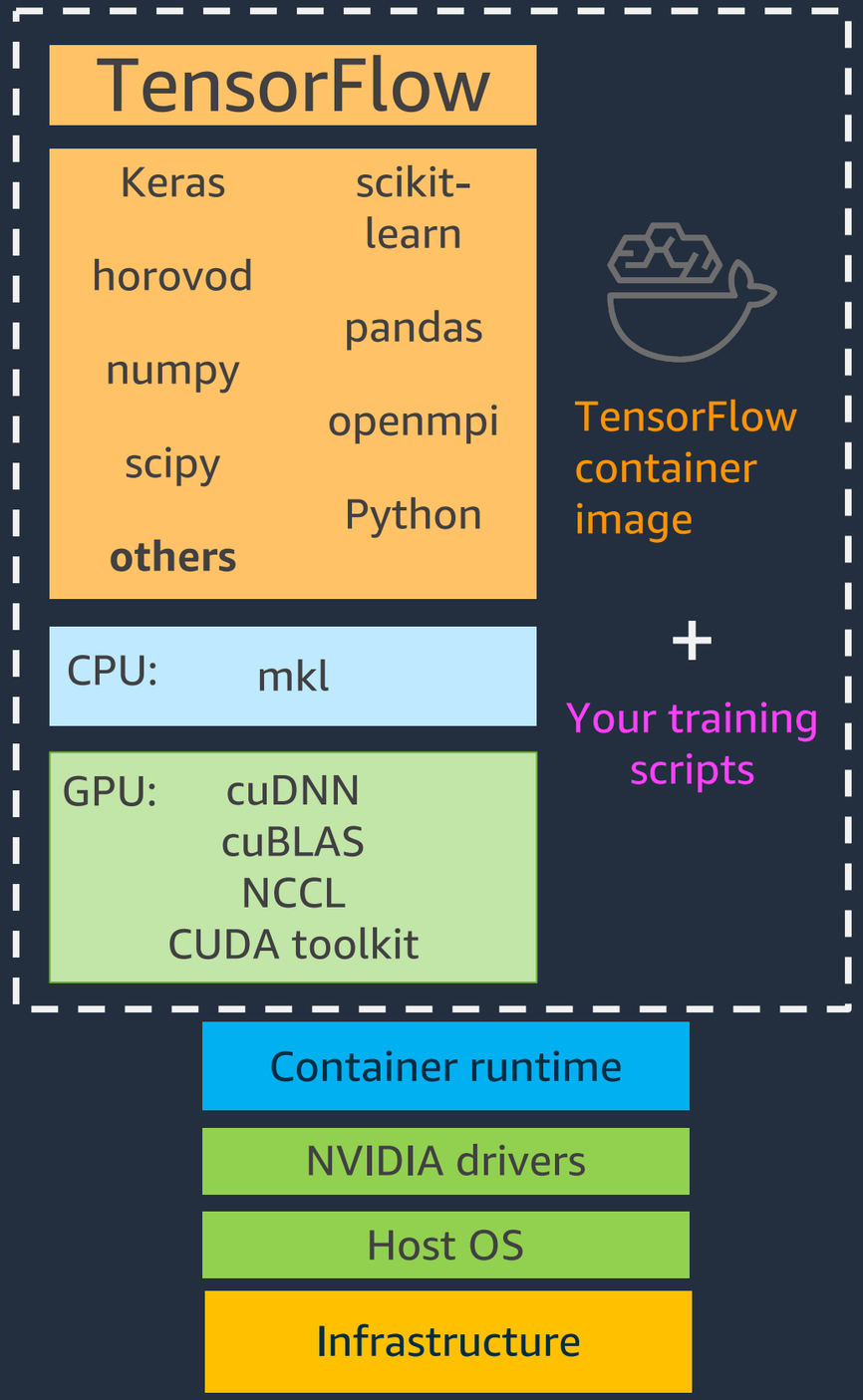
Bring Your Own Container
Full control, you build the container
R, C++, etc

Fully Managed, Distributed, Auto-Scaled, Secured

aws

Containers for Machine Learning

Containers for machine learning



ML environments that are:

- Lightweight
- Portable
- Scalable
- Consistent

Packages:

