Bring Your Own Model

Machine Learning Immersion Day

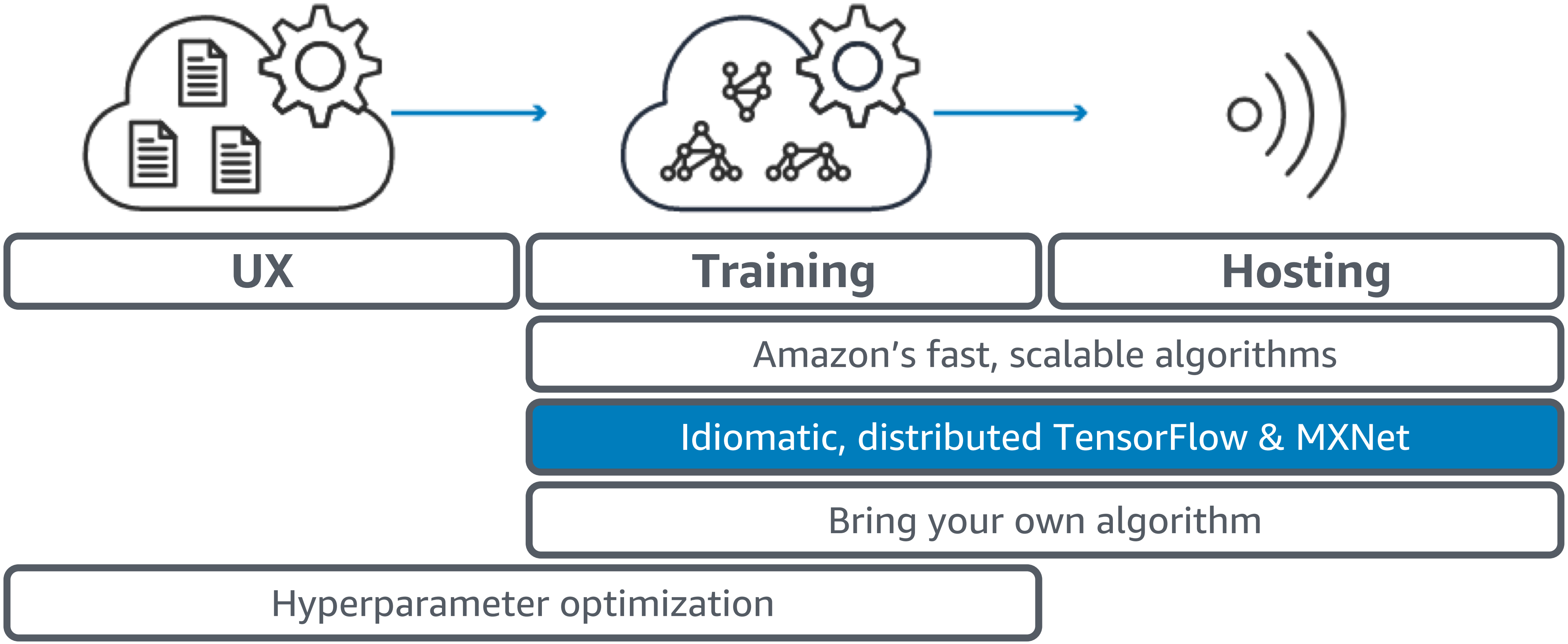
Module 3

Agenda

09:00 Welcome and Introductions
09:15 Data Science Lifecycle
10:30 Lab 1: Understanding and preparing data with S3, Glue and Athena
11:15 Lunch
12:00 Model training, testing and deploying with Sagemaker
12:45 Lab 2: Train, test and deploy your first model with Sagemaker
13:45 Break
14:00 Continuous Delivery of ML models
14:30 Lab 3: Continuous Delivery of ML models to Amazon SageMaker

