



このコンテンツは公開から3年以上経過しており内容が古い可能性があります  
最新情報については[サービス別資料](#)もしくはサービスのドキュメントをご確認ください

# [AWS Black Belt Online Seminar]

## Amazon SageMaker

サービスカットシリーズ

Makoto Shimura, Solutions Architect

2019/02/06

# 自己紹介

## 志村 誠

- アナリティクススペシャリスト ソリューションアーキテクト
  - データ分析・機械学習系サービスを担当
  - 前職はログ解析基盤構築・データ分析等
  - 好きなサービス
    - Amazon Athena
    - AWS Glue
    - そして Amazon SageMaker



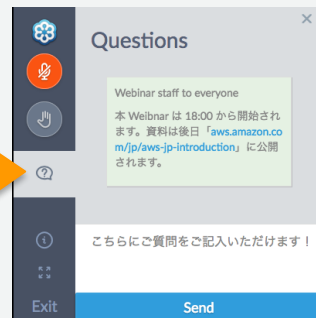
# AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は  
お答えできませんのでご了承下さい

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください  
#awsblackbelt

# 内容についての注意点

- 本資料では2019年02月06日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

# 本日のアジェンダ

- 機械学習システムでよくある課題
- Amazon SageMaker を動かしてみる
- Amazon SageMaker の概要
- SageMaker SDK による開発の流れ
- コンポーネント詳細 [開発 | 学習 | 推論]
- Amazon SageMaker 活用法
- その他

# 機械学習システムでよくある問題

# 典型的な機械学習の流れ

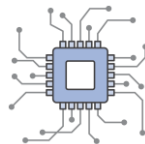
## 開発

学習に使うコードを記述  
小規模データで動作確認



## 学習

大量の GPU  
大規模データの処理  
試行錯誤の繰り返し



## 推論

大量の CPU や GPU  
継続的なデプロイ  
様々なデバイスで動作



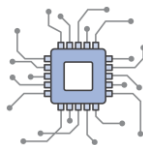
# 典型的な機械学習の流れ

## 開発

データサイエンティストが開発環境で作業  
開発と学習を同じ1台のインスタンスで実施  
Deep Learning であれば GPU インスタンスを使用



## 学習



## 推論

エンジニアがプロダクション環境に構築  
API サーバにデプロイ  
エッジデバイスで動作





# 典型的な機械学習の流れ

## 開発 & 学習

- 環境構築が大変
- 複数の学習ジョブを並列で実行するのが大変
- 複数マシンを使った分散学習を実現するのが大変
- 学習結果を管理するのが大変

学習

推論

エンジニアがプロダク  
ツの構築  
API サーバにデプロイ  
エッジデバイスで動作

## 推論

- 推論用の API サーバ構築とメンテが大変
- エッジデバイスへのデプロイが大変
- バッチ推論の仕組みを構築するのが面倒

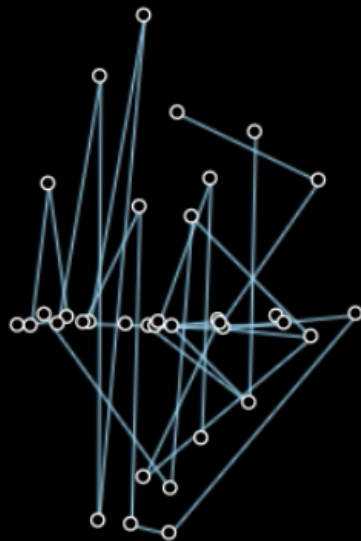


# Amazon SageMaker

すべての開発者とデータサイエンティストのための Machine Learning。

Amazon SageMaker の使用を開始する

Amazon SageMaker は、すべての開発者とデータサイエンティストに Machine Learning モデルの構築、トレーニング、デプロイ手段を提供します。Amazon SageMaker は、機械学習のワークフロー全体をカバーする完全マネージド型サービスです。データをラベル付けして準備し、アルゴリズムを選択してトレーニングを行い、デプロイのための調整と最適化を行い、実行します。モデルをより少ない労力と費用で、本番稼働させることができます。



# Amazon SageMaker を動かしてみ

# ボタンをクリックして 開発用のノートブック インスタンスを作成

Amazon SageMaker ×

- ダッシュボード
- 検索ページ
- ▼ Ground Truth
  - ラベリングジョブ
  - データセットのラベリング
  - ラベリングワークフォース
- ▼ ノートブック
  - ノートブックインスタンス
  - ライフサイクル設定
  - Git リポジトリ
- ▼ トレーニング
  - アルゴリズム
  - トレーニングジョブ
  - ハイパーパラメータの調整ジョブ
- ▼ 推論
  - モデルパッケージ
  - モデル
  - エンドポイント設定
  - エンドポイント
  - バッチ変換ジョブ
- AWS Marketplace

## 構築、トレーニング、デプロイ

アイデアから本番稼働に ML モデルを移行するための最も迅速で簡単な方法です。

### 仕組み



#### ラベル

アクティブラーニングおよび人間によるラベリングを使用して、Amazon SageMaker 内で非常に正確なトレーニングデータセットのラベリングジョブをセットアップおよび管理する



#### ビルド

他の AWS のサービスに接続し、Amazon SageMaker ノートブックでデータを変換する



#### トレーニング

分散型トレーニングのために、Amazon SageMaker のアルゴリズムとフレームワークを使用するか、お客様独自のものを持ち込む

### 今すぐ始める

ノートブックの AWS データを参照し、トレーニングジョブを通じてアルゴリズムを駆使したモデルの作成を行います。クラウド上のノートブックインスタンスを使用して開始します。

[ノートブックインスタンスの作成](#)

[概要から開始](#)

### 料金 (米国)

Amazon SageMaker では、使用した分のみ料金が発生します。オーサリング、トレーニング、ホスティングは秒単位で課金され、最低料金や前払いの義務はありません。

[詳細情報](#)

### 関連サービス

[AWS Glue](#)

[Amazon EC2](#)

[Amazon Elastic Block Store \(EBS\)](#)

### その他のリソース

## ノートブックインスタンスの作成

Amazon SageMaker は、Jupyter ノートブックを実行する構成済みでフルマネージド型のノートブックインスタンスを提供します。ノートブックインスタンスには、一般的なモデルトレーニングおよびホスティングの演習のためのコード例が用意されています。 [詳細はこちら](#)

ノートブックインスタンス設定

ノートブックインスタンス名

ノートブックインスタンスのタイプ  
ml.t2.medium

Elastic Inference [詳細情報](#)

なし

IAM ロール  
AmazonSageMakerFullAccess IAM ポリシーがアタッチされたロールを自動的に作成します。  
AmazonSageMaker-ExecutionRole-20181218T153399

VPC - オプション  
VPC が指定されていないため、ノートブックインスタンスには、SageMaker で提供されたインターネットアクセスが提供されます。  
非 VPC

ライフサイクル設定 - オプション  
デフォルトのスケジューリングとプラグインでノートブック環境をカスタマイズします。  
設定なし

暗号化キー - オプション  
ノートブックデータを暗号化します。既存の KMS キーを選択するか、キーの ARN を入力します。  
カスタム暗号化なし

ボリュームサイズ (GB 単位) - オプション  
ノートブックインスタンスのボリュームサイズ (GB 単位)。最小 5 GB。最大 16384 GB (16 TB)。  
5

▼ Git リポジトリ - オプション

▼ デフォルトのリポジトリ

リポジトリ  
Jupyter はこのリポジトリで開始されます。リポジトリは、ホームディレクトリに追加されます。  
なし

追加のリポジトリを追加

▼ タグ - オプション

キー 値 削除

タグの追加

キャンセル ノートブックインスタンスの作成

オプションはたくさんあるが、以下の3つだけ指定すれば OK

- ノートブックインスタンス名
- ノートブックインスタンスのタイプ (デフォルトで ml.t2.medium 指定済)
- IAM ロール (次ページ参照)

上記設定を終えたら、インスタンスの作成ボタンをクリック

IAM ロール

ノートブックインスタンスでは、SageMaker と S3 を含む他のサービス呼び出すアクセス許可が必要です。ロールを選択するか、AmazonSageMakerFullAccess IAM ポリシーがアタッチされたロールを自動的に作成します。

AmazonSageMaker-ExecutionRole-20181218T133399

新しいロールの作成

カスタム IAM ロールの ARN の入力

既存のロールの使用

IAM ロールの項目をクリックして「新しいロールの作成」を選択

IAM ロールを作成する

IAM ロールを選すと、お客様に代わって他の AWS のサービスでアクションを実行するアクセス許可が Amazon SageMaker に与えられます。ここでロールを作成すると、AmazonSageMakerFullAccess 作成する IAM ポリシーで記述されたアクセス許可が付与されます。

