
aws
INNOVATE
ONLINE CONFERENCE

aws INNOVATE
ONLINE CONFERENCE

Open-source ML frameworks on Amazon SageMaker

Pedro Paez
Specialist Solutions Architect, AWS

Deep learning frameworks on AWS

Pre-configured environments to quickly build deep learning applications

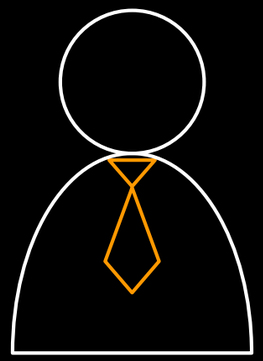
aws INNOVATE
ONLINE CONFERENCE

AWS is framework agnostic

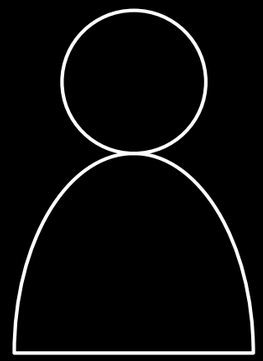
Choose from popular frameworks

TensorFlow mxnet PYTORCH Chainer Caffe2 ONNX Keras GLUON

Run them fully managed



Or run them yourself



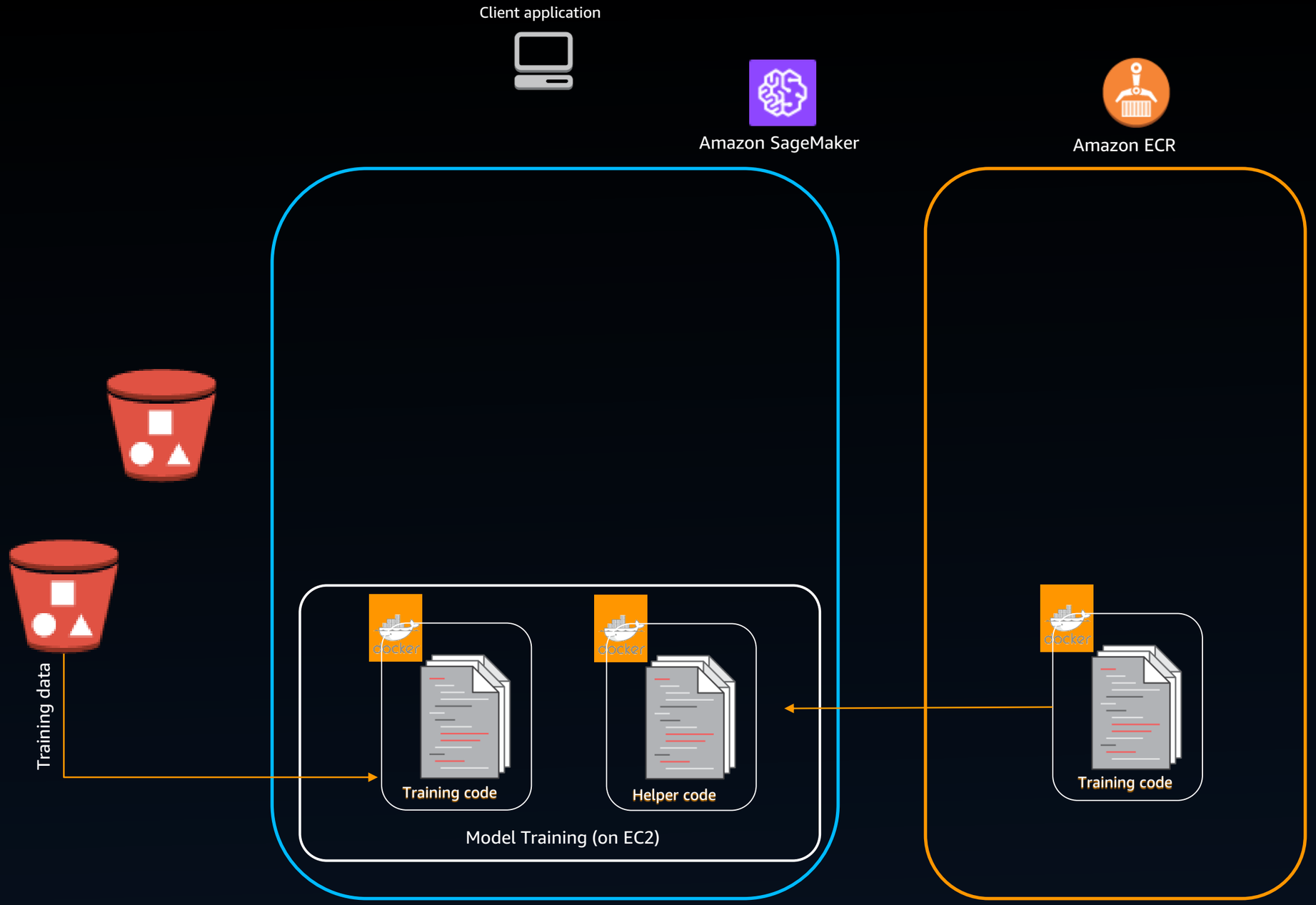
aws | intel

Deep learning on Amazon SageMaker

- Built-in support for TensorFlow, PyTorch, Chainer, MXNet
- Fully customizable Bring Your Own (BYO) container option
- Distributed GPU training
- Code portability
- Experiment tracking
- One-click training and one-click deployment

You provide

1. Your data in Amazon S3
2. A base docker image, Amazon SageMaker built-in or BYO
3. Your source code in Amazon S3 or local file system
4. The Amazon EC2 instance type and count