15:15 Bring your own model
15:45 Wrap Up

Amazon SageMaker components - Frameworks



UX

Training

Hosting

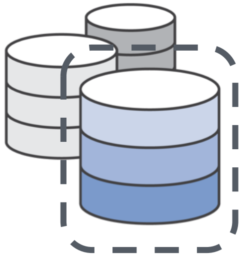
Amazon’s fast, scalable algorithms

Idiomatic, distributed TensorFlow & MXNet

Bring your own algorithm

Hyperparameter optimization

Frameworks



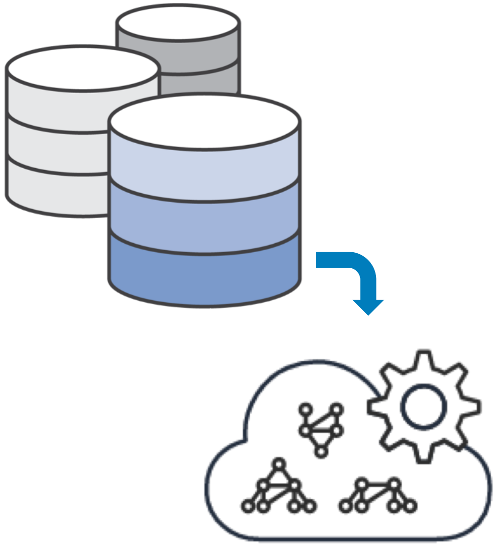
Sample your data…

TensorFlow™

mxnet

GLUON

… explore and refine models in a single Notebook Instance



Use the same code to train on the full dataset in a cluster of GPU instances…

… deploy to production

Tensorflow example

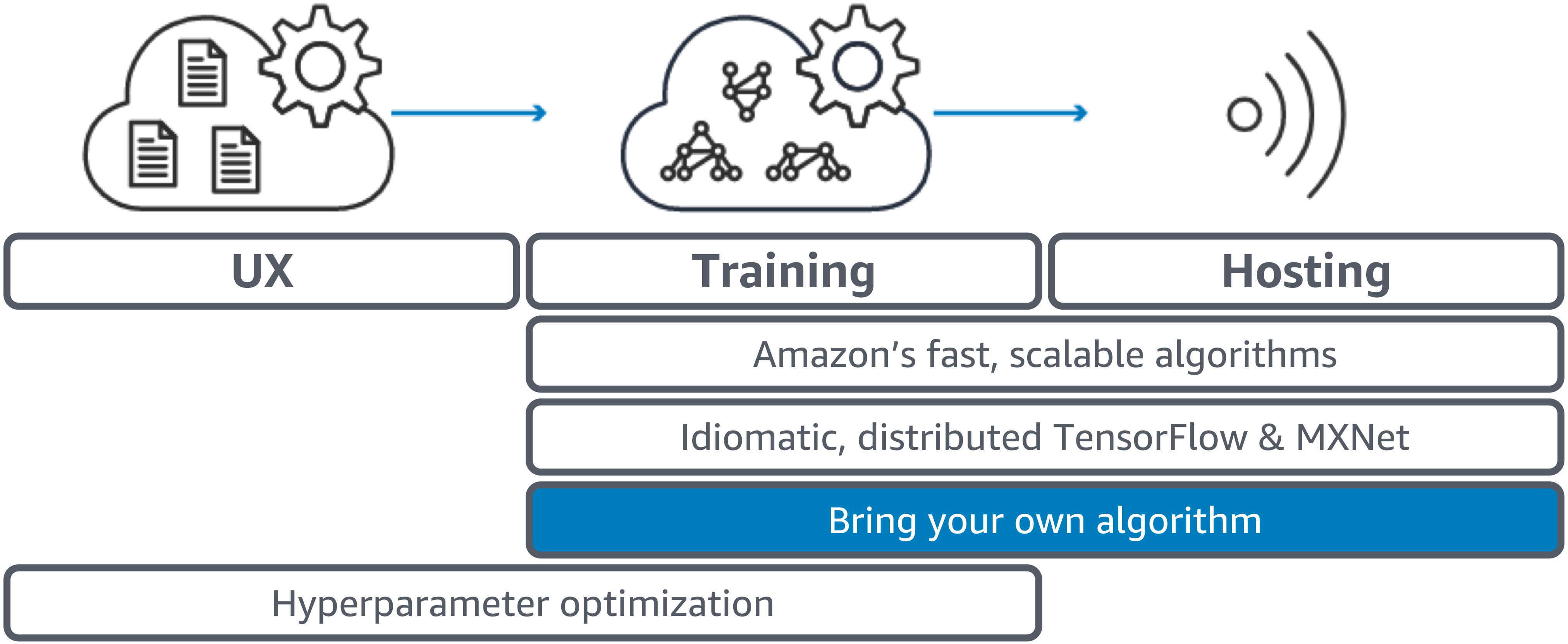
Use `sagemaker.tensorflow` package to invoke your custom Tensorflow code, provided in a script containing specific functions:

- Exactly one of the following:
  - `model_fn`: defines the model that will be trained.
  - `keras_model_fn`: defines the tf.keras model that will be trained.
  - `estimator_fn`: defines the tf.estimator.Estimator that will train the model.
- `train_input_fn`: preprocess and load training data.
- `eval_input_fn`: preprocess and load evaluation data.