作成する IAM ロールにより、以下へのアクセスが提供されます。

指定する S3 バケット - オプション

- 特定の S3 バケット
- 任意の S3 バケット
- なし

名前に「sagemaker」が含まれる任意の S3 バケット

名前に「sagemaker」が含まれる任意の S3 オブジェクト

タグ「sagemaker」と値「true」が含まれる任意の S3 オブジェクト

SageMaker へのアクセスを許可するバケットポリシーを持つ S3 バケット

キャンセル    **ロールの作成**

指定する S3 バケットで「任意の S3 バケット」を選択して「ロールの作成」をクリック

\* 実際に運用を行う際には、適切な権限を付与した IAM ロールを作成することを推奨します  
ここでご紹介している手順は、あくまでクイックスタートとお考えください



数分待つと、ステータスが InService になるので、  
「開く Jupyter」を押して、Jupyter Notebook の画面にアクセス

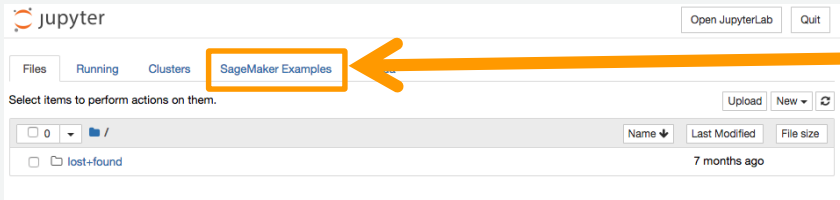
ノートブックインスタンス

アクション ▼ ノートブックインスタンスの作成

Q ノートブックインスタンス を検索

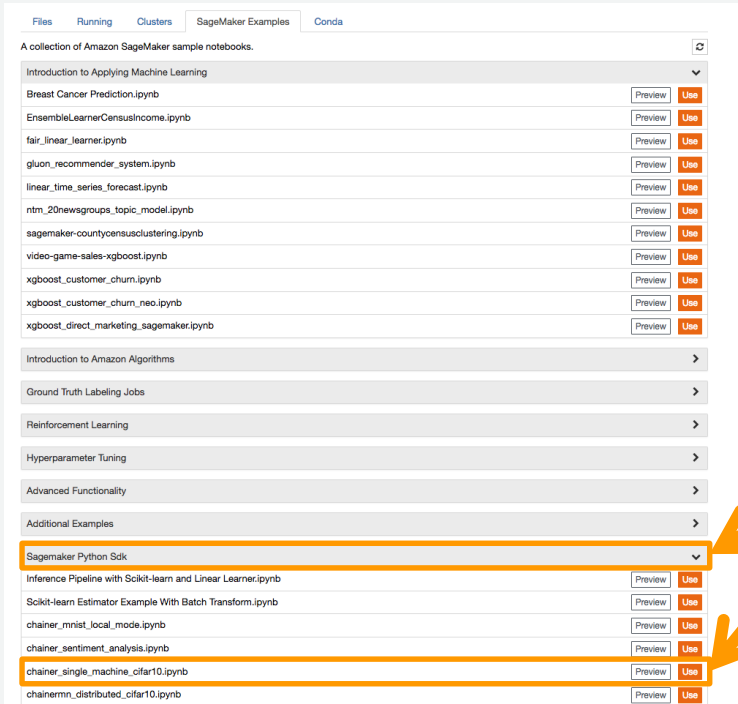
	名前	インスタンス	作成時刻	ステータス	アクション
<input type="radio"/>	dev-instance-m5-4xlarge	mL.m5.4xlarge	Dec 04, 2018 04:18 UTC	✔ InService	開く Jupyter   開く JupyterLab
<input type="radio"/>	dev-p3	mL.p3.2xlarge	Nov 21, 2018 01:51 UTC	✔ InService	開く Jupyter   開く JupyterLab
<input type="radio"/>	dev-t2	mL.t2.medium	Nov 21, 2018 01:50 UTC	⊖ Stopped	開始
<input type="radio"/>	aws-glue-sagemaker2	mL.t2.medium	Oct 07, 2018 06:54 UTC	✔ InService	開く Jupyter   開く JupyterLab
<input type="radio"/>	pyspark-notebook	mL.t2.medium	May 17, 2018 08:41 UTC	✔ InService	開く Jupyter   開く JupyterLab

\* 「開く JupyterLab」を選択すると、JupyterLab の画面が開きます。お好きなほうでご利用ください



Jupyter Notebook 画面が開くので、  
ここから自由に開発を実施可能

また「SageMaker Examples」タブに  
多数のサンプルノートブックが配置



「SageMaker Python Sdk」セクションを  
クリックすると、ノートブックが表示

「chainer\_single\_machine\_cifar10.ipynb」の  
右にある「Use」をクリックして、  
ポップアップの「Create copy」を選択



jupyter chainer\_single\_machine\_cifar10 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted conda\_chainer\_p36

### Training with Chainer

[VGG](#) is an architecture for deep convolution networks. In this example, we train a convolutional network to perform image classification using the CIFAR-10 dataset. CIFAR-10 consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We'll train a model on SageMaker, deploy it to Amazon SageMaker hosting, and then classify images using the deployed model.

The Chainer script runs inside of a Docker container running on SageMaker. For more information about the Chainer container, see the [sagemaker-chainer-containers](#) repository and the [sagemaker-python-sdk](#) repository:

- <https://github.com/aws/sagemaker-chainer-containers>
- <https://github.com/aws/sagemaker-python-sdk>

For more on Chainer, please visit the Chainer repository:

- <https://github.com/chainer/chainer>

This notebook is adapted from the [CIFAR-10](#) example in the Chainer repository.

```
In [ ]: # Setup
from sagemaker import get_execution_role
import sagemaker

sagemaker_session = sagemaker.Session()

# This role retrieves the SageMaker-compatible role used by this Notebook Instance.
role = get_execution_role()
```

### Downloading training and test data

We use helper functions provided by `chainer` to download and preprocess the CIFAR10 data.

```
In [ ]: import chainer

from chainer.datasets import get_cifar10

train, test = get_cifar10()
```

開いたノートブックに、Chainer で 1 台のマシンで、SageMaker の学習・推論機能を使った例が、説明と合わせて記載されている

ノートブックを順に実行していけば Sagemaker で Chainer による画像認識の学習を行い、作成されたモデルをエンドポイントにして推論を行うことができる

```
from sagemaker.chainer.estimator import Chainer
```

```
chainer_estimator = Chainer(entry_point='chainer_cifar_vgg_sample_machine.py',  
                             source_dir="src",  
                             role=role,  
                             sagemaker_session=sagemaker_session,  
                             train_instance_count=1,  
                             train_instance_type='ml.p2.xlarge',  
                             hyperparameters={'epochs': 50, 'batch-size': 64})
```

```
chainer_estimator.fit({'train': train_input, 'test': test_input})
```

Sagemaker SDK が提供されており  
SDK 経由でジョブを実行可能

SDK は github で公開されている\*

開発した Chainer コードを指定して  
`estimator.fit()` で学習を実行

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

```
response = predictor.predict(image_data)
```

```
for i, prediction in enumerate(response):  
    print('image {}: prediction: {}'.format(i, prediction.argmax(axis=0)))
```

学習が終わったら、  
`estimator.deploy()` で推論用の  
エンドポイントを作成

`predictor.predict()` で実際に  
推論を実行することが可能

# Amazon SageMaker の概要

# Amazon SageMaker とは

- 機械学習システムでよくある問題を解消し、データサイエンティストやエンジニアが素早くプロセスを回せるようにするためのサービス
- 機械学習のインフラ構築・運用を自動化するだけでなく、そのほかのさまざまな機能も提供
- **東京リージョン**を含む、13 リージョンにてサービスを展開



ラベリング



開発



学習



モデル変換



推論

# Amazon SageMaker の各コンポーネント



**ラベリング**：機械学習のためのインプットデータ作成を支援する、ウェブベースのツールを提供。画像や文章などに対して効率的にラベル付けを行えるようになる



**開発**：学習するためのコードの記述や、入力データの加工整形を行うための環境として、Jupyter Notebook や機械学習ライブラリ群がインストール済みのインスタンスを提供



**学習**：API を叩くと学習用インスタンスが立ち上がり、学習ジョブを実行。複数ジョブの同時実行や、複数インスタンスでの分散学習、ハイパーパラメータチューニングに対応



**モデル変換**：学習済みのモデルに対して、実行環境に最適化された形でのモデル変換機能を提供。EC2 インスタンスやエッジデバイスに最適化された形のモデルを作成可能



**推論**：API を叩くと、指定したモデルを読み込んで、オートスケーリングや AB テストに対応した API エンドポイントが作成される。大量データをバッチで推論する処理もサポート

# Amazon SageMaker の各コンポーネント



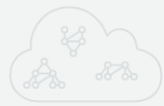
**ラベリング**：機械学習のためのインプットデータ作成を支援する、ウェブベースのツールを提供。画像や文章などに対して効率的にラベル付けを行えるようになる



