AWS INNOVATE

ONLINE CONFERENCE 2019

MACHINE LEARNING & AI EDITION

Opensource ML Frameworks on Amazon SageMaker Featuring TensorFlow, PyTorch, MXnet, Keras and Gluon (Level 300)

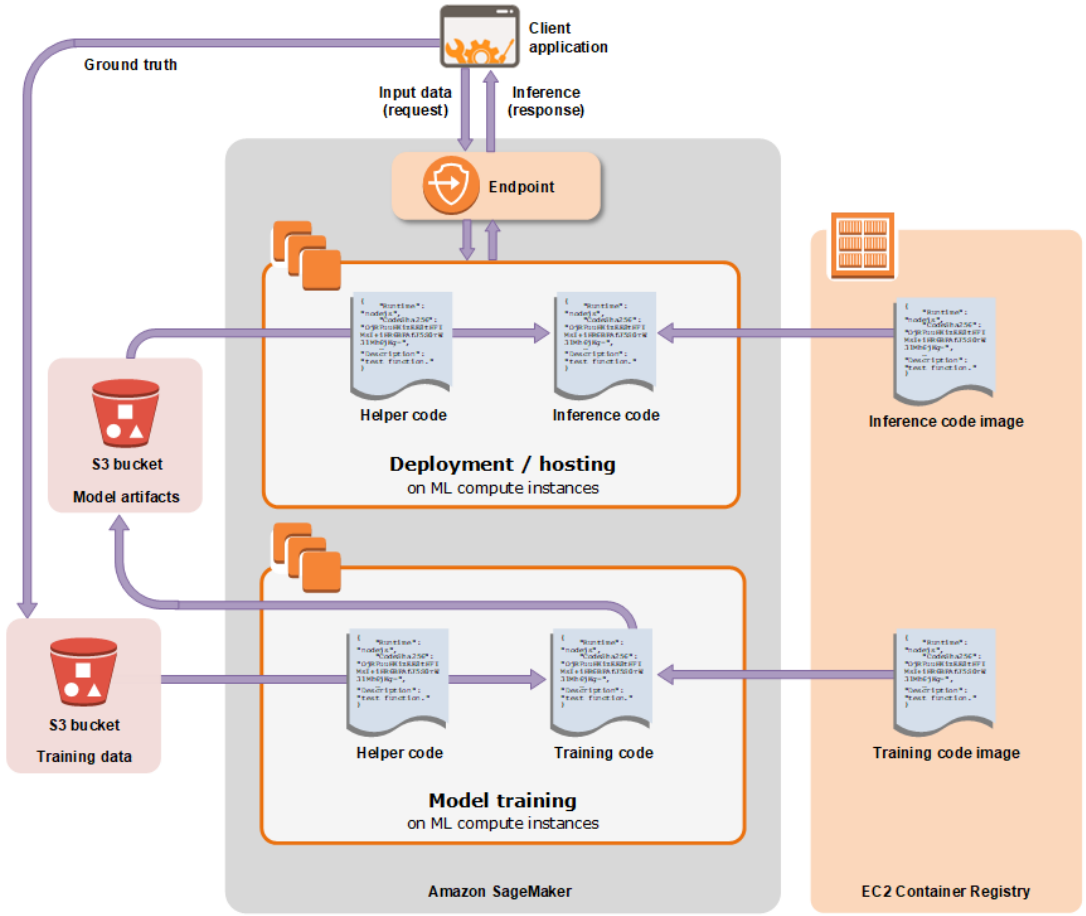
Aparna Elangovan, Solutions Architect, AWS ANZ