- Training code
- Dependencies
- Configurations

aws

TensorFlow

Keras
horovod
numpy
scipy
others

scikit-learn
pandas
openmpi
Python

CPU: mkl

GPU: cuDNN
cuBLAS
NCCL
CUDA toolkit

TensorFlow container image

+

Your training scripts

push

pull

Amazon ECR

Container registry

TensorFlow

Keras
horovod
numpy
scipy
others

scikit-learn
pandas
openmpi
Python

CPU: mkl

GPU: cuDNN
cuBLAS
NCCL
CUDA toolkit

TensorFlow container image

+

Your training scripts

Container runtime

NVIDIA drivers
Host OS

Development system

Container runtime

NVIDIA drivers
Host OS

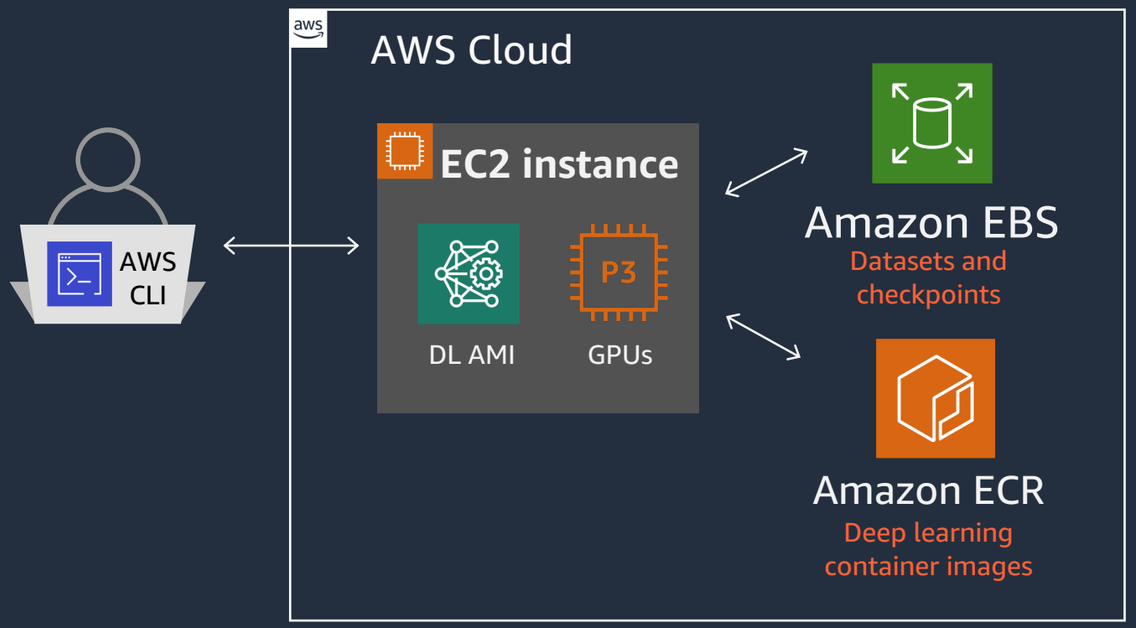
Training cluster

AWS Deep Learning Containers

- Prepackaged Docker container images fully configured and validated
- Optimized for performance with latest NVIDIA driver, CUDA libraries, and Intel libraries
- Consistent and reproducible deployment and lightweight
- Optimized for distributed machine learning
- Runs on Amazon ECS, Amazon EKS and Amazon SageMaker

aws

Machine Learning Setups on AWS Today



Code and dependencies

Collaborative development

Performance optimizations

Scaling

Infrastructure management

ML Infrastructure and Cluster Management

ML services

Fully managed service that covers the entire machine learning workflow

Amazon SageMaker


Jupyter notebook instances


High-performance algorithms


Large-scale training


Optimization


One-click deployment


Fully managed with auto scaling

Management

Deployment, scheduling, scaling, and management of containerized applications


Amazon Elastic Container Service (Amazon ECS)


Amazon Elastic Kubernetes Service (Amazon EKS)

Compute

Where the containers run


Amazon EC2

Image registry

Container image repository


Amazon Elastic Container Registry (Amazon ECR)

aws

3 Ways to Train Using Amazon SageMaker

Use built-in algorithms

(Bring Your Own Data)

- K-Means Clustering
- Principal Component Analysis
- Neural Topic Modelling
- Factorization Machines
- Linear Learner (Regression)
- BlazingText
- Reinforcement learning
- XGBoost
- Topic Modeling (LDA)
- Image Classification
- Seq2Seq
- Linear Learner (Classification)
- DeepAR Forecasting

Use deep learning frameworks

(Bring Your Own Data)

(Bring your own training script)

TensorFlow

Keras

mxnet

GLUON

HOROVOD

PYTORCH

Chainer

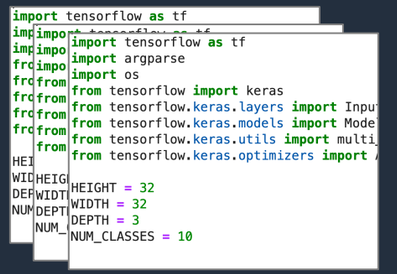
ONNX

Use custom containers

(Bring your own data)

(Bring your own container)