**開発**：学習するためのコードの記述や、入力データの加工整形を行うための環境として、Jupyter Notebook や機械学習ライブラリ群がインストール済みのインスタンスを提供



**学習**：API を叩くと学習用インスタンスが立ち上がり、学習ジョブを実行。複数ジョブの同時実行や、複数インスタンスでの分散学習、ハイパーパラメータチューニングに対応

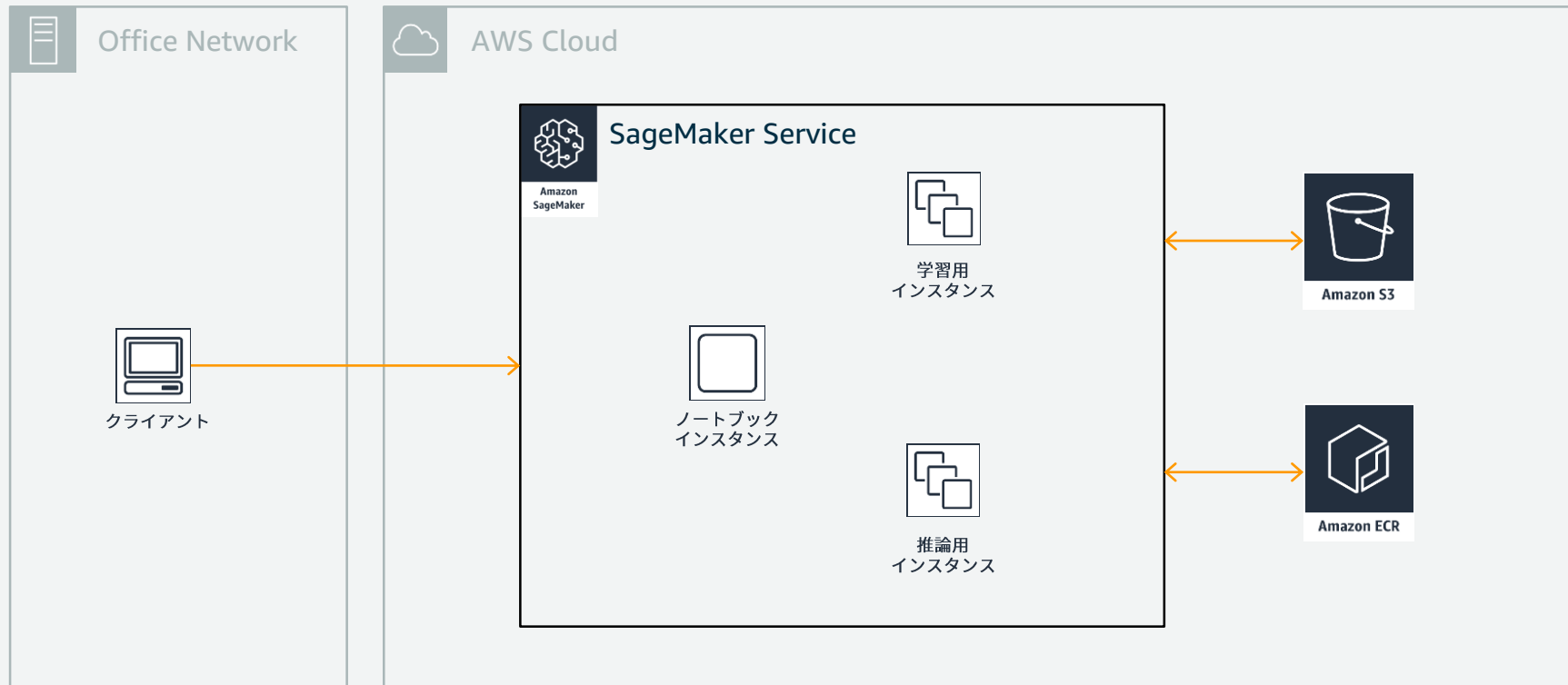


**モデル変換**：学習済みのモデルに対して、実行環境に最適化された形でのモデル変換機能を提供。EC2 インスタンスやエッジデバイスに最適化された形のモデルを作成可能

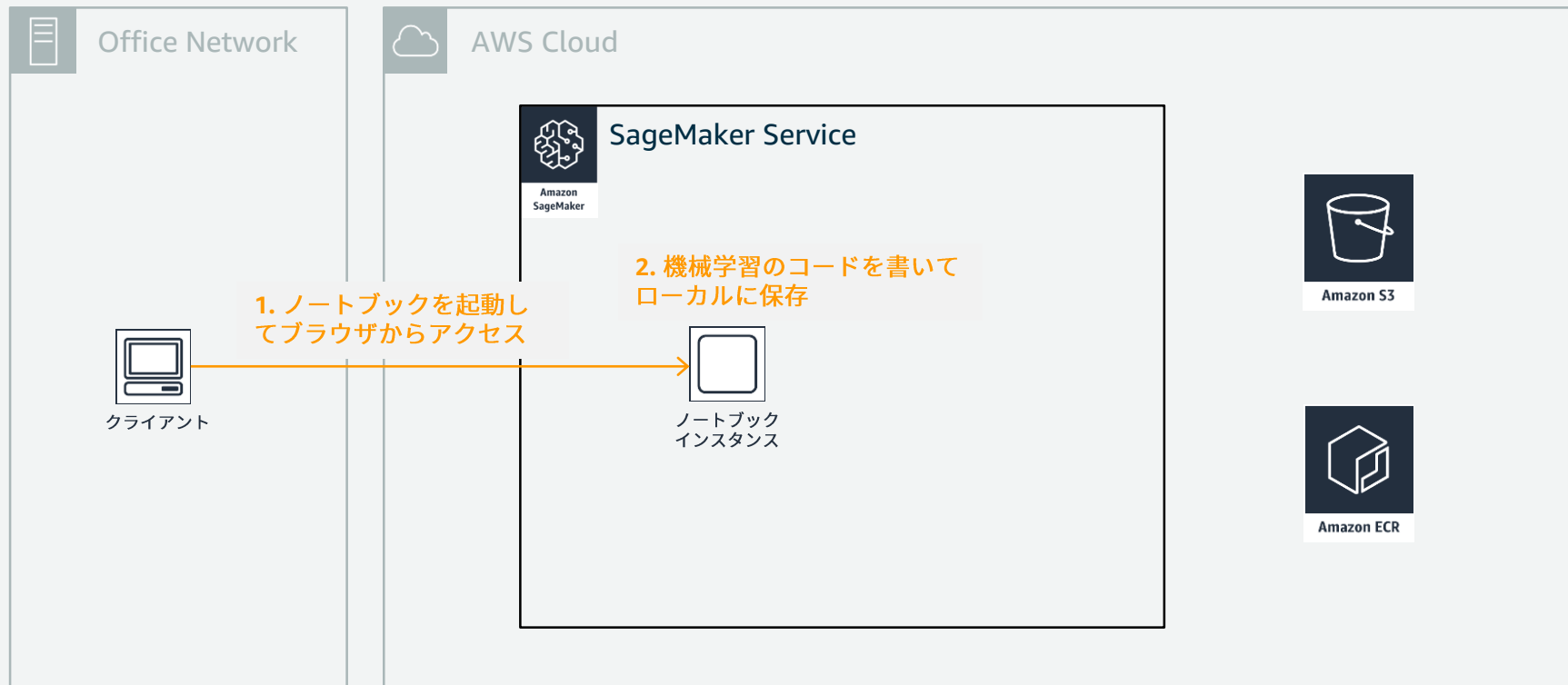


**推論**：API を叩くと、指定したモデルを読み込んで、オートスケーリングや AB テストに対応した API エンドポイントが作成される。大量データをバッチで推論する処理もサポート

# Amazon SageMaker のアーキテクチャ

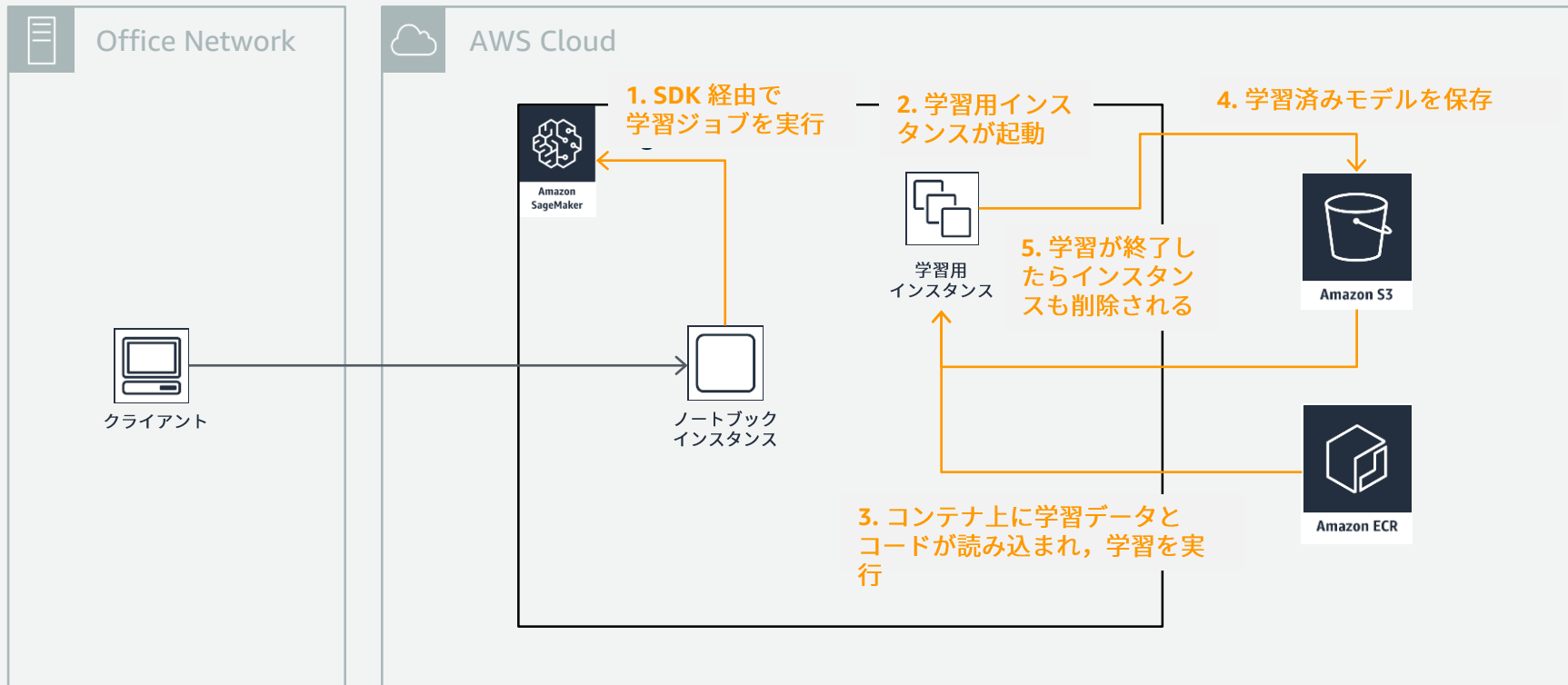


# Amazon SageMaker による開発の流れ

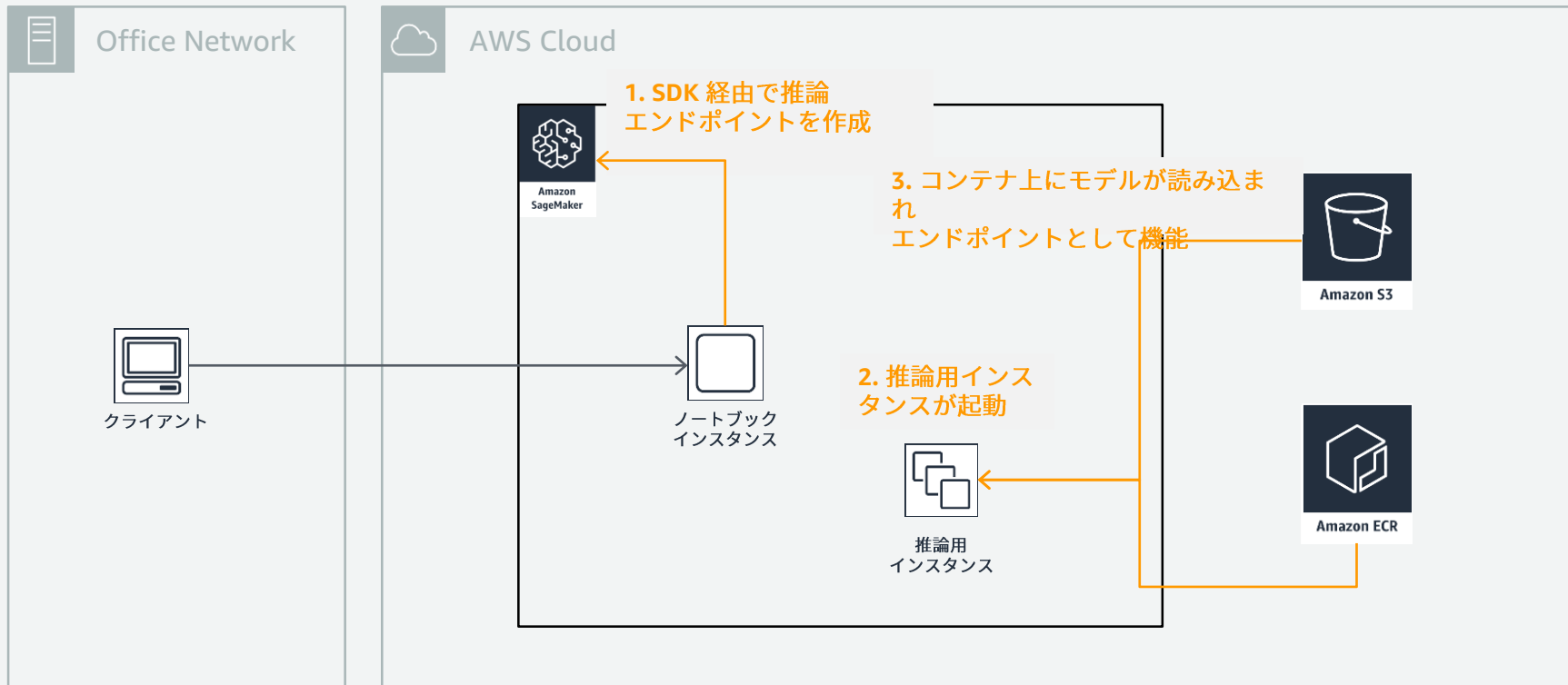