aws

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

Amazon SageMaker – Training phase



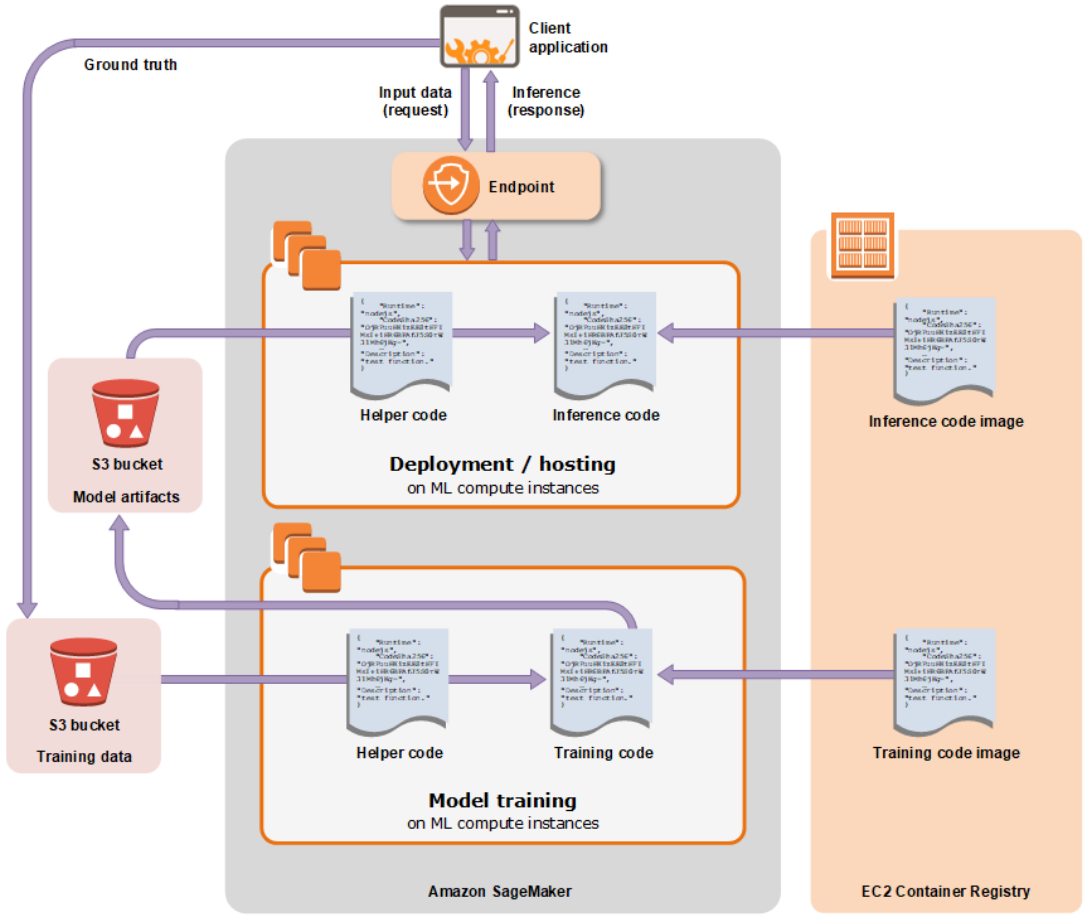
You provide

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

Amazon SageMaker

- ✓ Automatically provisions and tears down Amazon EC2 instances for your training job
- ✓ Launches docker image in the EC2 instance type required for training & inference
- ✓ Downloads code & data from S3 to container host
- ✓ Executes training job
- ✓ Copies model from container host to S3
- ✓ Copies results from container host to S3
- ✓ Logs / print results captured in Amazon CloudWatch
- ✓ Captures training infrastructure & performance metrics in CloudWatch

Amazon SageMaker – Inference phase



You provide

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

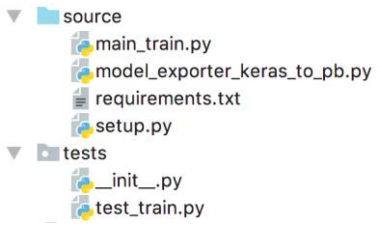
SageMaker

- ✓ Launches docker image in the EC2 instance type
- ✓ Downloads model from S3 to container host
- ✓ Auto scales instances based on scaling policy
- ✓ Automatically captures metrics such as latency, #invocations
- ✓ Logs / print results captured in CloudWatch