Custom container

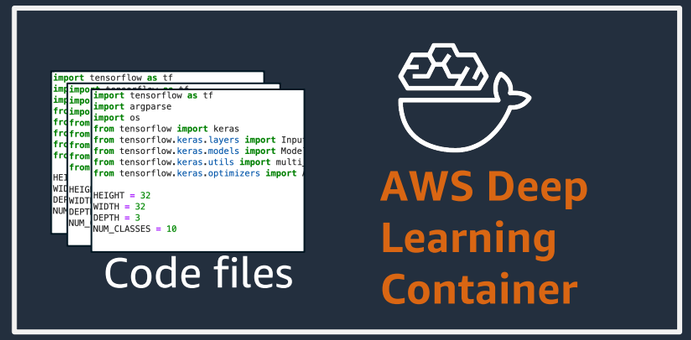


Code files

aws

Amazon SageMaker
Bring your own Script

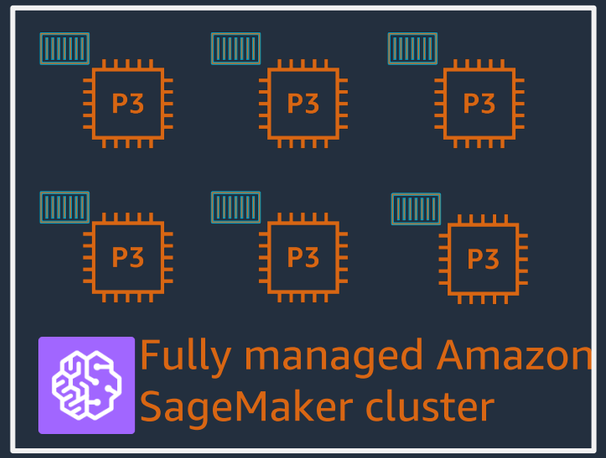
High Level Workflow



Container registry

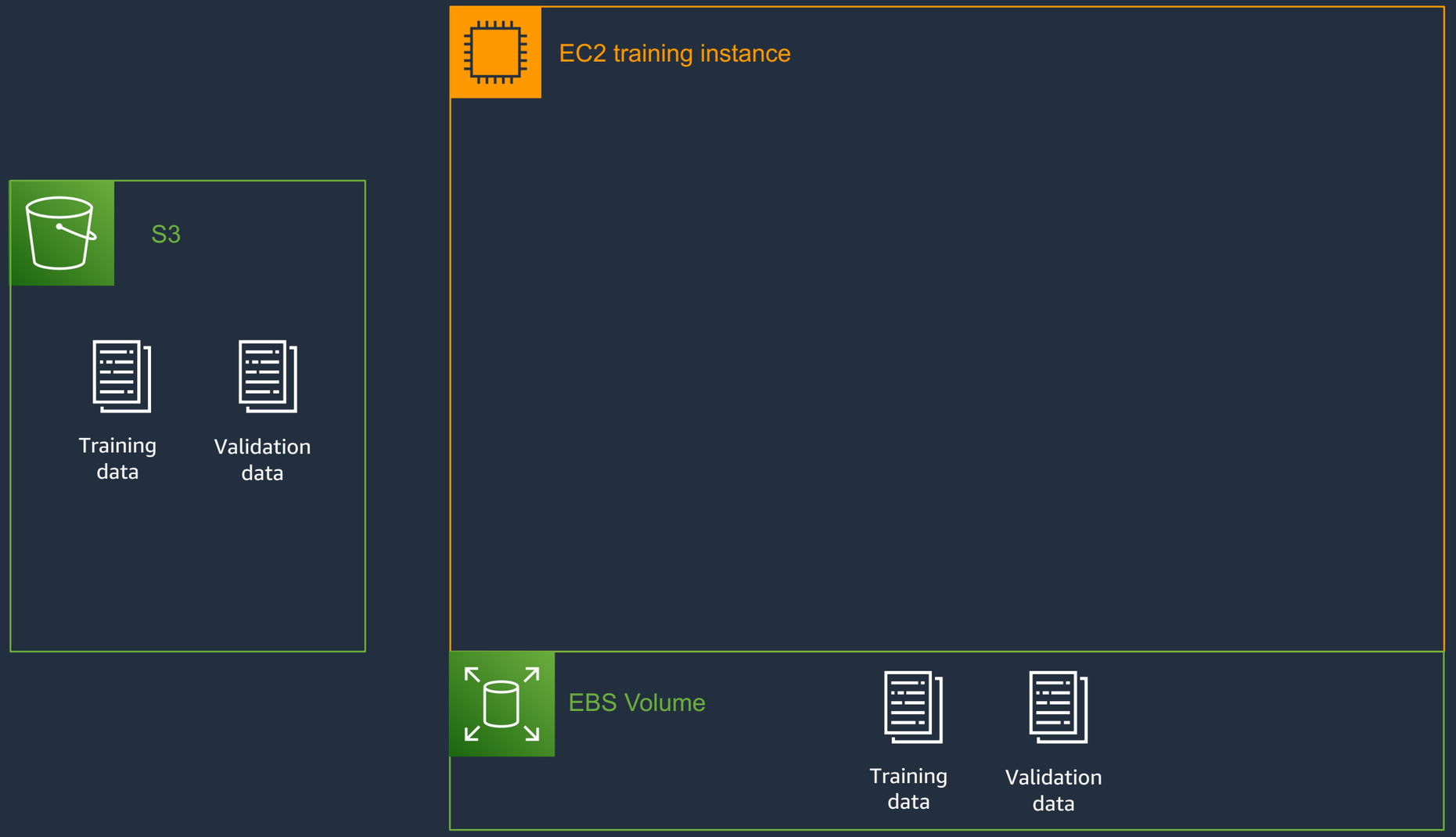
```
from sagemaker.tensorflow import TensorFlow

cifar10_estimator = TensorFlow(entry_point='source/cifar10.py',
                               role=role,
                               framework_version='1.14',
                               train_instance_count=1,
                               train_instance_type='ml.p3.xlarge')
```