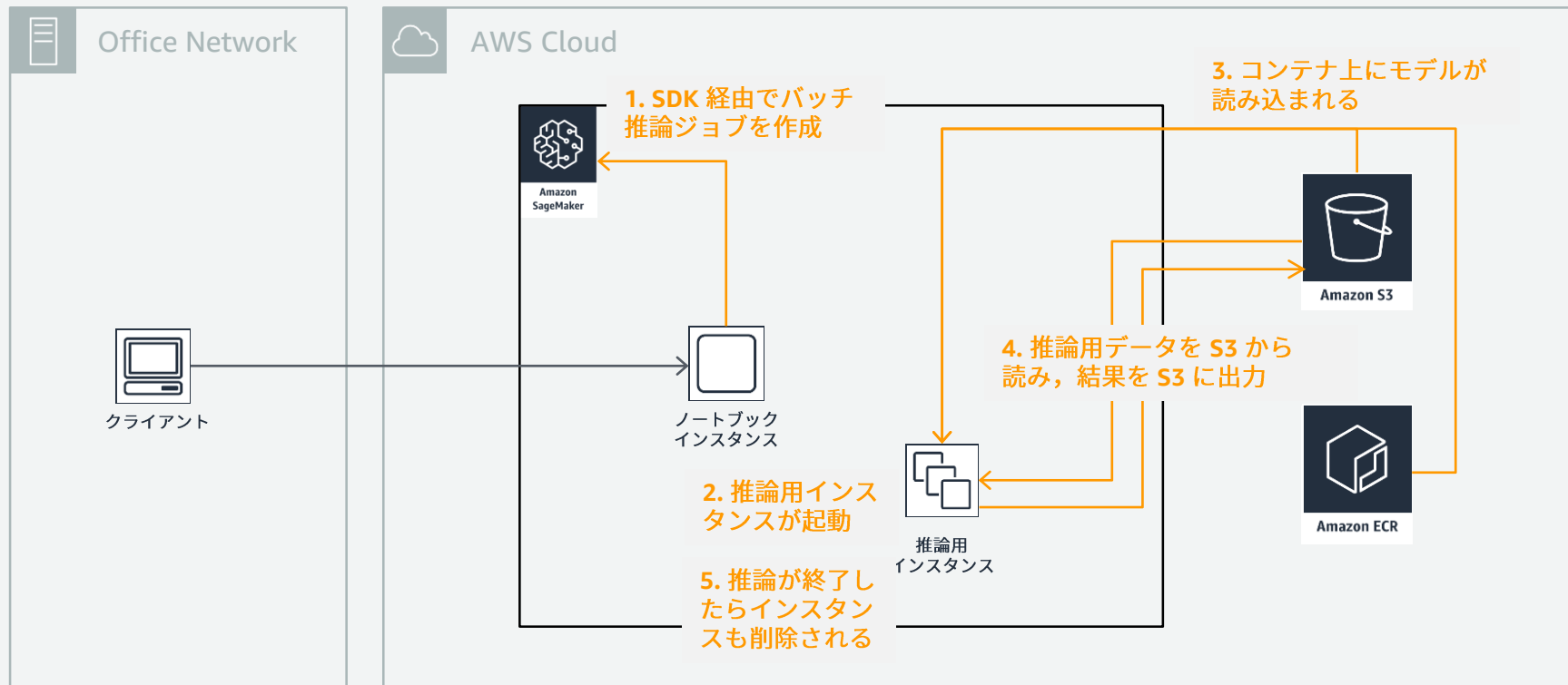
# Amazon SageMaker による学習の流れ



# Amazon SageMaker によるエンドポイント推論の流れ



# Amazon SageMaker によるバッチ推論の流れ



# SageMaker SDK による開発の流れ

# SageMaker Python SDK & Examples

- github にて、SDK のコードおよびドキュメントが公開されている
- SDK を使ったノートブックのサンプルも同様に、多数 github 上に公開



Amazon SageMaker

## SageMaker Python SDK

**Build** **Python** **Release** **2020**

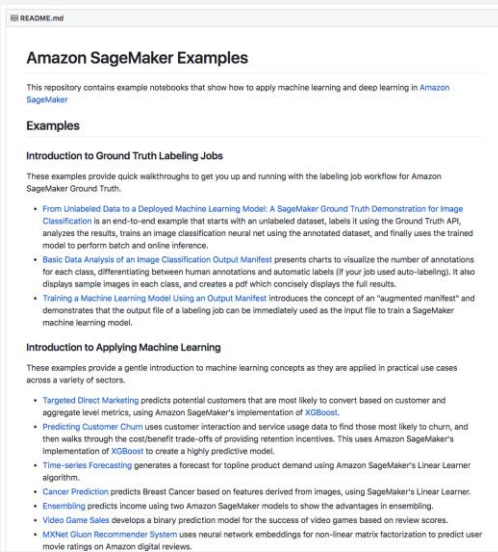
SageMaker Python SDK is an open source library for training and deploying machine learning models on Amazon SageMaker.