Tensorflow example

```
from sagemaker.tensorflow import TensorFlow
tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole’,
                            training_steps=10000, evaluation_steps=100,
                            train_instance_count=1, train_instance_type='ml.p2.xlarge')
tf_estimator.fit('s3://bucket/path/to/training/data’)

# ----------------

def model_fn(features, labels, mode, hyperparameters):
  # Logic to do the following:
  # 1. Configure the model via TensorFlow operations
  # 2. Define the loss function for training/evaluation
  # 3. Define the training operation/optimizer
  # 4. Generate predictions
  # 5. Return predictions/loss/train_op/eval_metric_ops in EstimatorSpec object
  return EstimatorSpec(mode, predictions, loss, train_op, eval_metric_ops)

# ----------------

def train_input_fn(training_dir, hyperparameters):
 # Logic to do the following:
 # 1. Reads the **training** dataset files located in training_dir
 # 2. Preprocess the dataset
 # 3. Return 1) a dict of feature names to Tensors with
 # the corresponding feature data, and 2) a Tensor containing labels
 return features, labels
```

Amazon SageMaker components - BYOM



UX

Training

Hosting

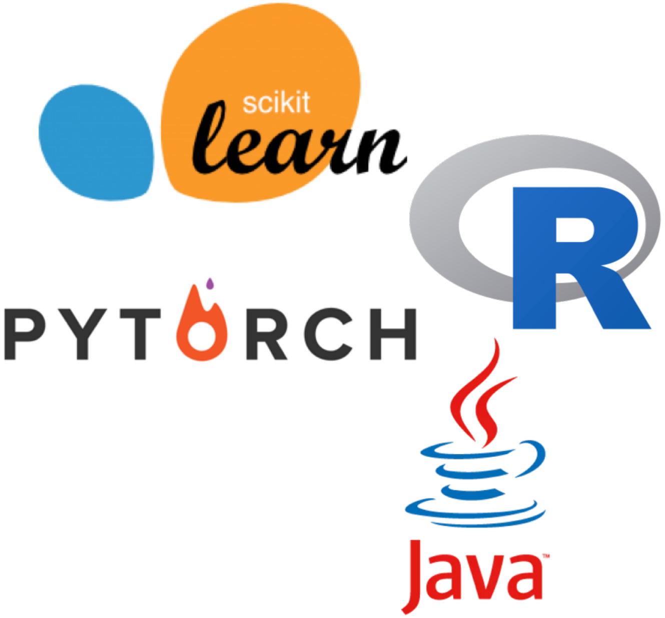
Amazon's fast, scalable algorithms

Idiomatic, distributed TensorFlow & MXNet

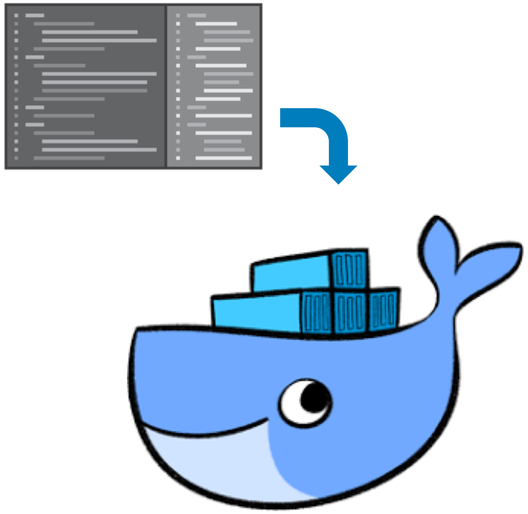
Bring your own algorithm

Hyperparameter optimization

Bring Your Own Model



Pick your preferred framework...