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



Hyperparameters and Environment Variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the tra
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input fi
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifac
                    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 result
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "resu
                    )

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

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

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default=1
parser.add_argument("--batch-size", help="The mini batch size", defau
```

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 tra
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input fi
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifac
                    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 resul
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "res
                    )

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

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

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default=1
parser.add_argument("--batch-size", help="The mini batch size", defau
```

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.d

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())))
```

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

aws

Hyperparameters and Environment Variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the tra
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifac
                    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 result
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "res
                    )

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

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

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default=1
parser.add_argument("--batch-size", help="The mini batch size", defau
```

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 = [{"Nam
                                        "Regex": "## val
                                       ,{"Name": "mean
                                        "Regex": "## val
                                       ,{"Name": "mean
                                        "Regex": "## val
                                       ],
                               train_instance_count=1,
                               train_instance_type='ml.c4.x

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())))
```

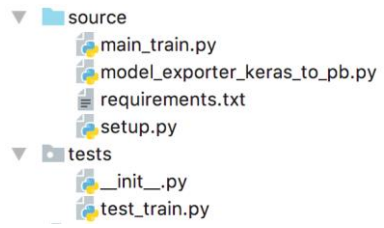
2) **Map S3 prefix to local download directory**

The key name e.g. 'train' matches environment variable SM_CHANNEL_**TRAIN** suffix **TRAIN** , for train data directory

Hyperparameters and Environment Variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the tra
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input fil
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifact
                    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 result
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "resu
                    )

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

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

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default=1
parser.add_argument("--batch-size", help="The mini batch size", defaul
```

main_train.py



Submit your training job

```
from sagemaker.tensorflow import Te
from time import gmtime, strftime

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

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())))
```

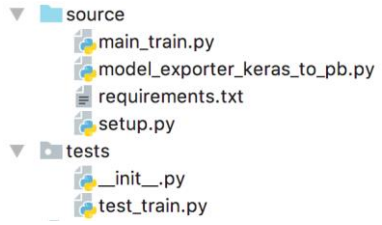
3) Pass hyper parameters
The hyperparameter dict key name e.g. “traindata”, “epochs” matches the argument name “--traindata” , “--epochs”

aws

Hyperparameters and Environment Variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the tra
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input fi
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifac
                    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 resul
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "res
                    )

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

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

# Additional parameters for your code
parser.add_argument("--epochs", help="The number of epochs", default=1
parser.add_argument("--batch-size", help="The mini batch size", defau
```

main_train.py



Submit your training job

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

s3_model_pa

abalone_est
```

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

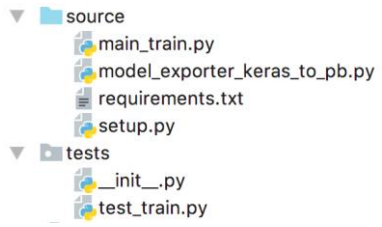
```
                validationdata': 'abalone_test.csv',
                'epochs': 10,
                batch-size': 32},
model_dir = s3_model_path,
metric_definitions = [{"Name": "mean_squared_error",
                       "Regex": "## validation_metric_mse ##: (\d*[.]?\d*)"}
                      ,{"Name": "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())))
```


aws

Hyperparameters and Environment Variables

```
# Train dir files
parser.add_argument("--traindata", help="The input file wrt to the tr
# The environment variable SM_CHANNEL_TRAIN is defined
parser.add_argument('--traindata-dir',
                    help='The directory containing training artifacts
                    default=os.environ.get('SM_CHANNEL_TRAIN', "."))
# val dir files
parser.add_argument("--validationdata", help="The validation input fi
parser.add_argument('--validationdata-dir',
                    help='The directory containing validation artifac
                    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 resul
                    default=os.environ.get('SM_OUTPUT_DATA_DIR', "res
                    )

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

# This is where the model needs to be saved to
parser.add_argument("--snapshot_dir", help="The directory to save the
                    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", defau
```