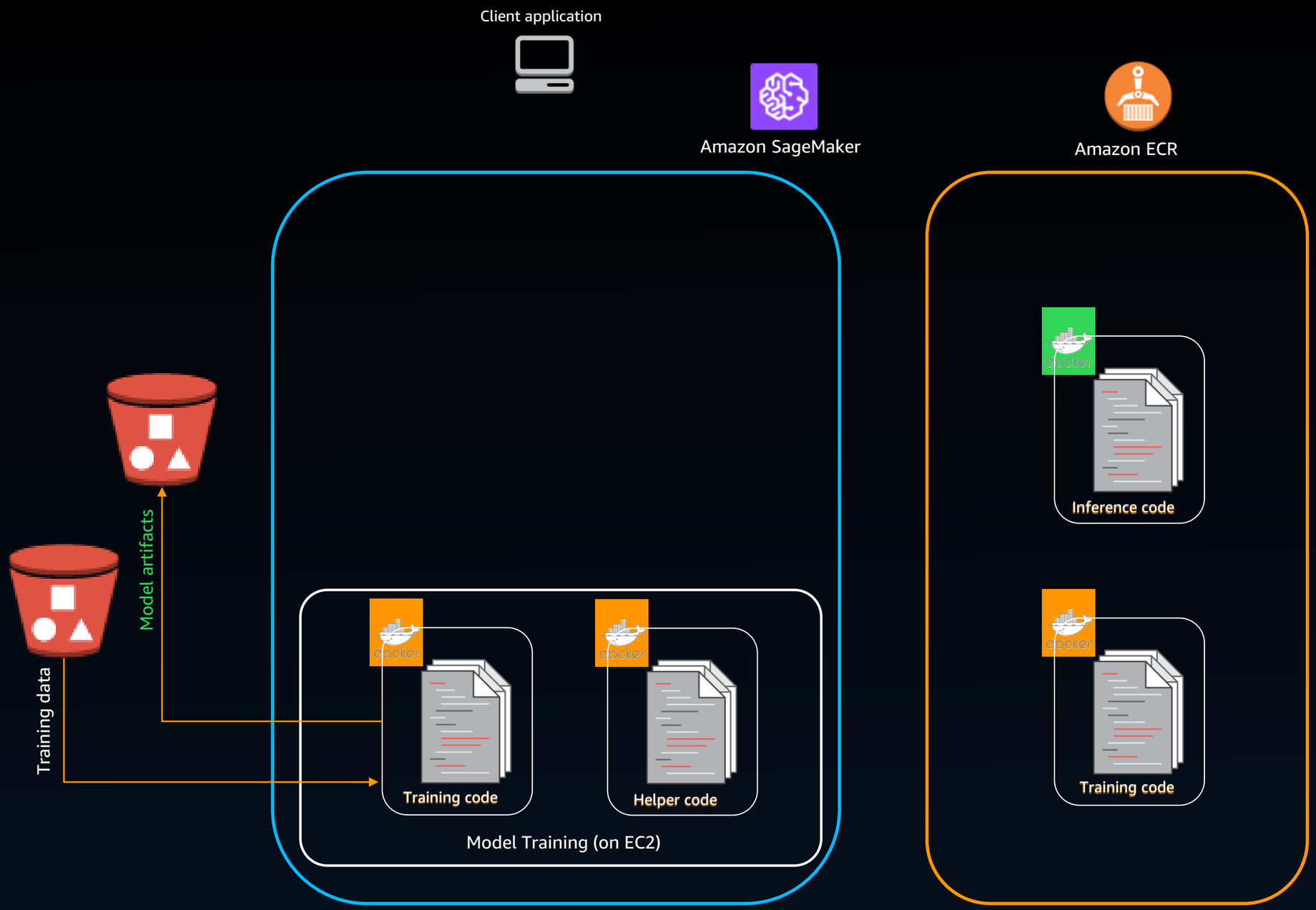
Amazon SageMaker

✓Automatically provisions and tears down Amazon EC2 instances for your training job

✓Downloads code & data from Amazon S3 to container host

✓Launches docker image in the Amazon EC2 instance type required for training & inference