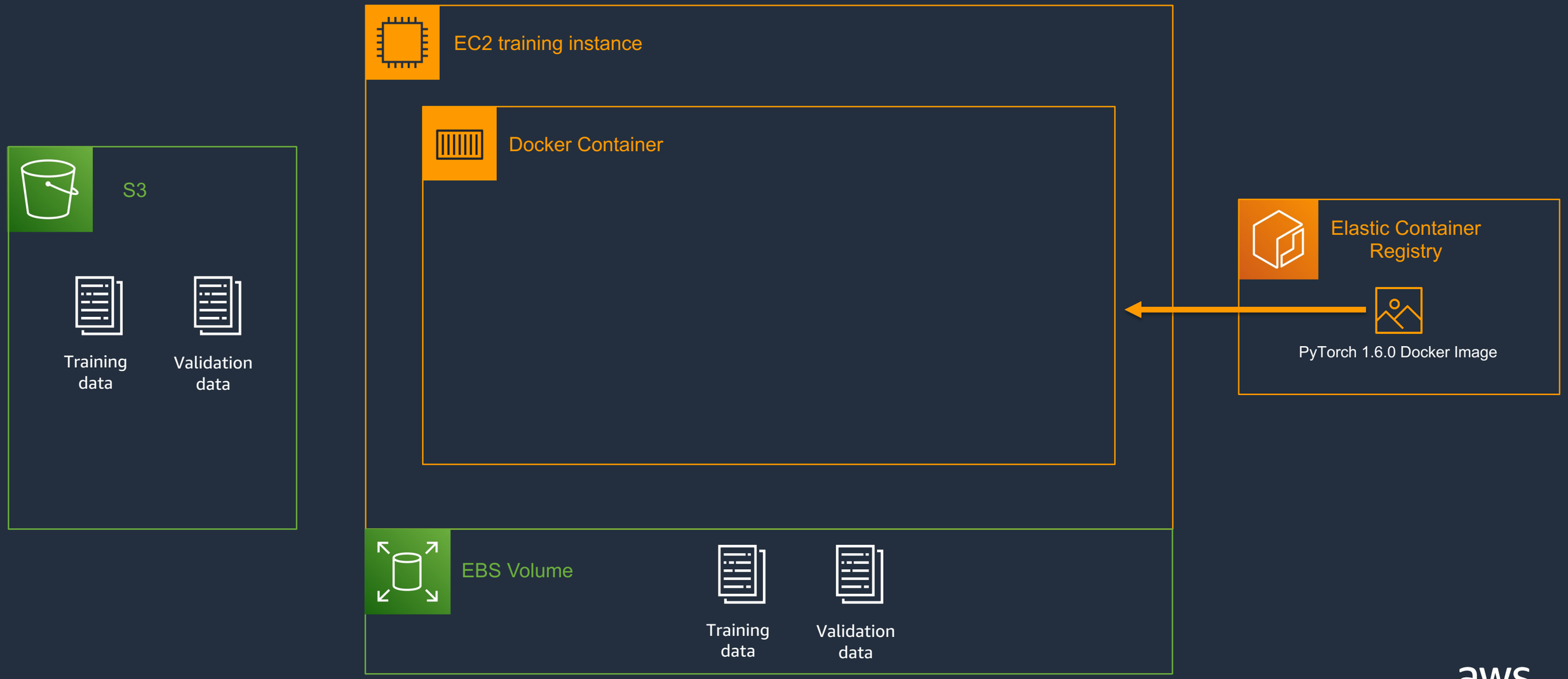


aws

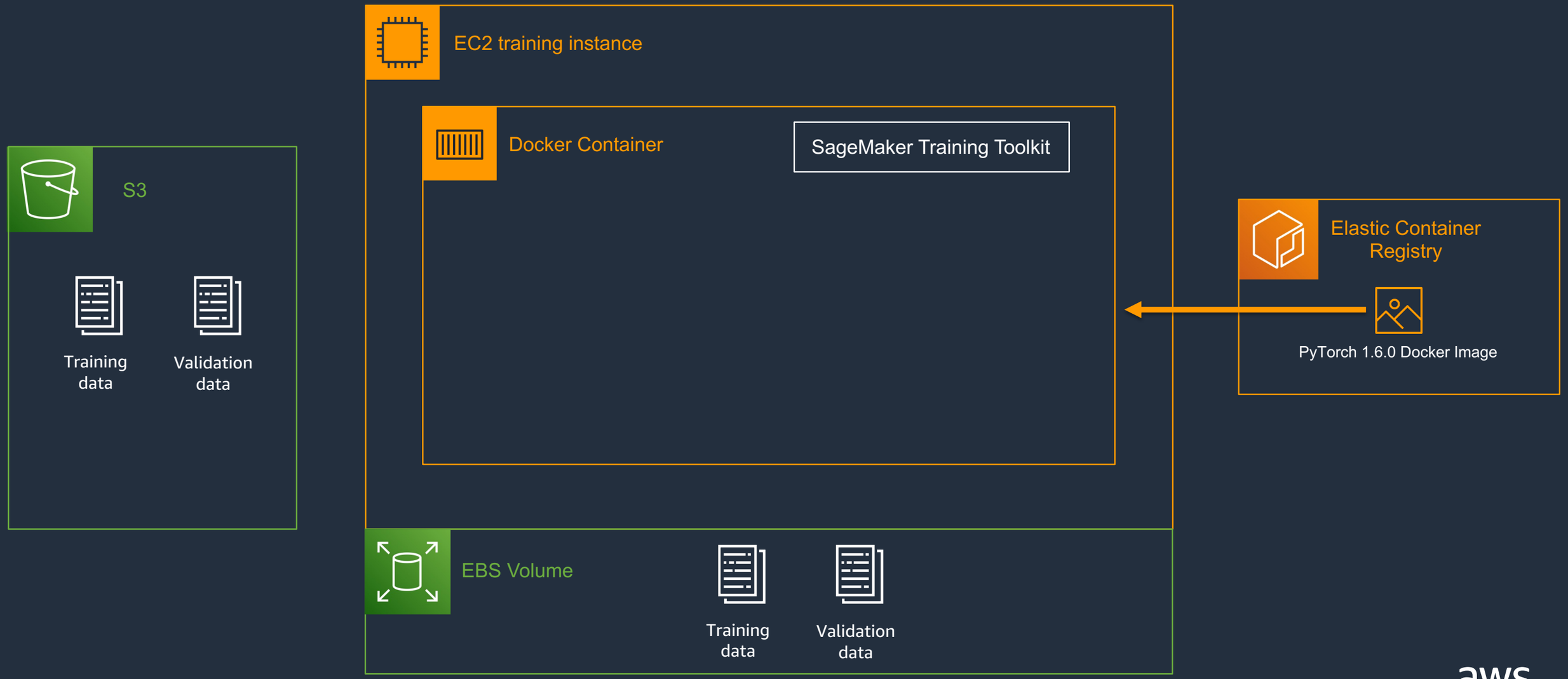
Copy the data from S3 to EBS volume of the EC2 instance