With the SDK, you can train and deploy models using popular deep learning frameworks Apache MXNet and TensorFlow. You can also train and deploy models with Amazon algorithms, which are scalable implementations of core machine learning algorithms that are optimized for SageMaker and GPU training. If you have your own algorithms built into SageMaker compatible Docker containers, you can train and host models using these as well.

For detailed API reference please go to: [Read the Docs](#)

### Table of Contents

1. Installing SageMaker Python SDK
2. SageMaker Python SDK Overview
3. MXNet SageMaker Estimators
4. TensorFlow SageMaker Estimators
5. Chainer SageMaker Estimators
6. PyTorch SageMaker Estimators
7. Scikit-learn SageMaker Estimators
8. SageMaker Reinforcement Learning Estimators
9. SageMaker SparkML Serving
10. AWS SageMaker Estimators
11. Using SageMaker Algorithm Estimators
12. Consuming SageMaker Model Packages
13. BYO Docker Containers with SageMaker Estimators
14. SageMaker Automatic Model Tuning
15. SageMaker Batch Transform
16. Secure Training and Inference with VPC
17. BYO Model
18. Inference Pipelines
19. SageMaker Workflow



## Amazon SageMaker Examples

This repository contains example notebooks that show how to apply machine learning and deep learning in Amazon SageMaker.

### Examples

#### Introduction to Ground Truth Labeling Jobs

These examples provide quick walkthroughs to get you up and running with the labeling job workflow for Amazon SageMaker Ground Truth.

- [From Unlabeled Data to a Deployed Machine Learning Model: A SageMaker Ground Truth Demonstration for Image Classification](#) is an end-to-end example that starts with an unlabeled dataset, labels it using the Ground Truth API, analyzes the results, trains an image classification neural net using the annotated dataset, and finally uses the trained model to perform batch and online inference.
- [Basic Data Analysis of an Image Classification Output Manifest](#) presents charts to visualize the number of annotations for each class, differentiating between human annotations and automatic labels (if your job used auto-labeling). It also displays sample images in each class, and creates a pdf which concisely displays the full results.
- [Training a Machine Learning Model Using an Output Manifest](#) introduces the concept of an "augmented manifest" and demonstrates that the output file of a labeling job can be immediately used as the input file to train a SageMaker machine learning model.

#### Introduction to Applying Machine Learning

These examples provide a gentle introduction to machine learning concepts as they are applied in practical use cases across a variety of sectors.

- [Targeted Direct Marketing](#) predicts potential customers that are most likely to convert based on customer and aggregate level metrics, using Amazon SageMaker's implementation of XGBoost.
- [Predicting Customer Churn](#) uses customer interaction and service usage data to find those most likely to churn, and then walks through the cost/benefit trade-offs of providing retention incentives. This uses Amazon SageMaker's implementation of XGBoost to create a highly predictive model.
- [Time-series Forecasting](#) generates a forecast for topline product demand using Amazon SageMaker's Linear Learner algorithm.
- [Cancer Prediction](#) predicts Breast Cancer based on features derived from images, using SageMaker's Linear Learner.
- [Ensembling](#) predicts income using two Amazon SageMaker models to show the advantages in ensembling.
- [Video Game Sales](#) develops a binary prediction model for the success of video games based on review scores.
- [MXNet Gluon Recommender System](#) uses neural network embeddings for non-linear matrix factorization to predict user movie ratings on Amazon digital reviews.

<https://github.com/aws/sagemaker-python-sdk>  
<https://github.com/aws-labs/amazon-sagemaker-examples>

# SageMaker SDK の基本コード

```
from sagemaker.chainer.estimator import Chainer
```

```
estimator = Chainer(entry_point='mnist.py',  
                    train_instance_type='ml.p2.xlarge',  
                    train_instance_count=1)
```

```
estimator.fit({'train': train_input})
```

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

```
response = predictor.predict(test_image)
```

```
transformer = estimator.transformer(instance_type='ml.m4.xlarge',  
                                    initial_instance_count=1)
```

```
transformer.transform(test_data,  
                    content_type='text/csv')
```

ジョブを実行するためにまず、Estimator クラスのオブジェクトを作成  
Chainer の場合は、専用の Estimator が  
ある。ローカルにある開発済みのスクリ  
プトを指定

fit() を実行すると、指定したインスタ  
ンスが立ち上がり、用意された Chainer  
コンテナを読み込み、S3 データを使って  
学習ジョブを実行する

学習が終わったら、deploy() メソッドを  
叩くと、裏側でエンドポイントが作成さ  
れる。predict() で実際に推論を実施可  
能

バッチ推論は transformer.transform()  
で実行。S3 から対象データを読み込んで、  
推論結果も S3 にファイル出力

# 実行スクリプトの基本的な書き方

```
class MLP(chainer.Chain):
    def __init__(self, n_units, n_out):
        super(MLP, self).__init__()
        ...

    def __call__(self, x):
        h1 = F.relu(self.l1(x))
        h2 = F.relu(self.l2(h1))
        return self.l3(h2)

if __name__ == '__main__':

    parser = argparse.ArgumentParser()
    parser.add_argument('--epochs', type=int, default=50)
    ...
    parser.add_argument('--test', type=str, default=os.environ['SM_CHANNEL_TEST'])
    args, _ = parser.parse_known_args()

    train_data = np.load(os.path.join(args.train, 'train.npz'))['images']
    ...

    model = L.Classifier(MLP(1000, 10))
    optimizer = chainer.optimizers.Adam()
    optimizer.setup(model)
    ...
    trainer.run()
    serializers.save_npz(os.path.join(args.model_dir, 'model.npz'), model)

def model_fn(model_dir):
    model = L.Classifier(MLP(1000, 10))
    serializers.load_npz(os.path.join(model_dir, 'model.npz'), model)
    return model.predictor
```

Chainerの場合、基本的には main 関数の中に学習処理をベタ書きすれば OK  
ハイパーパラメータや入力データのパス等は、SageMaker 側で引数として引き渡してくれるので、それを argparse で取り出すだけ

推論エンドポイントにおけるモデルのロード処理を、model\_fn() 内に記述しておく。あとは SageMaker で提供される推論コンテナが、そのモデルを使ってくれる

# SDK でサポートされるアルゴリズム・フレームワーク

種類	フレームワーク	コンテナの準備	スクリプトの準備	データの準備
ビルトイン・アルゴリズム	-	不要	不要	必要
マーケットプレイス	-	不要	不要	必要
Deep Learning フレームワーク	Tensorflow (含 Keras) Chainer PyTorch MXNet (含 Keras)	不要	必要	必要
機械学習 フレームワーク	scikit-learn	不要	必要	必要
強化学習 フレームワーク	Coach Ray	不要	必要	必要
独自アルゴリズム	-	必要	必要	必要



# SDK でサポートされるアルゴリズム・フレームワーク

種類	フレームワーク	コンテナの準備	スクリプトの準備	データの準備
ビルトイン・アルゴリズム	-	不要	不要	必要
マーケットプレイス	-	不要	不要	必要
Deep Learning フレームワーク	Tensorflow (含 Keras) Chainer PyTorch MXNet (含 Keras)	不要	必要	必要
機械学習 フレームワーク	scikit-learn	不要	必要	必要
強化学習 フレームワーク	Coach Ray	不要	必要	必要
独自アルゴリズム	-	必要	必要	必要

# ビルトイン・アルゴリズム

- Linear Learner
- Factorization Machines
- XGBoost
- Image Classification
- seq2seq
- K-means
- k-NN
- Object2Vec
- Semantic Segmentation
- PCA
- LDA
- Neural Topic Model
- DeepAR Forecasting
- BlazingText (word2vec)
- Random Cut Forest
- Object Detection
- IP Insights