✓Executes training job



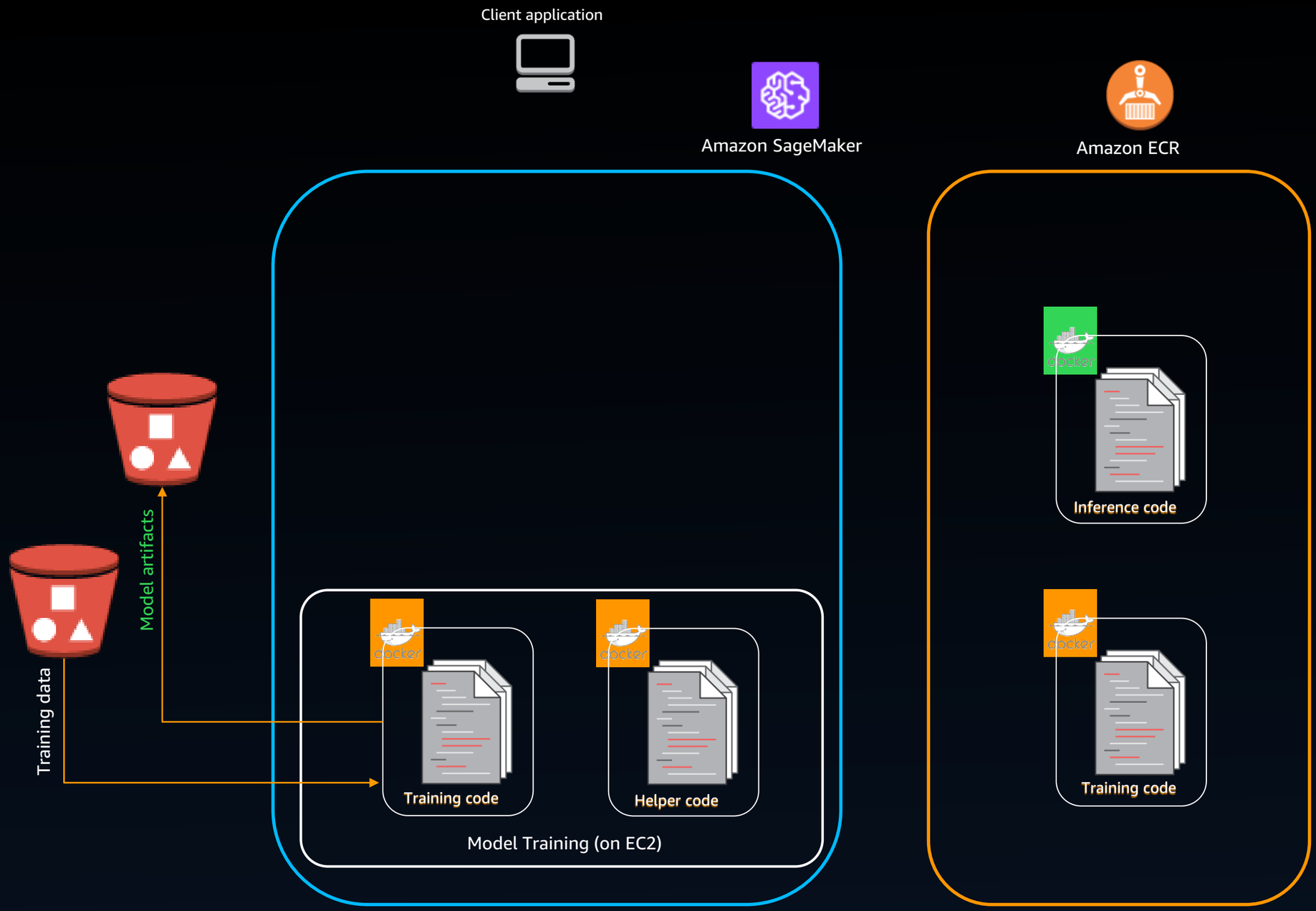
Amazon SageMaker

✓Copies model from container host to Amazon S3

✓Copies results from container host to Amazon S3

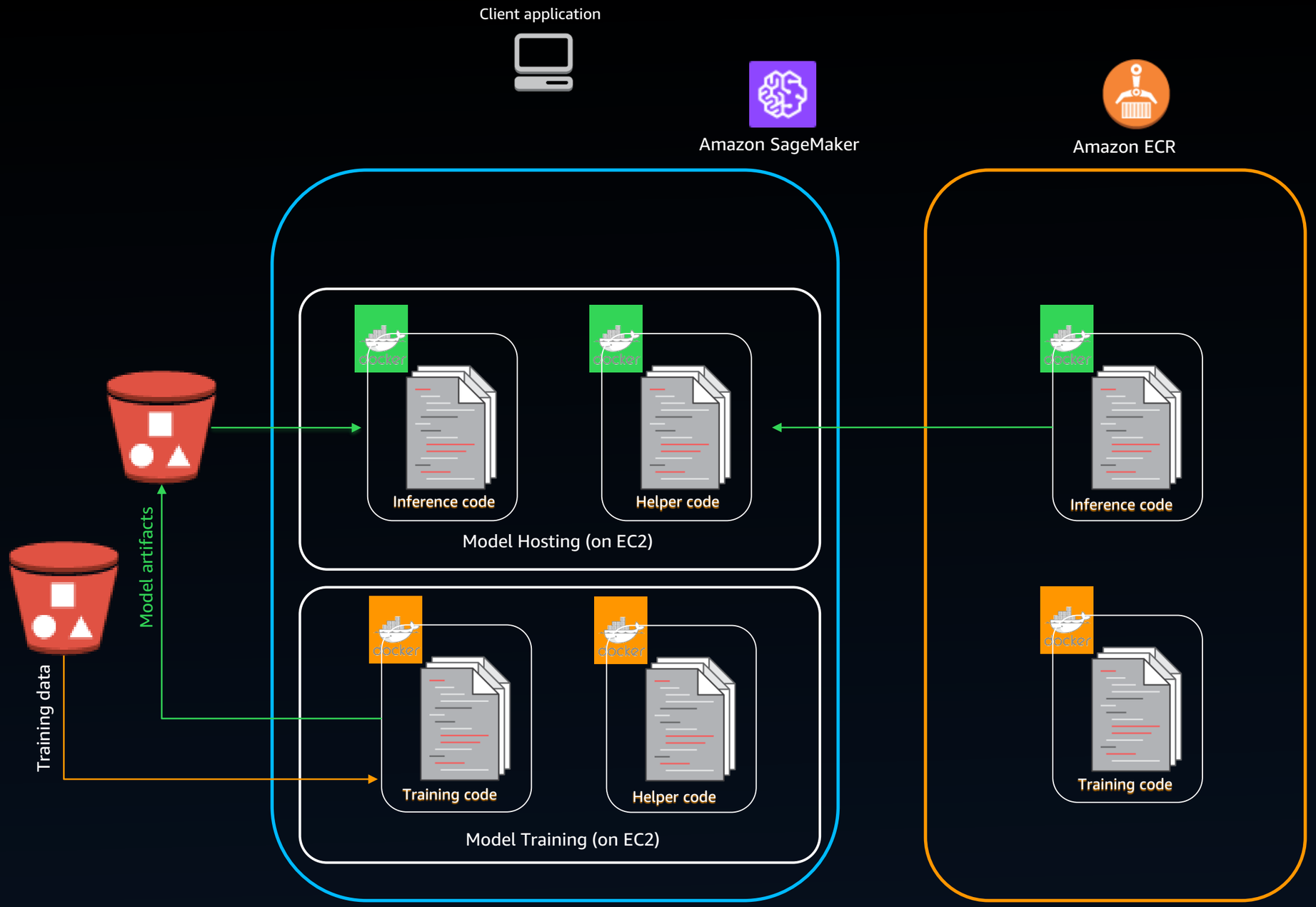
✓Logs / print results captured in Amazon CloudWatch