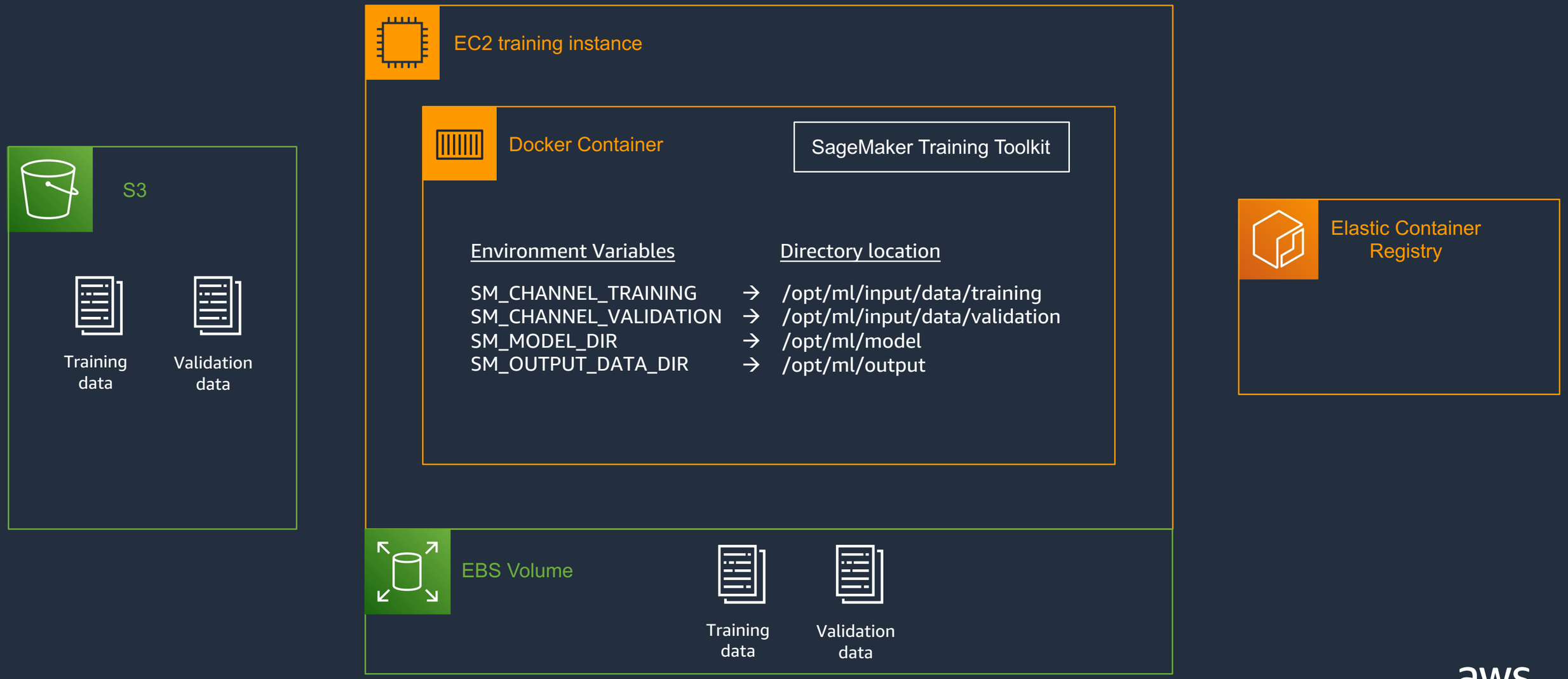
Docker Container



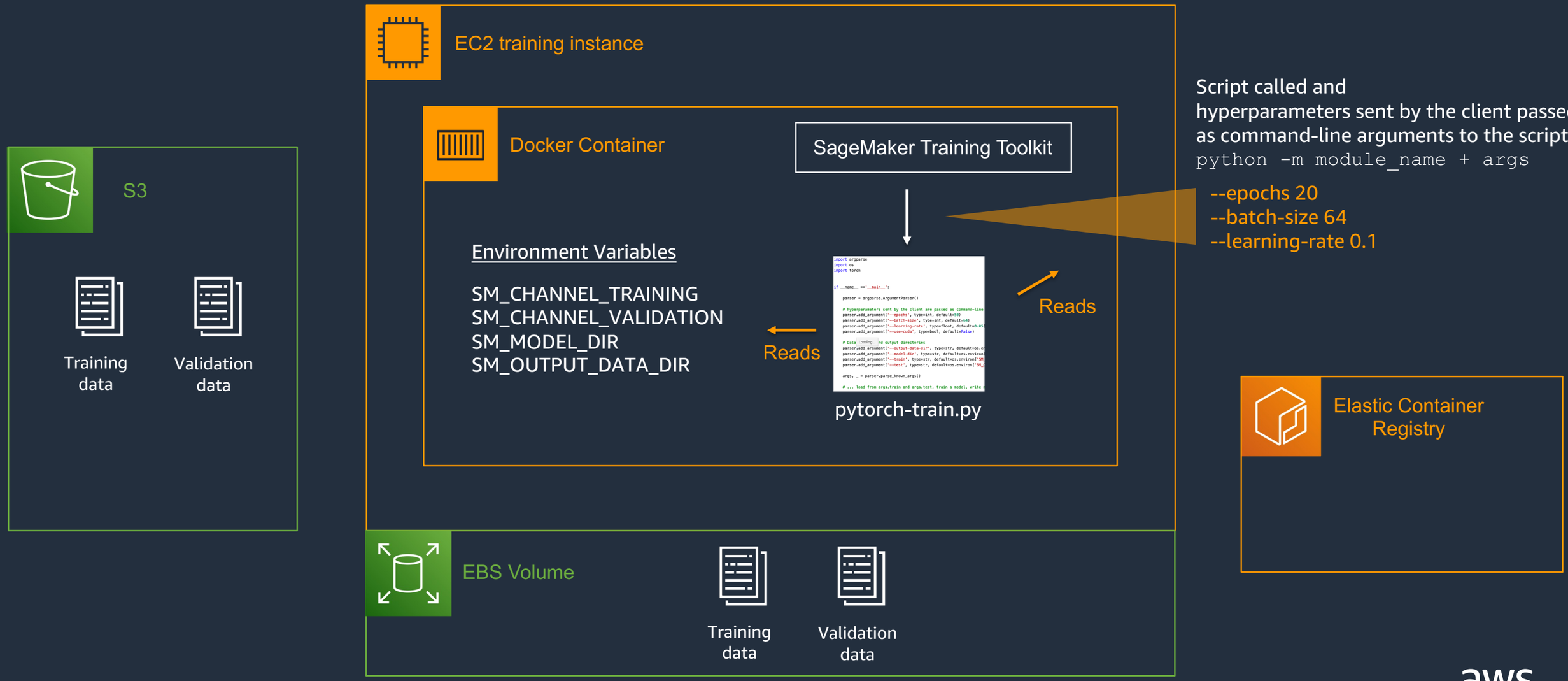
SageMaker Training Toolkit



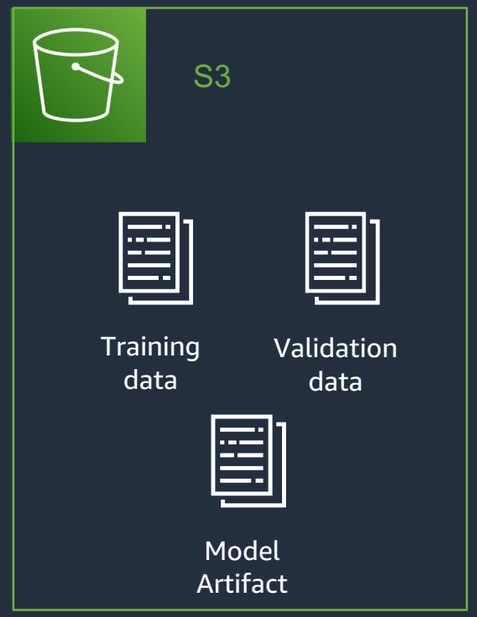
Environment variables and directory location