<https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html>

# ビルトイン・アルゴリズムの使い方

```
from sagemaker.estimator import Estimator

estimator = Estimator(container,
                      train_instance_count=1,
                      train_instance_type='ml.c4.xlarge')

estimator.fit({'train': s3_train_data})

predictor = estimator.deploy(initial_instance_count=1,
                             instance_type='ml.m4.xlarge')

result = predictor.predict(test=data)
```

アルゴリズムごとに、スクリプトまで含まれたコンテナがあらかじめ用意されているので、そのコンテナ ID を指定。コンテナ ID はドキュメントに記載

[https://docs.aws.amazon.com/ja\\_jp/sagemaker/latest/dg/sagemaker-algo-docker-registry-paths.html](https://docs.aws.amazon.com/ja_jp/sagemaker/latest/dg/sagemaker-algo-docker-registry-paths.html)

# Tensorflow 等のフレームワーク

- Tensorflow, Chainer, PyTorch, MXNet, scikit-learn については、実行用コンテナがあり、SDK にも専用のクラスが用意されている
- Tensorflow, MXNet コンテナで Keras モデルを記述することも可能
  - Keras での記述については AWS blog のエントリーを参照\*
- 基本は **main 関数に処理をベタ書き** するだけ。詳細な仕様については、github 上のドキュメントを参照
- 実行用コンテナもすべて github 上に公開されている

```
estimator = Tensorflow(entry_point='mnist.py' ...)  
estimator = Chainer(entry_point='mnist.py' ...)  
estimator = PyTorch(entry_point='mnist.py' ...)  
estimator = MXNet(entry_point='mnist.py' ...)  
estimator = SKLearn(entry_point='mnist.py' ...)
```

<https://github.com/aws/sagemaker-python-sdk>

<https://github.com/aws/sagemaker-containers>

\* <https://aws.amazon.com/jp/blogs/news/amazon-sagemaker-keras/>

# 独自アルゴリズムの使用

SageMaker では ECR に配置した任意のコンテナを使って学習・推論を行うことが可能。インターフェースとして、学習時は `docker run $IMAGE_ID train` を、推論時は同様に `serve` を、コンテナ側で用意する必要あり

```
from sagemaker.estimator import Estimator

estimator = Estimator(container,
                      train_instance_count=1,
                      train_instance_type='ml.c4.xlarge')

estimator.fit({'train': s3_train_data})

predictor = estimator.deploy(initial_instance_count=1,
                             instance_type='ml.m4.xlarge')

result = predictor.predict(test=data)
```

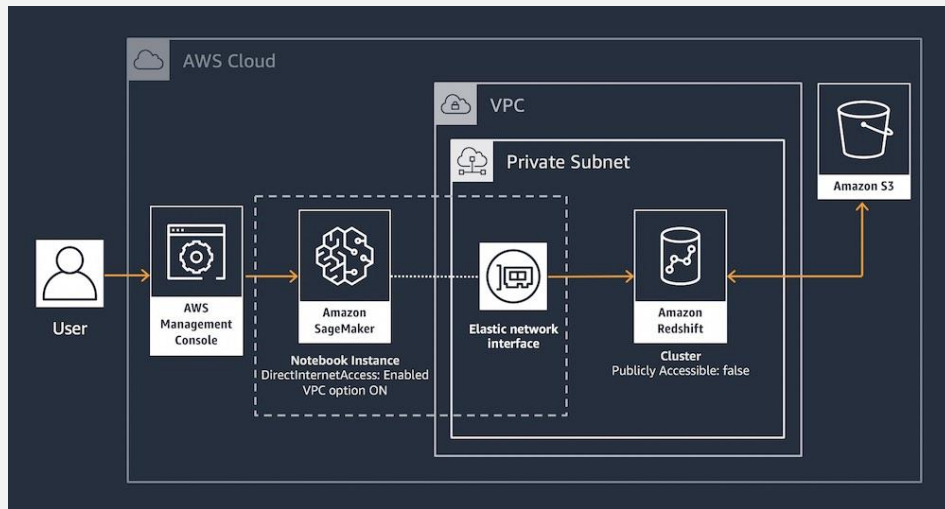
ECR にプッシュした自作コンテナの ID を指定する

<https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-training-algo.html>  
<https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-inference-code.html>

# コンポーネント詳細 [開発]

# ノートブック上で前処理やプロトタイピング

- t2 から p3 まで，幅広いインスタンスタイプを選択可能
- ストレージサイズも 5GB-16TB の間で指定可能
- ノートブック上でトイデータを使って，開発・学習・推論をクイックに行い，問題のあたりをつけることが可能
- Redshift, Athena, EMR 等と連携して，ノートブック上で学習データの前処理を行うこともできる



<https://aws.amazon.com/jp/blogs/news/build-amazon-sagemaker-notebooks-backed-by-spark-in-amazon-emr/>

<https://aws.amazon.com/jp/blogs/news/build-fast-flexible-secure-machine-learning-platform-using-amazon-sagemaker-and-amazon-redshift/>

# git インテグレーション

- SageMaker に git リポジトリを登録することが可能
- ノートブックを起動する際に，登録済みリポジトリをアタッチしておくことで，リポジトリが最初から含まれた状態でノートブックを利用可能
- JupyterLab であれば，GUI での操作も可能

Amazon SageMaker > Git repositories

Git repositories				
<input type="text" value="Search git repositories"/>				<input type="button" value="Delete"/> <input type="button" value="Edit"/> <input type="button" value="Add repository"/>
<input type="text" value="1"/>				
Name	URL	ARN	Creation time	
<input type="radio"/> <a href="#">sagemaker-handson</a>	<a href="https://git-codecommit.us-east-1.amazonaws.com/v1/repos/sagemaker-handson">https://git-codecommit.us-east-1.amazonaws.com/v1/repos/sagemaker-handson</a>	arn:aws:sagemaker:us-east-1:666254511816:code-repository/sagemaker-handson	Nov 28, 2018 16:43 UTC	

```
File Edit View Run Kernel Git Tabs Settings Help
sample_mecab_word2vec / master
History master v +
Input message to commit staged changes
Staged(1)
README.md
Changed(0)
Untracked(1)
.ipynb_checkpoints/
Terminal 1
remote: Total 37 (delta 13), reused 37 (delta 13), done.
sh-4.2$ ls
anaconda2  examples  README
anaconda3  Nvidia_Cloud_EULA.pdf SageMaker
sh-4.2$ cd sample_mecab_word2vec/
sh-4.2$ ls
anaconda2  examples  README
anaconda3  Nvidia_Cloud_EULA.pdf SageMaker
sh-4.2$ cd SageMaker/
sh-4.2$ cd sample_mecab_word2vec/
sh-4.2$ ls
corpus.py  dicformat.py  lib  LICENSE  README
sh-4.2$ vim README.md
sh-4.2$ git st
git: 'st' is not a git command. See 'git --help'

The most similar commands are
status
reset
stage
stash
sh-4.2$ git status
On branch master
Your branch is up-to-date with 'origin/master'
```



# ライフサイクル設定

- インスタンス起動時と開始時のセットアップ処理を自動化可能
  - 開始時は、インスタンスを停止→再開した際にも実行される
  - インスタンス作成時に、ライフサイクル設定を付与しておくだけ
- 環境変数の設定や、特定ライブラリのインストールなどの定番処理をまとめておくことで、環境構築を自動化

### Configuration setting

Name

Alphanumeric characters and "-", no spaces. Maximum 63 characters.

### Scripts

Start notebook    **Create notebook**

When selected during creation of a new notebook instance, this script will be run once during its initial creation. This script will not be run on existing notebook instances.

```
1 #!/bin/bash
2 |
3 set -e
4 touch /home/ec2-user/note
5 wget http://bit.ly/sagemaker-handson-notebooks
```

Cancel    Update

# コンポーネント詳細 [学習]

# 分散学習および複数ジョブの同時実行

- ビルトインアルゴリズム, および scikit-learn 以外の対応フレームワークでは, `instance_count` を 2 以上にすることで, 自動的に分散学習環境を構築し, 実行
  - もちろんスクリプトは自分で分散学習対応の形で書く必要あり
- 自前テナでは, テナ内の `/opt/ml/input/config/resourceConfig.json` に, SageMaker がホスト情報を配置するので, それを利用

```
{  
  "current_host": "algo-1",  
  "hosts": ["algo-1", "algo-2", "algo-3"]  
}
```

- SDK から学習を行う際に, `estimator.fit(wait=False)` とすると, ジョブの終了を待たないため, ノートブックから連続でジョブ実行が可能

# ハイパーパラメータのチューニング

- `Estimator` の初期化時に `hyperparameters` で引き渡すパラメータに関して、ベイズ最適化によるパラメータの自動チューニングを実行可能
- SageMaker で行うすべての学習ジョブに対応（自作コンテナの場合でも！）
- ターゲットメトリクスも自由に指定可能

```
tuner = HyperparameterTuner(estimator,
                             objective_metric_name,
                             hyperparameter_ranges,
                             metric_definitions,
                             objective_type='Minimize',
                             max_jobs=9,
                             max_parallel_jobs=3,
                             base_tuning_job_name='hpo-r-byoa-XX')

tuner.fit({'train': 's3://{}/{}/train'.format(bucket, prefix)})
```