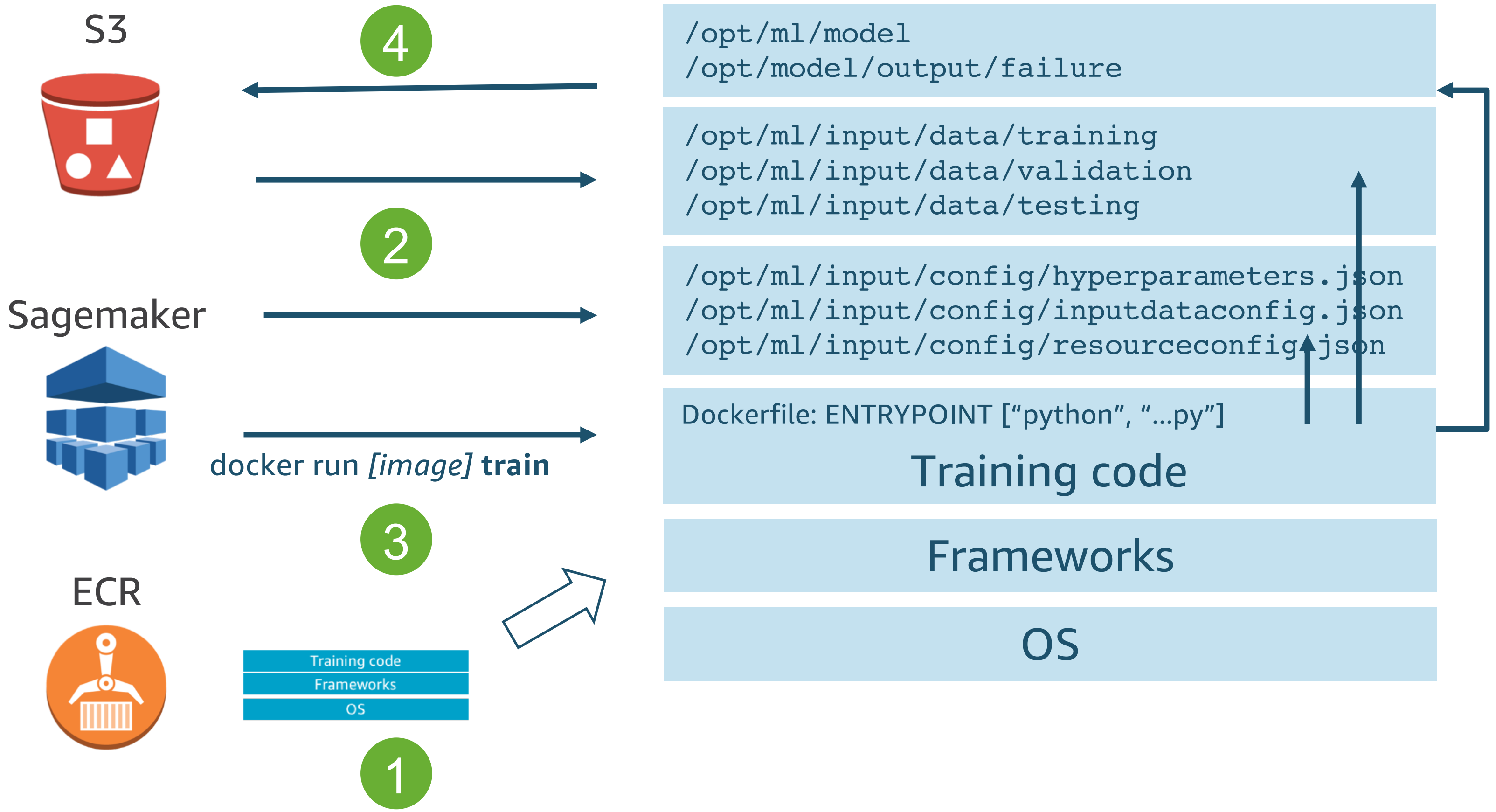


... add algorithm code to a Docker container...