main_train.py



Submit your training job

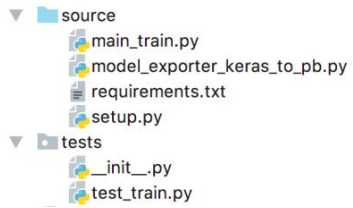
```
from sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime

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' : '
                            'validationdata' : 'abal
                            'epochs': 10,
                            'batch-size': 32},
                    model_dir = s3_model_path,
                    metric_definitions = [{"Name": '
                                "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 the instance type for your training. Increase the instance count if your code supports distributed training

aws

Model metrics & Framework Versions

```
# 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]))
```

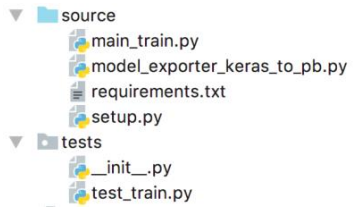


main_train.py

aws

Model metrics & Framework Versions

```
# 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(tra
val_x, val_y = input_transformer_load(os.path.join(val_di
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=bat
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalo
model.save(model_keras_path)

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



main_train.py

Submit your training job

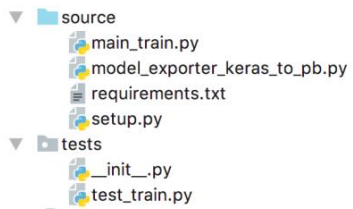
```
from sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = TensorFlow(entry_point='main_train.py',
                    source_dir="./source",
                    role=role,
                    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())))
```

Model metrics & Framework Versions

```
# 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(tra
val_x, val_y = input_transformer_load(os.path.join(val_di
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=bat
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalo
model.save(model_keras_path)
```

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



main_train.py

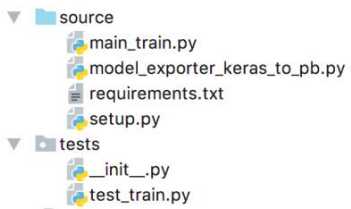
Submit your training job

```
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
                source_dir="./source",
                role=role,
                py_version="py3",
                framework_version = "1.1
                hyperparameters={'traind
                        '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())))
```

6) Model metrics regex to trace and visualize your model performance

Model metrics & Framework Versions

```
# 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(tra
val_x, val_y = input_transformer_load(os.path.join(val_di
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=batc
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalon
model.save(model_keras_path)

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



main_train.py

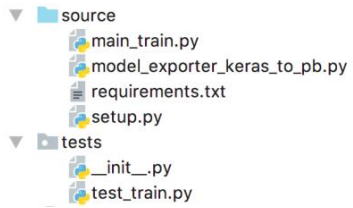
Submit your training job

```
from sagemaker.tensorflow import Ten
from time import gmtime, strftime
s3_model_path = "s3://{}/models".forn
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())))
```

7) Specify python version (py3)
TensorFlow framework version
(1.11.0)

Model metrics & Framework Versions

```
# 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]))
```

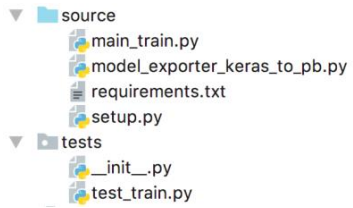


main_train.py

aws

Model metrics & Framework Versions

```
# 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(tra
val_x, val_y = input_transformer_load(os.path.join(val_di
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=bat
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalo
model.save(model_keras_path)

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



main_train.py

Submit your training job

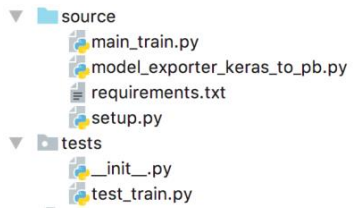
```
from sagemaker.tensorflow import TensorFlow
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sagemaker_session.default_bucket())
abalone_estimator = TensorFlow(entry_point='main_train.py',
                    source_dir="./source",
                    role=role,
                    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())))
```