<https://github.com/aws/sagemaker-python-sdk#sagemaker-automatic-model-tuning>

# ローカルでのテスト

- Tensorflow 等のフレームワークを利用する場合は，SageMaker で学習・推論を実行する前に，ローカルテストをすることが可能．コンテナをノートブックインスタンスに pull してきて，動作テストを行う形になる
- SageMaker 上で実行するより素早く動作確認ができる
- 事前に nvidia-docker などのインストールの必要あり
- テスト時は，インスタンスタイプを `local` にするだけ

```
# training
estimator = Tensorflow(entry_point='mnist.py', train_instance_type='local', train_instance_count=1)
# inference endpoint
predictor = estimator.deploy(instance_type='local', initial_instance_count=1)
# batch inference
transformer = estimator.transformer(instance_type='local', instance_count=1)
```

<https://github.com/aws/sagemaker-python-sdk#local-mode>

# 学習ジョブ管理のための Search 機能（ベータ）

- アルゴリズム，ハイパーパラメータ設定，学習データ，タグ等で，合致するデータを検索することが可能
- 検索結果を Accuracy や Loss 等のメトリクスでソートすることが可能
- デプロイされたモデルについて，どのデータが使われたかという Linage をトレースすることも可能

Results: Training jobs

HyperParameter mini_batch_size	HyperParameter predictor_type	Metric train:binary_f_beta	Metric train:progress	Metric train:objective_loss	Metric train:binary_classification_accuracy
300	binary_classifier	0.966639518737793	100	0.023814236745238304	0.9934399724006653
100	binary_classifier	0.9652714133262634	100	0.023504912853240967	0.993179976940155
200	binary_classifier	0.9647442698478699	100	0.023259807378053665	0.9930800199508667

Production variants

Model name	Training job	Variant name	Instance type
linear-learner-2018-██████████	linear-learner-2018-██████████	AllTraffic	mL.m4.xlarge

<https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-now-comes-with-new-capabilities-for-accelerating-machine-learning-experimentation/>

# コンポーネント詳細 [推論]

# オートスケーリング

基本はターゲットトラッキングスケーリングポリシーを使用

バリエーション (= エンドポイントにデプロイするモデル) ごとにオートスケーリングポリシーの設定が可能

ターゲットメトリクスは、以下の 2 種類

- 1 インスタンスの分間の平均リクエスト数
- カスタムメトリクス

スケールさせる最小および最大のインスタンス数、クールダウン期間も併せて指定

### Variant automatic scaling [Learn more](#)

Variant name	Instance type	Current instance count	Current weight
AllTraffic	mL.m4.xlarge	1	1

Minimum instance count:  - Maximum instance count:

IAM role  
Amazon SageMaker uses the following service-linked role for automatic scaling. [Learn more](#)  
AWSServiceRoleForApplicationAutoScaling\_SageMakerEndpoint

### Scaling policy [Learn more](#)

Policy name  
SageMakerEndpointInvocationScalingPolicy

Target metric	Target value
SageMakerVariantInvocationsPerInstance	<input type="text" value="300"/>
Scale in cool down (seconds)	Scale out cool down (seconds)
<input type="text" value="3600"/>	<input type="text" value="600"/>

Disable scale in

<https://docs.aws.amazon.com/sagemaker/latest/dg/endpoint-auto-scaling.html#endpoint-auto-scaling-add-policy>  
[https://docs.aws.amazon.com/ja\\_jp/autoscaling/application/userguide/application-auto-scaling-target-tracking.html](https://docs.aws.amazon.com/ja_jp/autoscaling/application/userguide/application-auto-scaling-target-tracking.html)



# A/B テスト

- 複数のモデルそれぞれに、以下のような項目を設定可能
  - インスタンスタイプ
  - インスタンス数
  - リクエスト振分の重み
- エンドポイントからのレスポンスにモデル名（バリエーション名）が含まれるため、クライアント側で、ログに出力

モデル名	バリエーション名	インスタンスタイプ	初期インスタンス数	初期重み
<a href="#">linear-learner-2018-02-28-02-32-38-500</a>	Logistic Regression	mL.m4.xlarge	3	0.8
<a href="#">decision-trees-sample</a>	Decision Tree	mL.c5.9xlarge	3	0.1
<a href="#">sagemaker-tensorflow-py2-cpu-2018-01-22-11-49-31-334</a>	variant-name-3	mL.p3.2xlarge	5	0.1

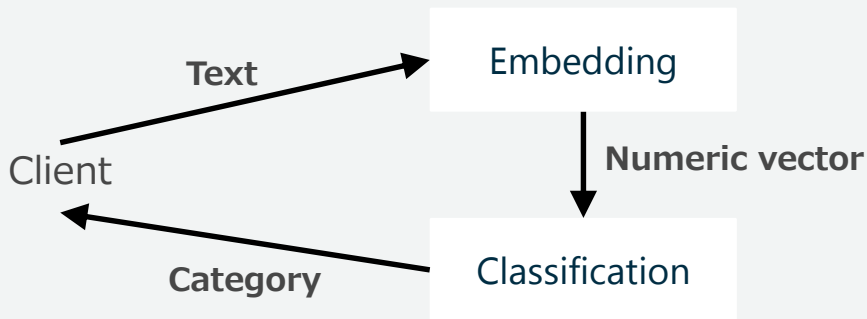
```
HTTP/1.1 200
Content-Type: ContentType
x-Amzn-Invoked-Production-Variant: InvokedProductionVariant

Body
```

# 推論パイプライン

複数の推論エンドポイントを、一連のパイプラインとして定義可能  
前処理用のコンテナ、分類用のコンテナ、後処理用のコンテナといった形で処理をすべて SageMaker の中で記述することができる

例えば以下のように、テキストを受け取って Embedding 処理で数値ベクトル変換し、分類用のコンテナで処理をさらに実施



<https://docs.aws.amazon.com/sagemaker/latest/dg/inference-pipelines.html>

# Elastic Inference によるコスト効率の良い推論

- Elastic Inference は、CPU のみの EC2 インスタンスに、GPU で ML の推論を行うためのアクセラレータをつけることができる機能
  - CPU・メモリ・GPU のサイズを自由に組み合わせることが可能
  - 最大 75% のコスト削減が見込める
- 以下のフレームワークに対応
  - AWS enhanced versions of TensorFlow
  - Apache MXNet (including ONNX)

Accelerator Type	F32 Throughput in TFLOPS	F16 Throughput in TFLOPS	Memory in GB
m1.eia1.medium	1	8	1
m1.d.eia1.large	2	16	2
m1.d.eia1.xlarge	4	32	4

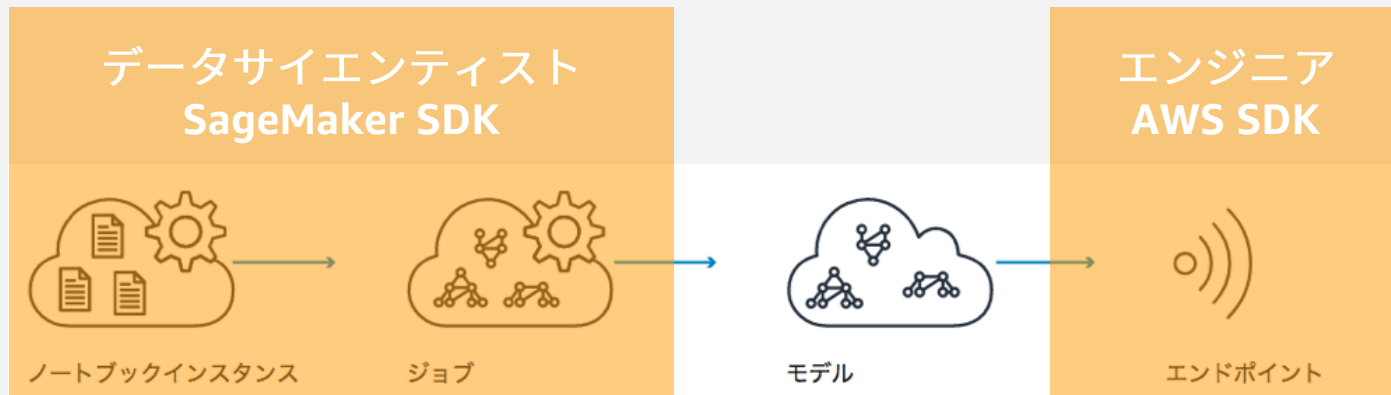
The screenshot shows the 'Edit Production Variant' interface in the AWS SageMaker console. The 'Elastic Inference' dropdown menu is open, displaying a search bar and a list of options: 'none', 'm1.eia1.medium', 'm1.eia1.large', and 'm1.eia1.xlarge'. The 'Save' button is highlighted in orange, indicating it is the active action.