✓Captures training infrastructure & performance metrics in Amazon CloudWatch



You provide

1. The trained model
2. The docker image
3. The Amazon EC2 instance type and initial count

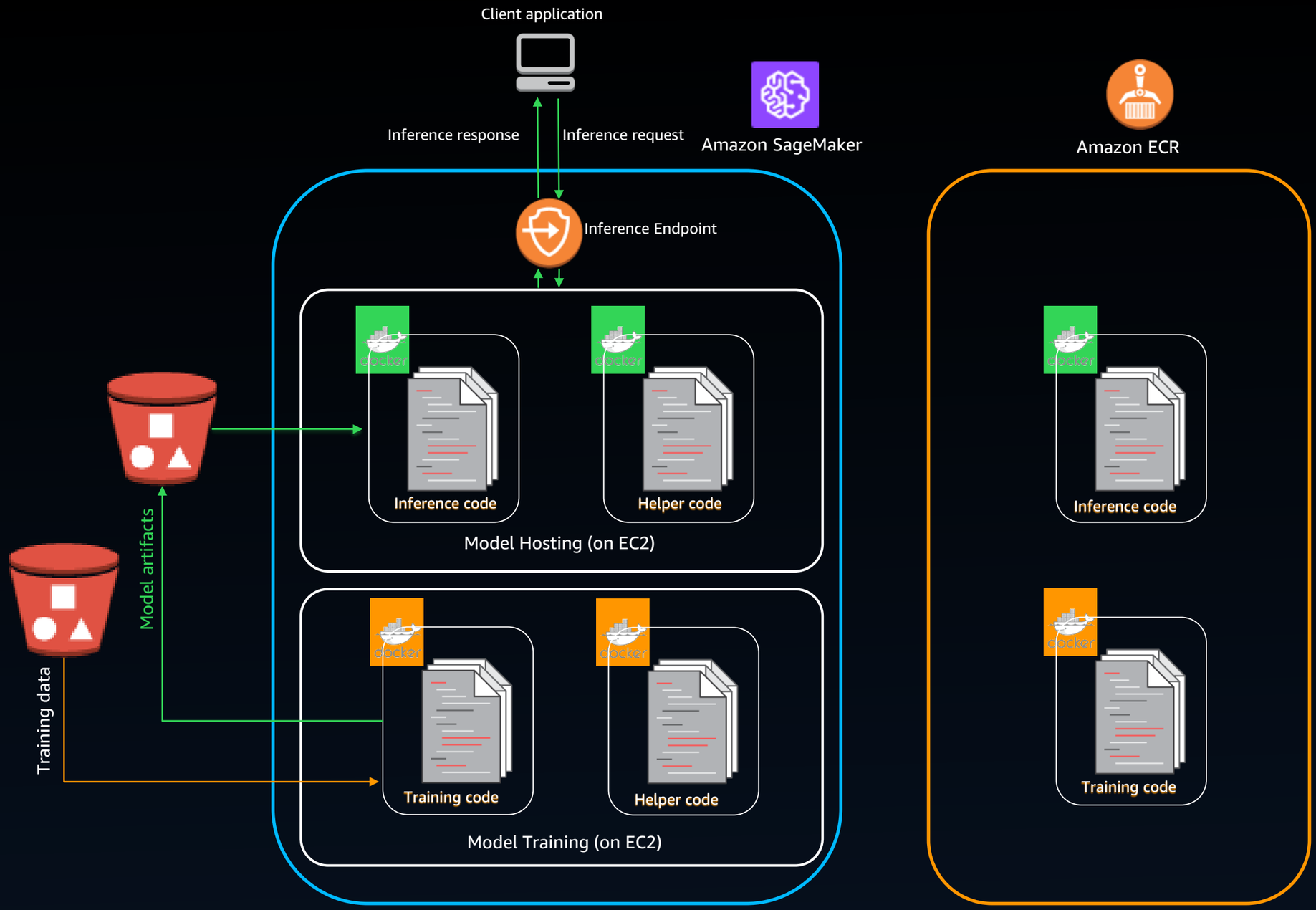


Amazon SageMaker

✓Launches docker image in the Amazon EC2 instance type

✓Downloads model from Amazon S3 to container host

✓Automatically scales instances based on scaling policy



Amazon SageMaker

✓Automatically captures metrics such as latency, #invocations

✓Logs / print results captured in Amazon CloudWatch

TensorFlow

NEW

Stock TensorFlow

65%

scaling efficiency with 256 GPUs

AWS-optimized TensorFlow

90%

scaling efficiency with 256 GPUs

30m

training time

14m

training time

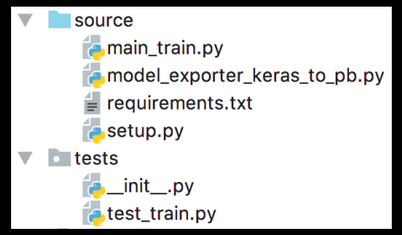
Fastest time for TensorFlow

- 85% of TensorFlow workloads in the cloud runs on AWS (2018 Nucleus report)
- Available w/ Amazon SageMaker and the AWS Deep Learning AMIs

Hyperparameters and environment variables

```python
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the training directory", required=True)
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts such as training data',
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input file wrt to the val directory", required=True)
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifacts such as validation data',
                    default=os.environ.get('SM_CHANNEL_VALIDATION', "."))

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save results",
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "result_data")
                    )

parser.add_argument("--model_dir", help="Do not use this.. required by SageMaker", default=None)

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save the snapshot to..",
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default=10, type=int)
parser.add_argument("--batch-size", help="The mini batch size", default=30, type=int)
```

main_train.py

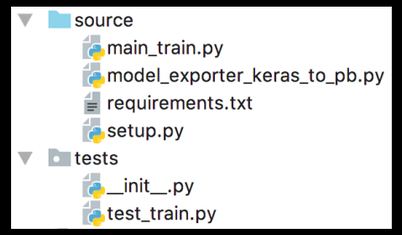


aws

Hyperparameters and environment variables

```python
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



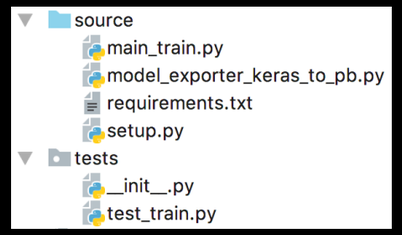
Submit your training job

```
From sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime

s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())

abalone_estimator = TensorFlow(entry_point='main_train.py',
                    source_dir="./source",
                    role=role,
                    py_version="py3",
                    framework_version = "1.11.0",
                    hyperparameters={'traindata' : 'abalone_train.csv',
                                'validationdata' : 'abalone_test.csv',
                                'epochs': 10,
                                'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": "mean_squared_error",
                                 "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                ,{"Name": "mean_absolute_error",
                                 "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                ,{"Name": "mean_absolute_percentage_error",
                                 "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                ],
                    train_instance_count=1,
                    train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                'validation':s3_input_prefix},
               job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

Hyperparameters and environment variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



Submit your training job

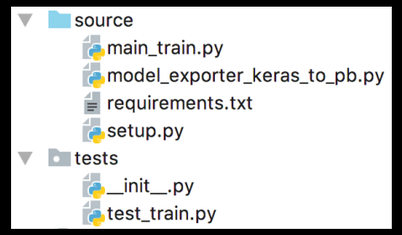
1) Specify source code
The entry point file and source code dir

```
From sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime

s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())

abalone_estimator = TensorFlow(entry_point='main_train.py',
                               source_dir="./source",
                               role=role,
                               py_version="py3",
                               framework_version = "1.11.0",
                               hyperparameters={'traindata' : 'abalone_train.csv',
                                        'validationdata' : 'abalone_test.csv',
                                        'epochs': 10,
                                        'batch-size': 32},
                               model_dir = s3_model_path,
                               metric_definitions = [{"Name": "mean_squared_error",
                                         "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                        ,{"Name": "mean_absolute_error",
                                         "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                        ,{"Name": "mean_absolute_percentage_error",
                                         "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                        ],
                               train_instance_count=1,
                               train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                      job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

Hyperparameters and environment variables

```python
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



Submit your training job

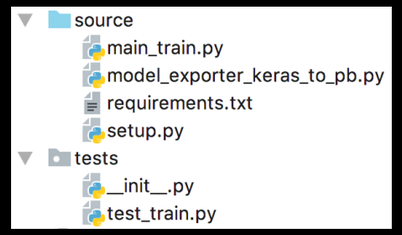
```
From sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime

s3_model_path = "s3://{}/models".format(sagemaker_

abalone_estimator = TensorFlow(entry_point='main_t
                    source_dir="./source",
                    role=role,
                    py_version="py3",
                    framework_version = "1.11.0",
                    hyperparameters={'traindata' : 'abalone_train.csv',
                                'validationdata' : 'abalone_test.csv',
                                'epochs': 10,
                                'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": "mean_squared_error",
                                 "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                ,{"Name": "mean_absolute_error",
                                 "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                ,{"Name": "mean_absolute_percentage_error",
                                 "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                ],
                    train_instance_count=1,
                    train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                      job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

2) Pass hyper parameters
The hyperparameter dict key name, e.g., “traindata”, “epochs”, matches the argument name “--traindata”, “--epochs”

Hyperparameters and environment variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



Subm

From sager
from time i

s3_model_

abalone_es

3) Save artifacts to output

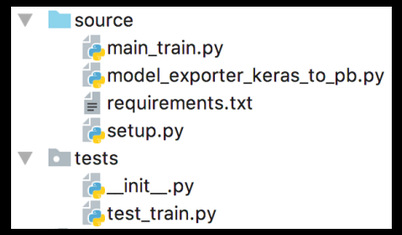
Save model to output to dir pointed by environment variable SM_MODEL_DIR. Artifacts placed here are automatically uploaded to Amazon S3 and available during inference.