Model metrics & Framework Versions

```
# 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(tra
val_x, val_y = input_transformer_load(os.path.join(val_di
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=bat
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalo
model.save(model_keras_path)
```

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



main_train.py

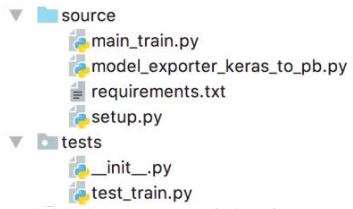
Submit your training job

```
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
                source_dir="./source",
                role=role,
                py_version="py3",
                framework_version = "1.1
                hyperparameters={'traind
                        '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())))
```

6) Model metrics regex to trace and visualize your model performance

Model metrics & Framework Versions

```
# 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(tra
val_x, val_y = input_transformer_load(os.path.join(val_di
# Start training
model.fit(train_x, train_y, epochs=epochs, batch_size=bat
# model evaluate
scores = model.evaluate(val_x, val_y)
# model save in keras default format
model_keras_path = os.path.join(model_snapshotdir, "abalo
model.save(model_keras_path)

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



main_train.py

Submit your training job

```
from sagemaker.tensorflow import Ten
from time import gmtime, strftime
s3_model_path = "s3://{}/models".forr
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())))
```

7) Specify python version (py3)
TensorFlow framework version
(1.11.0)

Demo – Abalone age prediction

SageMaker with TensorFlow, Keras in Python 3

PyTorch on SageMaker

PyTorch training with SageMaker

```
# 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)
```

PyTorch training with 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
```

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 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
```

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 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
```

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 = [{
                                              "Regex": "#
                                             ,{"Name": "
                                              "Regex": "#
                                             ,{"Name": "
                                              "Regex": "#
                                             ],
                            train_instance_count=
                            train_instance_type='n

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())))
```

2) **Map S3 prefix to local download directory**

The key name e.g. ‘train’ matches environment variable SM_CHANNEL_**TRAIN** suffix **TRAIN** , for train data directory

PyTorch training with 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
```

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

Submit your training job

```
from sagemaker.pytorch import PyTorch
from time import gmtime, strftime
s3_model_path = "s3://{}/models".format(sage
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 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
```

Submit your training job

```
from sagemaker.pytorch import PyTorch
from time import gmtime, strftime
s3_model_path = "s3:
abalone_estimator =
                    sou
                    role
                    py_
                    fram
                    hyp
                                    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())))
```

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

PyTorch training with 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
```

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'
                                             'validationdata' : 'aba
                                             'epochs': 10,
                                             'batch-size': 32},
                            model_dir = s3_model_path,
                            metric_definitions = [{"Name":
                                                   "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())))
```