Amazon ECS

... publish to ECS

BYOM - Training



S3

4

2

Sagemaker

docker run *[image]* **train**

3

ECR

Training code

Frameworks

OS

1

```
/opt/ml/model
/opt/model/output/failure
```

```
/opt/ml/input/data/training
/opt/ml/input/data/validation
/opt/ml/input/data/testing
```

```
/opt/ml/input/config/hyperparameters.json
/opt/ml/input/config/inputdataconfig.json
/opt/ml/input/config/resourceconfig.json
```

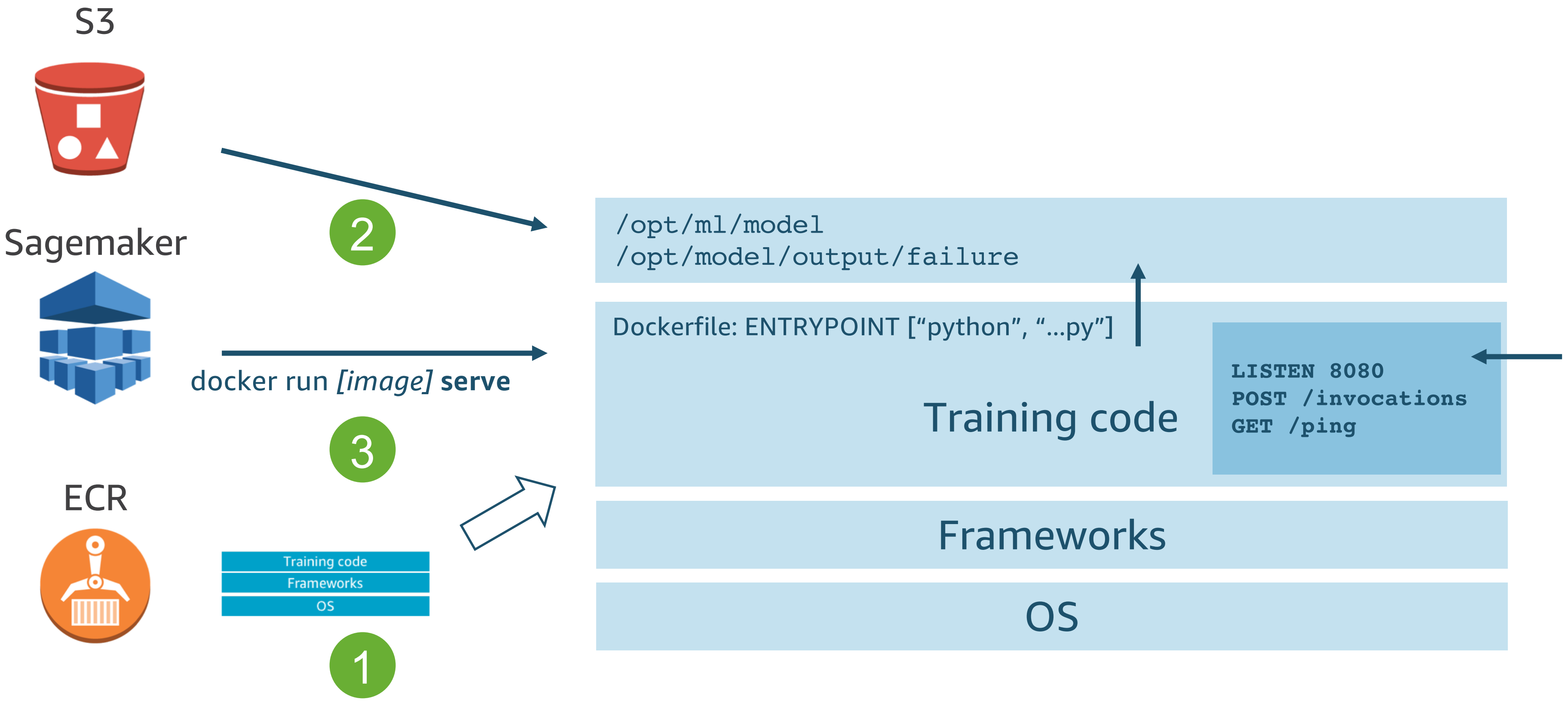
Dockerfile: ENTRYPOINT [“python”, “…py”]

Training code

Frameworks

OS

BYOM - Inference



S3

Sagemaker

ECR

docker run *[image]* **serve**

/opt/ml/model
/opt/model/output/failure

Dockerfile: ENTRYPOINT [“python”, “...py”]

Training code

```
LISTEN 8080
POST /invocations
GET /ping
```

Frameworks

OS

Questions