```
py_version = "py3",
framework_version = "1.11.0",
hyperparameters={'traindata' : 'abalone_train.csv',
                 'validationdata' : 'abalone_test.csv',
                 'epochs': 10,
                 'batch-size': 32},
model_dir = s3_model_path,
metric_definitions = [{"Name": "mean_squared_error",
                       "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                      ,{"Name": "mean_absolute_error",
                       "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                      ,{"Name": "mean_absolute_percentage_error",
                       "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                      ],
train_instance_count=1,
train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

Hyperparameters and environment variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



Submit your training job

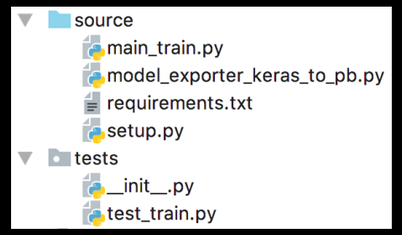
```
From sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime

s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())

abalone_estimator = TensorFlow(entry_point='main_train.py',
                    source_dir="./source",
                    role=role,
                    py_version="py3",
                    framework_version = "1.11.0",
                    hyperparameters={'traindata' : 'ab
                                'validationdata' : 'abalon
                                'epochs': 10,
                                'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": "m
                                 "Regex": "## validatio
                                ,{"Name": "mean_absolute_error",
                                 "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                ,{"Name": "mean_absolute_percentage_error",
                                 "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                ],
                    train_instance_count=1,
                    train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

4) Specify the instance type for your training. Increase the instance count if your code supports distributed training.

Hyperparameters and environment variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



Submit your training job

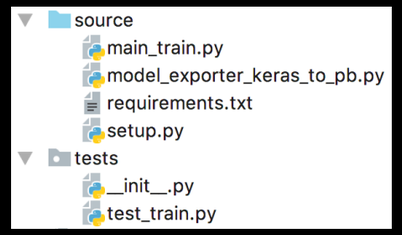
```
From sagemaker.tensorflow import Ten
from time import gmtime, strftime

s3_model_path = "s3://{}/models".form

abalone_estimator = TensorFlow(entry_
                    source_dir="./source",
                    role=role,
                    py_version="py3",
                    framework_version = "1.11.0",
                    hyperparameters={'traindata' : 'abalone_train.csv',
                                     'validationdata' : 'abalone_test.csv',
                                     'epochs': 10,
                                     'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": "mean_squared_error",
                                           "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                          ,{"Name": "mean_absolute_error",
                                           "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                          ,{"Name": "mean_absolute_percentage_error",
                                           "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                          ],
                    train_instance_count=1,
                    train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

5) Specify python version (py3) TensorFlow framework version (1.11.0).

Hyperparameters and environment variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the t
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifact
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input f
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifa
                    default=os.environ.get('SM_CHANNEL_VALIDATION',

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save resu
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "re
                    )

parser.add_argument("--model_dir", help="Do not use this.. required

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save th
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default
parser.add_argument("--batch-size", help="The mini batch size", defa
```

main_train.py



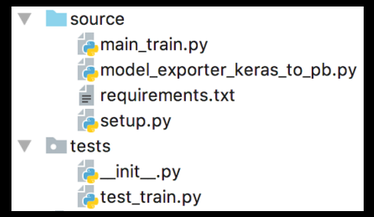
Submit your training job

```
From sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime

s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())

abalone_estimator = TensorFlow(entry_point='main_train.py',
                               source_dir="./source",
                               role=role,
                               py_version="py3",
                               framework_version = "1.11.0",
                               hyperparameters={'traindata' : 'abalone_train.csv',
                                        'validationdata' : 'abalone_test.csv',
                                        'epochs': 10,
                                        'batch-size': 32},
                               model_dir = s3_model_path,
                               metric_definitions = [{"Name
                                         "Regex": "## vali
                                        ,{"Name": "mean_
                                         "Regex": "## vali
                                        ,{"Name": "mean_
                                         "Regex": "## vali
                                        ],
                               train_instance_count=1,
                               train_instance_type='ml.c4.x

abalone_est mator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       ob_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

6) **Map Amazon S3 prefix to local download directory**
The key name, e.g., ‘train’, matches environment variable SM_CHANNEL _**TRAIN** suffix **TRAIN**, for train data directory.

Model metrics

```
# For a mean squared error regression problem
# Using RMSProp optimiser with mean squared error
metrics = ['mse', 'mae', 'mape']
model.compile(optimizer='rmsprop',
              loss='mse', metrics=metrics)

# load train & test data
train_x, train_y = input_transformer_load(os.path.join(training_dir, training_filename))
val_x, val_y = input_transformer_load(os.path.join(val_dir, val_filename))
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size, validation_data=(val_x, val_y))
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalone_age_predictor.h5")
model.save(model_keras_path)

# Step 2: Log your metrics in a special format so that it can be extracted using a regular express
# This allows SageMaker to report this metrics and allows hyper parameter tuning
# Note: Use a special marker for SageMaker to extract the metrics, say ## Metric ##
for i, m in enumerate(metrics):
    print("## validation_metric_{} ##: {}".format(m, scores[1+i]))
```



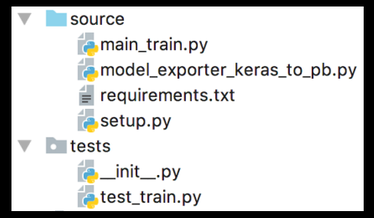
main_train.py

Model metrics

```
# For a mean squared error regression problem
# Using RMSProp optimiser with mean squared error
metrics = ['mse', 'mae', 'mape']
model.compile(optimizer='rmsprop',
              loss='mse', metrics=metrics)
```

```
# load train & test data
train_x, train_y = input_transformer_load(os.path.join(
val_x, val_y = input_transformer_load(os.path.join(val_
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=b
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "aba
model.save(model_keras_path)
```

```
# Step 2: Log your metrics in a special format so that
# This allows SageMaker to report this metrics and allo
# Note: Use a special marker for SageMaker to extract t
for i, m in enumerate(metrics):
    print("## validation_metric_{} ##: {}".format(m, sc
```