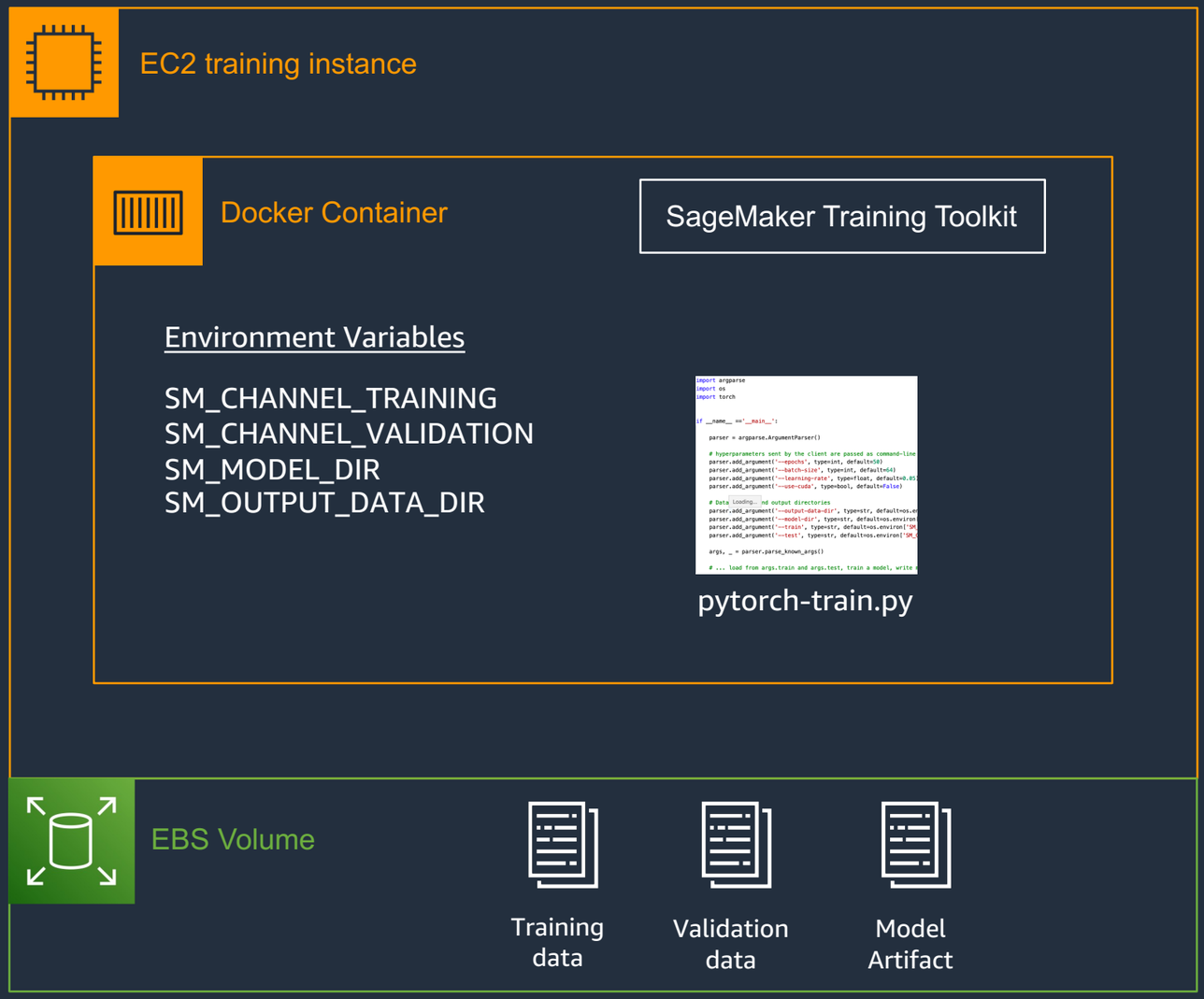


Your script is called with arguments present in the Estimator



Copy of the Model Artifact to S3 after training



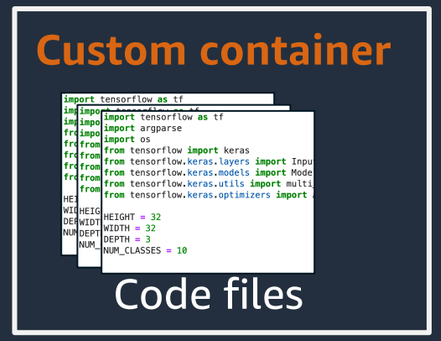


aws

Amazon SageMaker
Bring your own Container

High Level Workflow

Docker build



Container registry

```
ic = sagemaker.estimator.Estimator(training_image,
                                   role,
                                   train_instance_count=1,
                                   train_instance_type='ml.p3.xlarge',
                                   train_volume_size = 50,
                                   train_max_run = 360000,
                                   input_mode= 'File',
                                   output_path=s3_output_location,
                                   sagemaker_session=sess)
```

Bring Your Own Docker File

Customer Managed

1. Write your model however you please
2. Point to your model within your Docker file
3. Register your container on ECR
4. Point to your container’s address in ECR
5. Don’t forget to implement a serve() function!

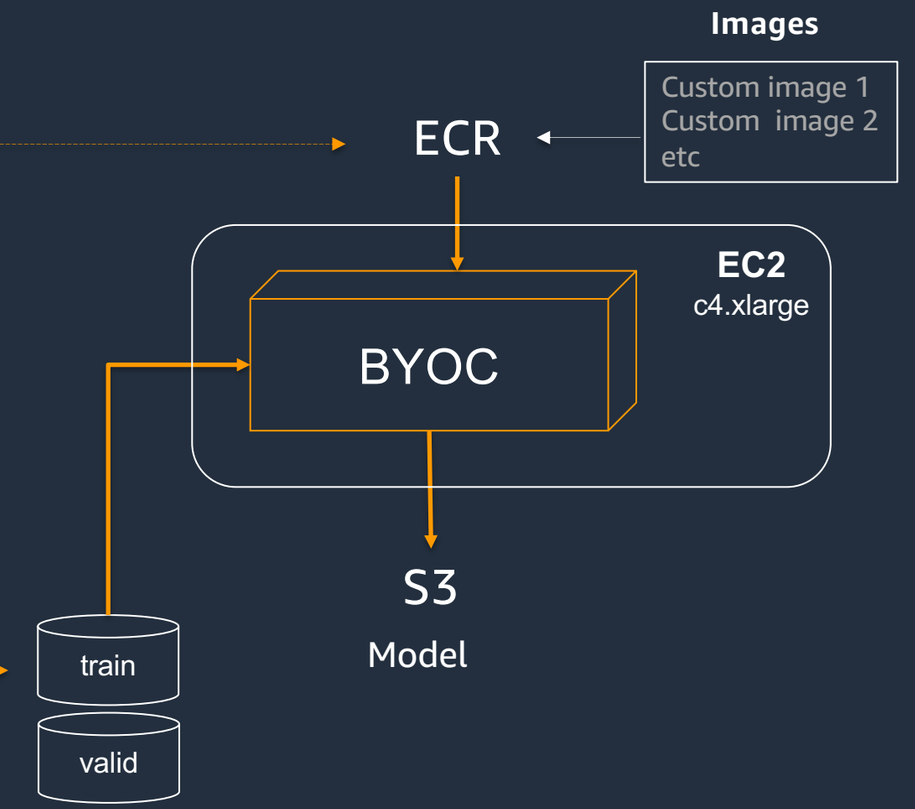
aws

Amazon SageMaker | Training

Bring your own container

```
estimator = Estimator(image='custom_container',
                      train_instance_type='ml.c4.xlarge',
                      train_instance_count=1,
                      hyperparameters=hyperparameters,
                      py_version='py3')
```

```
estimator.fit({'training': inputs})
```