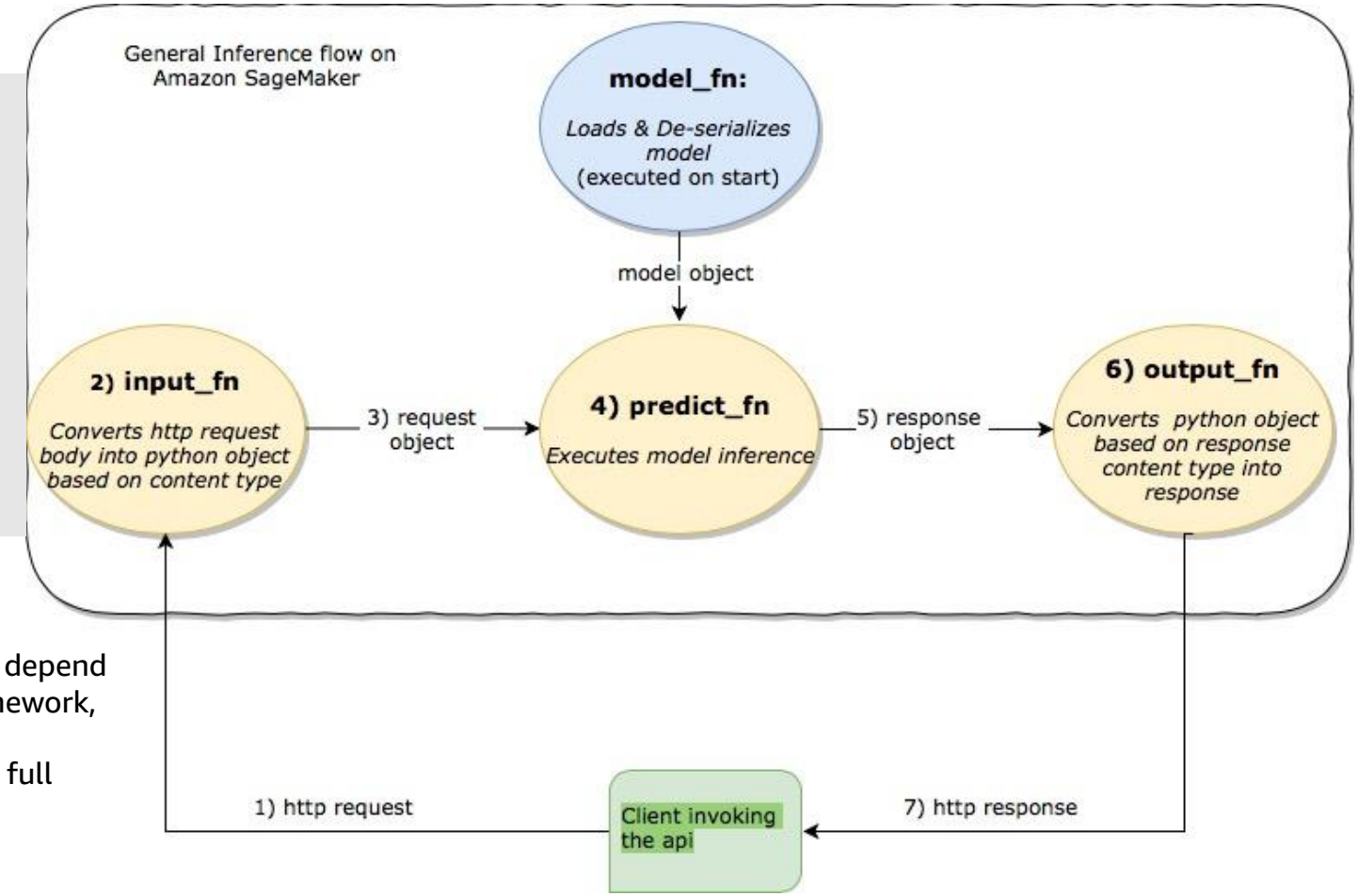
5) Specify the instance type for your training. Increase the instance count if your code supports distributed training

General 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 SageMaker container for the specific deep learning framework, including its version! Please check the SageMaker samples on https://github.com/awslabs/amazon-sagemaker-examples for full details

PyTorch inference with 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 CPU or GPU instance type you choose when you deploy the endpoint. This code supports both.

Deploy my estimator to a 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)

MXNet on SageMaker

MXNet training with SageMaker

```
# 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)
```

MXNet training with 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
```

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 training with 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
```

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 training with 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
```

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 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 a SageMaker Endpoint

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

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

MXNet inference with SageMaker – Custom Dataformats

```python
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 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)

Summary – Amazon SageMaker

- Built-in support for opensource ML Frameworks TensorFlow, PyTorch, Chainer, MXNet and Scikit-learn
- Simplifies porting existing code to use AWS Platform, making your code fully portable
- Simplifies model endpoint inference creation – One Click Deployment
- Automatic monitoring and tracking logs, infrstraucture and model performance
- Automatic experiment tracking including data, code, hyperparameters & logs

Learn why customers choose AWS for machine learning

Demo Arena

Watch how machine learning is used in real life applications

Ask the Experts

Get your questions answered by AWS Experts

Machine Learning on AWS

https://aws.amazon.com/machine-learning/

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/

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