main_train.py

Submit your training job

```
from sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = TensorFlow(entry_poin
                    source_dir="./source",
                    role=role,
                    py_version="py3",
                    framework_version = "1.11
                    hyperparameters={'trainda
                                'validationdata' :
                                'epochs': 10,
                                'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": "mean_squared_error",
                                "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                               ,{"Name": "mean_absolute_error",
                                "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                               ,{"Name": "mean_absolute_percentage_error",
                                "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                               ],
                    train_instance_count=1,
                    train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

7) Model metrics regex to trace and visualize your model performance.

Demo: Amazon SageMaker with TensorFlow, and Keras in Python 3

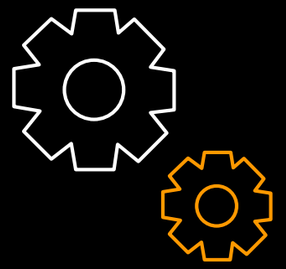
aws INNOVATE
ONLINE CONFERENCE

PyTorch

Flexible

Fast prototyping

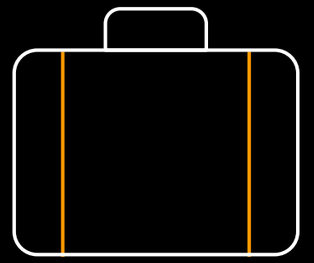
Seamless transition from research to production using Amazon SageMaker



Versatile

Train and run a variety of models, including CNN and mLSTM

Train custom models with Facebook's Fairseq toolkit on Amazon SageMaker



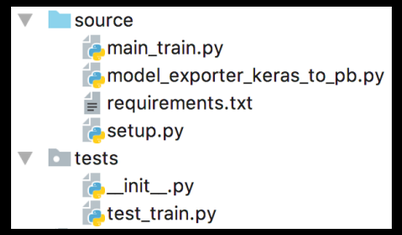
Portable

Develop models in PyTorch and transfer to other frameworks like MXNet for inference using ONNX

PyTorch training with Amazon SageMaker

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifa
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation input
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation arti
                    default=os.environ.get('SM_CHANNEL_VALIDATION'

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save re
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "
                    )

parser.add_argument("--model_dir", help="Do not use this.. require

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defau
parser.add_argument("--batch-size", help="The mini batch size", de
```

main_train.py



Submit your training job

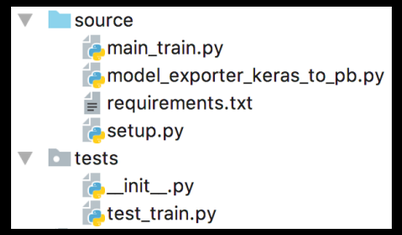
```
from sagemaker.pytorch import PyTorch
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = PyTorch(entry_point='main_train.py',
                            source_dir="./source",
                            role=role,
                            py_version="py3",
                            framework_version = "1.0.0",
                            hyperparameters={'traindata' : 'abalone_train.csv',
                                     'validationdata' : 'abalone_test.csv',
                                     'epochs': 10,
                                     'batch-size': 32},
                            model_dir = s3_model_path,
                            metric_definitions = [{"Name": "mean_squared_error",
                                      "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                     ,{"Name": "mean_absolute_error",
                                      "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                     ,{"Name": "mean_absolute_percentage_error",
                                      "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                     ],
                            train_instance_count=1,
                            train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

PyTorch training with Amazon SageMaker

1) Specify source code
The entry point file and source code dir

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifa
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation input
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation arti
                    default=os.environ.get('SM_CHANNEL_VALIDATION'

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save re
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "
                    )

parser.add_argument("--model_dir", help="Do not use this.. require

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defau
parser.add_argument("--batch-size", help="The mini batch size", de
```

main_train.py



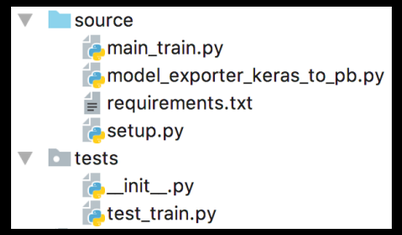
Submit your training job

```
from sagemaker.pytorch import PyTorch
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = PyTorch(entry_point='main_train.py',
                            source_dir="./source",
                            role=role,
                            py_version="py3",
                            framework_version = "1.0.0",
                            hyperparameters={'traindata' : 'abalone_train.csv',
                                             'validationdata' : 'abalone_test.csv',
                                             'epochs': 10,
                                             'batch-size': 32},
                            model_dir = s3_model_path
                            metric_definitions = [{"Name": "mean_squared_error",
                                                   "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                                  ,{"Name": "mean_absolute_error",
                                                   "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                                  ,{"Name": "mean_absolute_percentage_error",
                                                   "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                                  ],
                            train_instance_count=1,
                            train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

PyTorch training with Amazon SageMaker

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifa
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation input
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation arti
                    default=os.environ.get('SM_CHANNEL_VALIDATION'

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save re
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "
                    )

parser.add_argument("--model_dir", help="Do not use this.. require

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defau
parser.add_argument("--batch-size", help="The mini batch size", de
```

main_train.py



Submit your t

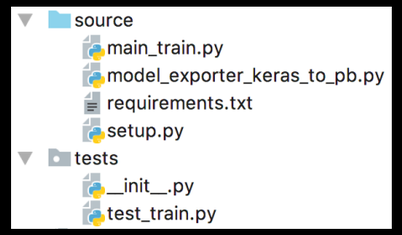
```
from sagemaker.pytorch im
from time import gmtime, s
s3_model_path = "s3://{}/m
abalone_estimator = PyTorc
                    source_dir=
                    role=role,
                    py_version
                    framework_version = "1.0.0",
                    hyperparameters={'traindata' : 'abalone_train.csv',
                                     'validationdata' : 'abalone_test.csv',
                                     'epochs': 10,
                                     'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": "mean_squared_error",
                                           "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                          ,{"Name": "mean_absolute_error",
                                           "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                          ,{"Name": "mean_absolute_percentage_error",
                                           "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                          ],
                    train_instance_count=1,
                    train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

2) Pass hyper parameters
The hyperparameter dict key name, e.g., “traindata”, “epochs” matches the argument name “--traindata”, “--epochs”

PyTorch training with Amazon SageMaker

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifa
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation input
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation arti
                    default=os.environ.get('SM_CHANNEL_VALIDATION'

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save re
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "
                    )

parser.add_argument("--model_dir", help="Do not use this.. require

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defau
parser.add_argument("--batch-size", help="The mini batch size", de
```

main_train.py



Submit your training job

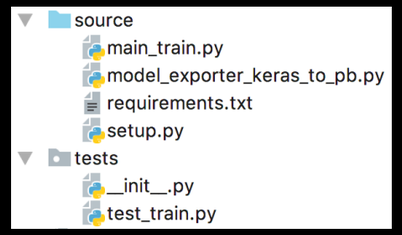
```
from sagemaker
from time impor
s3_model_path =
abalone_estimat
                'batch-size': 32},
model_dir = s3_model_path,
metric_definitions = [{"Name": "mean_squared_error",
                       "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                      ,{"Name": "mean_absolute_error",
                       "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                      ,{"Name": "mean_absolute_percentage_error",
                       "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                      ],
train_instance_count=1,
train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                      job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

3) Save artifacts to output
Save model to output to dir pointed by environment variable SM_MODEL_DIR. Artifacts placed here are automatically uploaded to Amazon S3 and available during inference

PyTorch training with Amazon SageMaker

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifa
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation input
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation arti
                    default=os.environ.get('SM_CHANNEL_VALIDATION'

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save re
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "
                    )

parser.add_argument("--model_dir", help="Do not use this.. require

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defau
parser.add_argument("--batch-size", help="The mini batch size", de
```