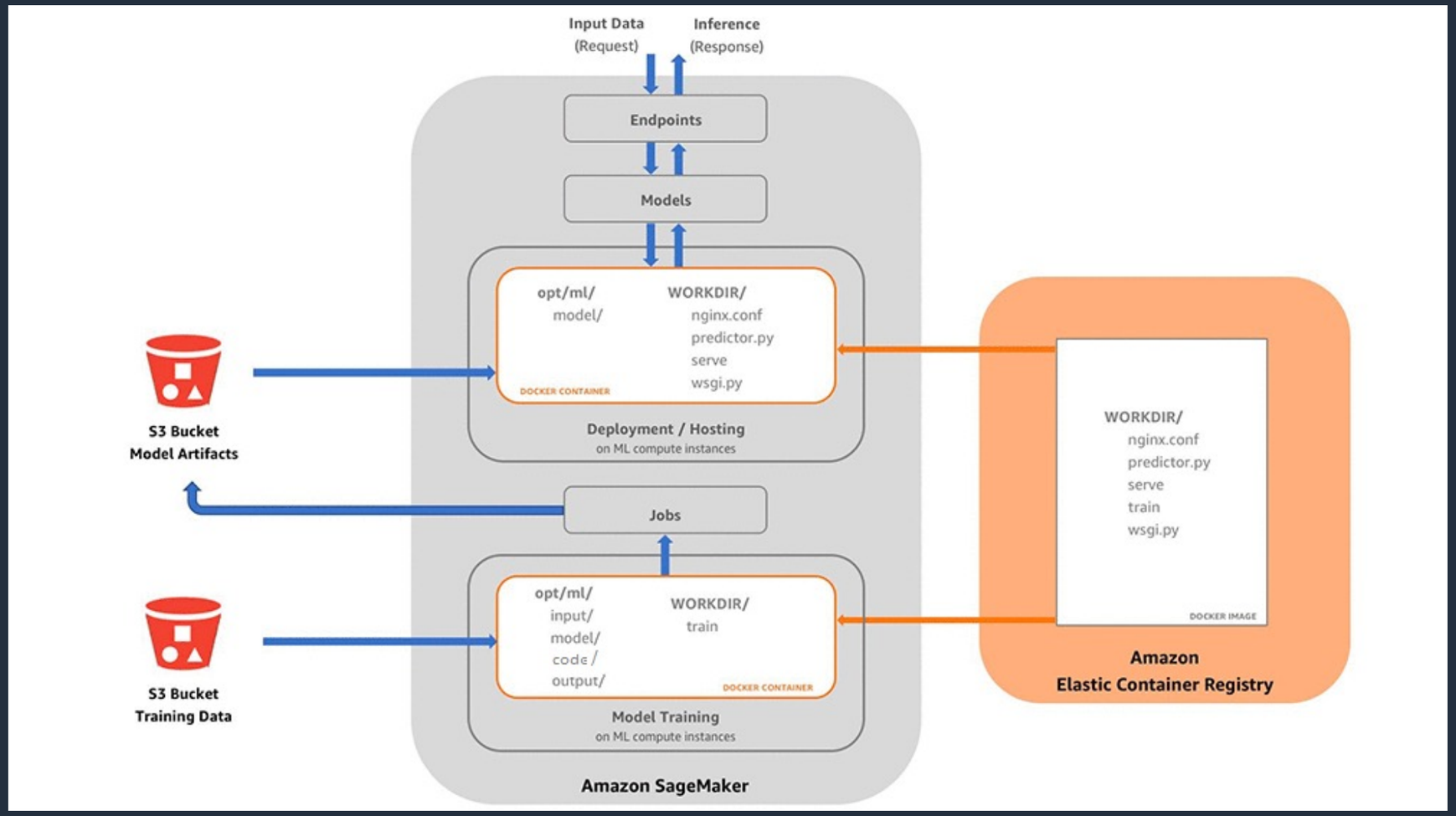


aws

Bring Your Own Container

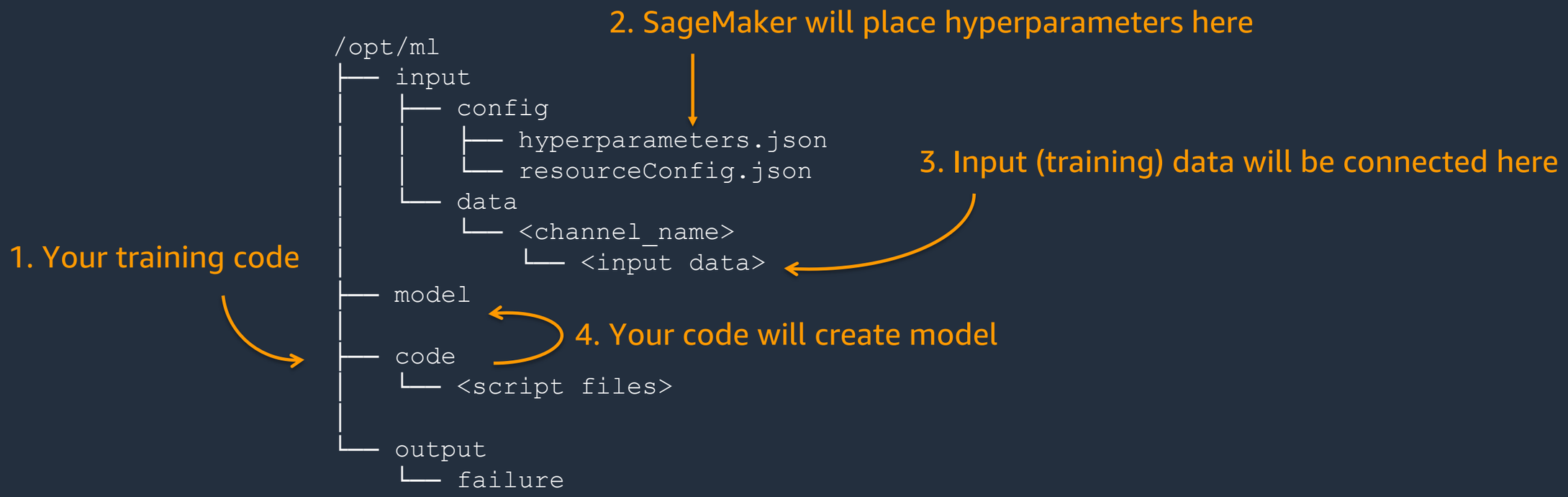
Custom Docker image:

- Training image
- Inference (serving) image



Amazon SageMaker | Training

Bring your own container



5. Final model from /opt/ml/model will be saved to S3 at the end of training session, before the container is destroyed.

Takeaways

Code and dependencies

Collaborative development

Performance optimizations

Scaling

Infrastructure management

- Containers let you build lightweight, portable and consistent ML environments
- AWS DL containers include frameworks optimized by experts to deliver the best performance on CPUs and GPUs
- Leverage Amazon ECR along with git for collaborative development
- Leverage Amazon SageMaker to manage large-scale ML workloads

aws

Thank you!