main_train.py



Submit your training job

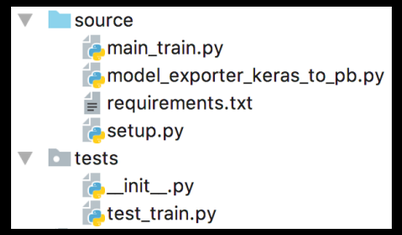
```
from sagemaker.pytorch import PyTorch
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = PyTorch(entry_point='main_train.py',
                            source_dir="./source",
                            role=role,
                            py_version="py3",
                            framework_version = "1
                            hyperparameters={'train
                                             'validationdata
                                             'epochs': 10,
                                             'batch-size': 32
                            model_dir = s3_model_p
                            metric_definitions = [{"N
                                                   "Regex": "##
                                                  ,{"Name": "m
                                                   "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                                  ,{"Name": "mean_absolute_percentage_error",
                                                   "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                                  ],
                            train_instance_count=1,
                            train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

4) Specify the instance type for your training. Increase the instance count if your code supports distributed training

PyTorch training with Amazon SageMaker

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifa
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation input
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation arti
                    default=os.environ.get('SM_CHANNEL_VALIDATION'

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save re
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "
                    )

parser.add_argument("--model_dir", help="Do not use this.. require

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defau
parser.add_argument("--batch-size", help="The mini batch size", de
```

main_train.py



Submit your training job

```
from sagemaker.pytorch import PyTorch
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = PyTorch(entry_point='main_train.py',
                            source_dir="./source",
                            role=role,
                            py_version="py3",
                            framework_version = ''
                            hyperparameters={'tra
                                             'validationda
                                             'epochs': 10,
                                             'batch-size': 3
                            model_dir = s3_model_
                            metric_definitions = [{'
                                                   "Regex": "#
                                                  ,{"Name": "
                                                   "Regex": "#
                                                  ,{"Name": "
                                                   "Regex": "#
                                                  ],
                            train_instance_count=1
                            train_instance_type='m
```

5) **Map Amazon S3 prefix to local download directory**
The key name, e.g., ‘train’, matches environment variable SM_CHANNEL _**TRAIN** suffix **TRAIN**, for train data directory

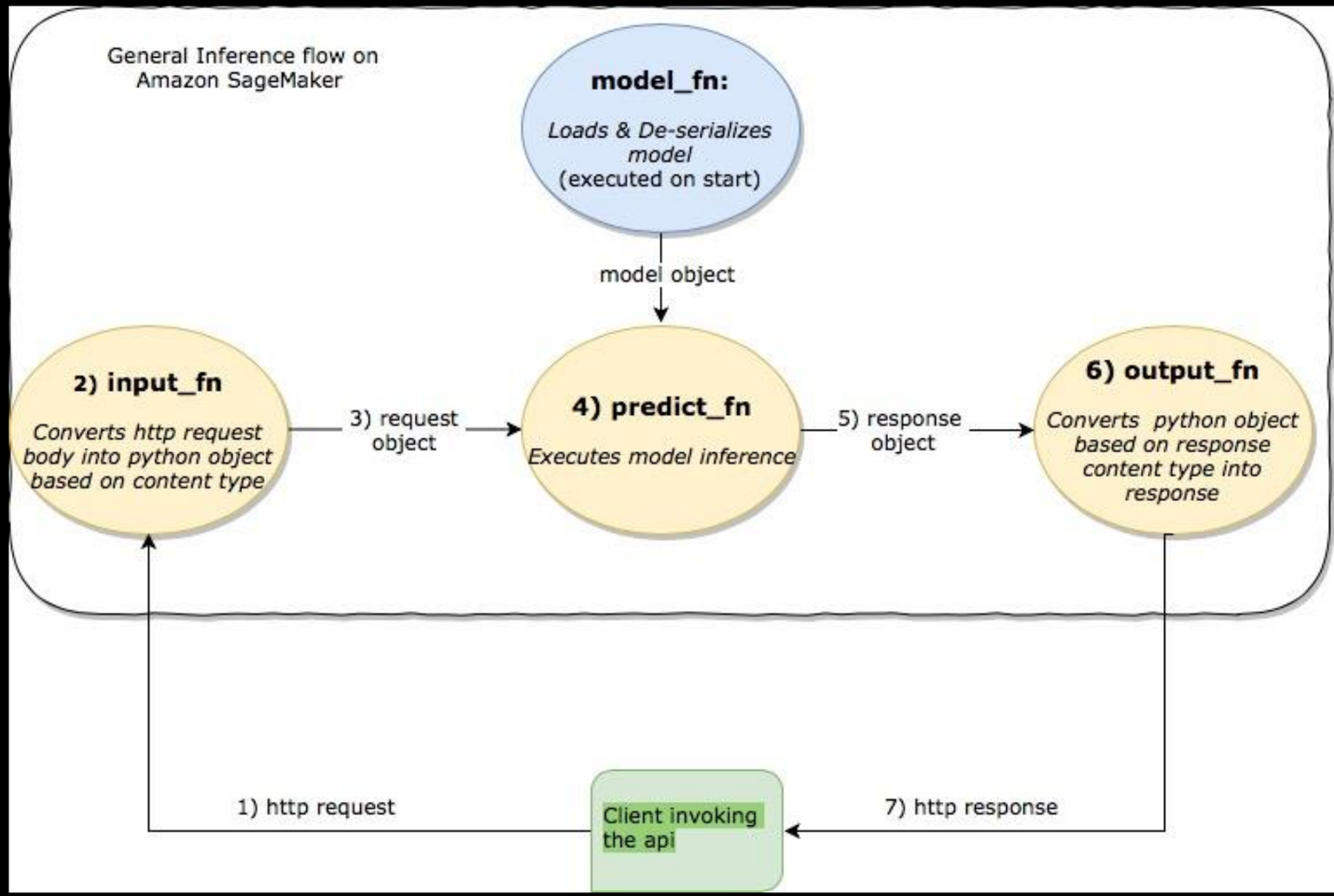

```
abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name='abalone-age-py3-{}'.format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

General Amazon SageMaker inference flow

Sample http request

POST /endpoints/abalone.. HTTP/1.1
Host: *runtime.sagemaker ..*
..
Authorization: AWS4-HMAC-SHA256 Credential ***
Content-Type: application/json
Accept: application/json

[[.34,5.6],[4,56]...]



Note: This is a general flow only. The exact function signatures depend on the Amazon SageMaker container for the specific deep learning framework, including its version! Please check the Amazon SageMaker samples on https://github.com/awslabs/amazon-sagemaker-examples for full details.

PyTorch inference with Amazon SageMaker: Load model

```
def model_fn(model_dir):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = torch.nn.DataParallel(Net())
    with open(os.path.join(model_dir, 'model.pth'), 'rb') as f:
        model.load_state_dict(torch.load(f))
    return model.to(device)
```

Note: Your model can be deployed on a CPU or GPU instance type you choose when you deploy the endpoint. This code supports both.

```
# Deploy my estimator to an Amazon SageMaker endpoint and get a Predictor predictor =
# E.g., m4.xlarge is a CPU instance type or if you use ml.p3.2xlarge then it is a GPU instance type

abalone_estimator.deploy(
instance_type='ml.m4.xlarge',
initial_instance_count=1)
```

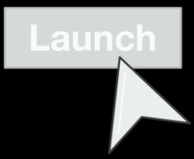
Apache MXNet

Start with high-quality, pre-trained models

- Gluon CV and Gluon NLP

Refine with fast, scalable training

- Keras-MXNet up to 2x faster than Keras-TensorFlow
- Near-linear scalability up to 256 GPUs
- Dynamic training



Deploy using familiar tools

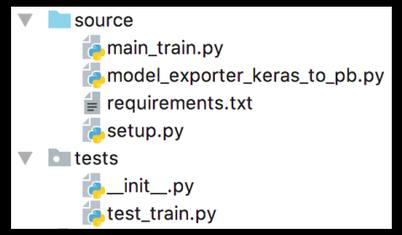
- Java/Scala APIs
- MXNet Model Server

MXNet training with Amazon SageMaker

Specify source code
The entry point file and source code dir

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to th
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artif
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."
# val dir files
parser.add_argument("--validationdata", help="The validation inpu
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation art
                    default=os.environ.get('SM_CHANNEL_VALIDATION

# output dir to save any files such as predictions, logs, etc
parser.add_argument("--outputdir", help="The output dir to save r
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', '
                    )

parser.add_argument("--model_dir", help="Do not use this.. requir

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save
                    default=os.environ.get('SM_MODEL_DIR', None))

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", defa
parser.add_argument("--batch-size", help="The mini batch size", d
```

main_train.py



Submit your training job

```
from sagemaker.mxnet import MXNet
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = MXNet(entry_point='main_train.py',
                          source_dir="./source"
                          role=role,
                          py_version="py3",
                          framework_version = "1.3.0"
                          hyperparameters={'traindata' : 'abalone_train.csv',
                                           'validationdata' : 'abalone_test.csv',
                                           'epochs': 10,
                                           'batch-size': 32},
                          model_dir = s3_model_path,
                          metric_definitions = [{"Name": "mean_squared_error",
                                                 "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                                                ,{"Name": "mean_absolute_error",
                                                 "Regex": "## validation_metric_mae ##: (\d*[.]?\d*)"}
                                                ,{"Name": "mean_absolute_percentage_error",
                                                 "Regex": "## validation_metric_mape ##: (\d*[.]?\d*)"}
                                                ],
                          train_instance_count=1,
                          train_instance_type='ml.c4.xlarge')

abalone_estimator.fit( {'train': s3_input_prefix,
                        'validation':s3_input_prefix},
                       job_name="ablone-age-py3-{}".format(strftime("%Y-%m-%d-%H-%M-%S", gmtime())))
```

MXNet inference with Amazon SageMaker

```
def model_fn(model_dir):
    """
    Load the gluon model. Called once when hosting service starts.
    :param: model_dir The directory where model files are stored.
    :return: a model (in this case a Gluon network)
    """
    net = gluon.SymbolBlock.imports(
        '%s/model-symbol.json' % model_dir,
        ['data'],
        '%s/model-0000.params' % model_dir,
    )
    return net
```

```
# Deploy your MXNet estimator to an Amazon SageMaker endpoint

abalone_estimator.deploy(
instance_type='ml.m4.xlarge',
initial_instance_count=1)
```

1. No code required if using default Amazon SageMaker MXNet Model Server.
2. Works well if the model is a single artifact of type MXNet nn.module.
3. Otherwise, write custom code, e.g., load Gluon model.

MXNet inference with Amazon SageMaker: Custom data formats

```
def input_fn(request_body, request_content_type, model):
    """An input_fn that loads a pickled numpy array"""
    if request_content_type == "application/python-pickle":
        array = np.load(StringIO(request_body))
        array.reshape(model.data_shpaes[0])
        return mx.io.NDArrayIter(mx.ndarray(array))
    else:
        # Handle other content-types here or raise an Exception
        # if the content type is not supported.
        pass
```

Note: This sample is for Amazon SageMaker with MXNet 1.3.0

# Deserialize the Invoke request body into an object we can perform prediction on
input_object = input_fn(request_body, request_content_type, model)

# Perform prediction on the deserialized object, with the loaded model
prediction = predict_fn(input_object, model)

# Serialize the prediction result into the desired response content type
ouput = output_fn(prediction, response_content_type)

Closing comments

aws INNOVATE
ONLINE CONFERENCE

Learn from AWS experts. Advance your skills and knowledge. Build your future in the AWS Cloud.

Digital Training

Free, self-paced online courses built by AWS experts

Classroom Training

Classes taught by accredited AWS instructors

AWS Certification

Exams to validate expertise with an industry-recognized credential

Ready to begin building your cloud skills?
Get started at: https://www.aws.training/

Why work with an APN Partner?

APN Partners are uniquely positioned to help your organization at any stage of your cloud adoption journey, and they:

- Share your goals—focused on your success
- Help you take full advantage of all the business benefits that AWS has to offer
- Provide services and solutions to support any AWS use case across your full customer life cycle

APN Partners with deep expertise in AWS services:

AWS Managed Service Provider (MSP) Partners

APN Partners with cloud infrastructure and application migration expertise

AWS Competency Partners

APN Partners with verified, vetted, and validated specialized offerings

AWS Service Delivery Partners

APN Partners with a track record of delivering specific AWS services to customers

Find the right APN Partner for your needs: https://aws.amazon.com/partners/find/

Thank you for attending AWS Innovate

We hope you found it interesting! A kind reminder to **complete the survey.**
Let us know what you thought of today's event and how we can improve the event experience for you in the future.


aws-apac-marketing@amazon.com


twitter.com/AWSCloud


facebook.com/AmazonWebServices


youtube.com/user/AmazonWebServices


slideshare.net/AmazonWebServices


twitch.tv/aws