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/working-with-ei.html>

# Amazon SageMaker 活用法

# SageMaker API の利用

- SageMaker SDK は、データサイエンティストの開発のためのもので、基盤エンジニアが運用することはあまり想定していない
- そのため運用においては、AWS SDK を使って SageMaker API を直接叩くことを推奨



# SageMaker の 3 要素は、それぞれ個別で利用可能

例 1: プロダクション環境がオンプレミスにすでにある場合  
スケーラブルな学習環境としてのみ SageMaker を利用可能



例 2: オンプレミスに豊富な GPU クラスタを持っている場合  
オンプレミスで学習済のモデルを AWS 上のプロダクション環境にデプロイ

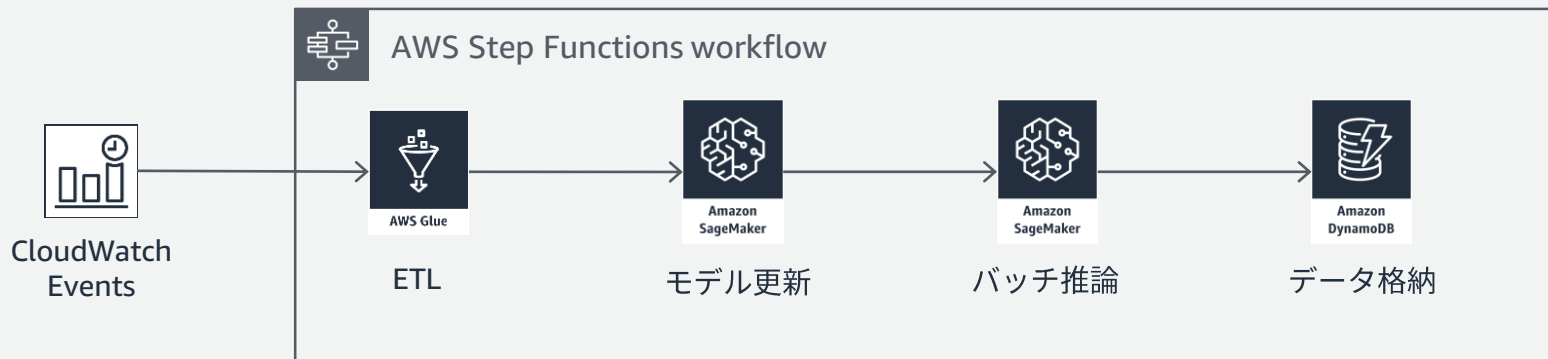


例 3: 部署全員に対してマネージドノートブック環境を提供したい場合  
管理不要のノートブックをホストする環境として SageMaker を利用



# Step Functions による ML ワークフロー

- Step Functions のステートマシンからは，SageMaker や AWS Glue に対して，Lambda を介することなく直接操作を行うことが可能
- Glue による前処理 – SageMaker で学習 – バッチ推論 – データ更新といった一連のパイプラインを，サーバレスの形で実現することが可能



<https://docs.aws.amazon.com/step-functions/latest/dg/connectors-sagemaker.html>

# 価格

- **オンデマンド ML インスタンス**
  - SageMaker の開発・学習・推論の各パートごとに、利用したインスタンスの料金が、従量課金として請求される（最低実行時間 1 分間）
- **ML 汎用ストレージ**
  - インスタンスにアタッチしたストレージの料金
  - バージニア北部リージョンで、0.14 USD/GB/月
- **データ処理量**
  - 開発・推論時の各インスタンスに対する入出力データの量に応じて課金
  - バージニア北部リージョンで、0.016 USD/GB



# 参考文献

## SageMaker Example Notebooks

- <https://github.com/aws-labs/amazon-sagemaker-examples>

## SageMaker SDK

- <https://github.com/aws/sagemaker-python-sdk>
- Doc は <https://readthedocs.org/projects/sagemaker/>

## SageMaker 公式ドキュメント

- [https://docs.aws.amazon.com/ja\\_jp/sagemaker/latest/dg/whatis.html](https://docs.aws.amazon.com/ja_jp/sagemaker/latest/dg/whatis.html)

# Q&A

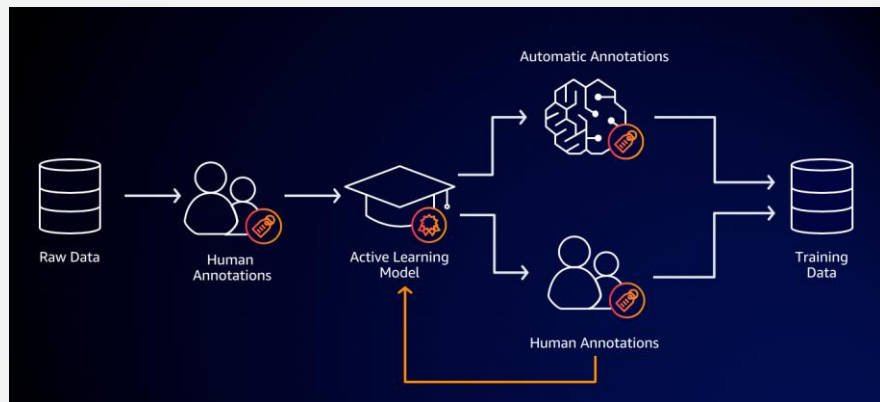
お答えできなかったご質問については  
AWS Japan Blog 「<https://aws.amazon.com/jp/blogs/news/>」にて  
資料公開と併せて、後日掲載します。

ご視聴ありがとうございました

# Appendix

# Ground Truth によるラベル付きデータの作成

- Amazon SageMaker Ground Truth は、機械学習の正解データ作成（アノテーションと呼ばれる）作業をサポートするマネージドサービス
- 画像や文章に正解ラベルを付与するのは、非常に手間のかかるプロセスだが、精度の高い正解データ作成は、機械学習活用には必須
- 画像認識、物体検出、テキスト分類等さまざまなタスクに対応
- アノテーターとして自社リソース、他社リソース、Amazon Mechanical Turk の 3 種類を利用可能
- 自動ラベルづけ機能等を活用して、アノテーション処理自体を自動化可能



## Task type [Info](#)

### Task selection

Select the task that a human worker will perform to label objects in your dataset.

#### Image classification

Get workers to categorize images into specific classes. [Info](#)



#### Bounding box

Get workers to draw bounding boxes around specified objects in your images. [Info](#)

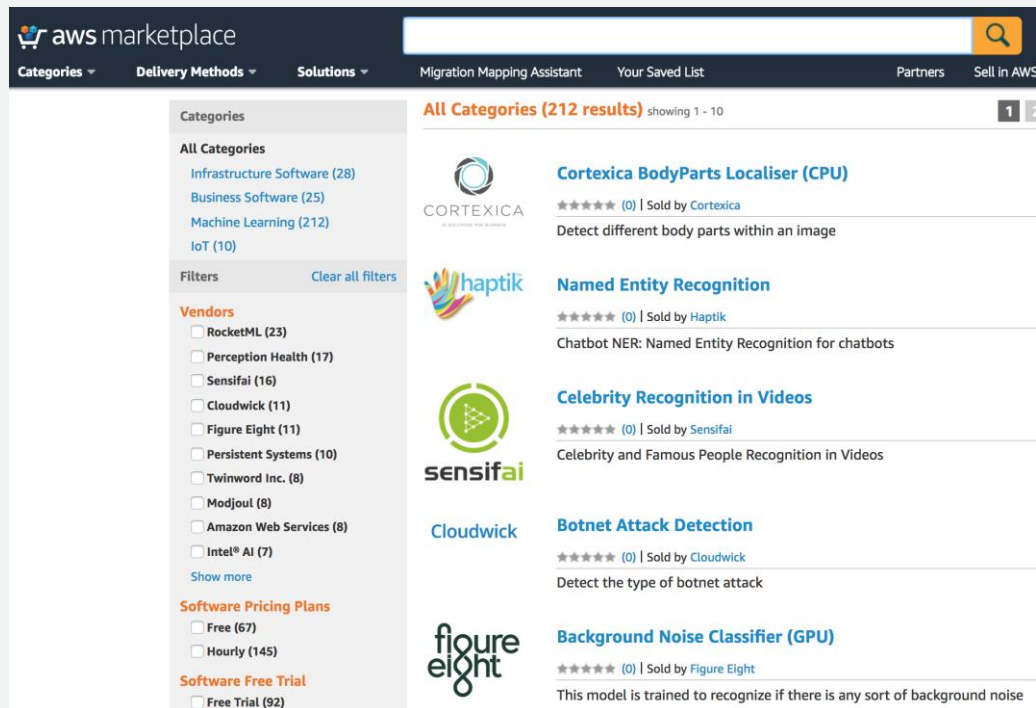


# ML Models in AWS Marketplace

さまざまな会社が提供する機械学習モデルを，マーケットプレイス経由でサブスクライブし，Amazon SageMaker の学習ジョブおよび，推論エンドポイントやバッチ推論ジョブで利用可能に

200 以上のアルゴリズムがすでに公開済み

自社アルゴリズムの販売も当然可能



The screenshot displays the AWS Marketplace interface. At the top, there's a search bar and navigation links for 'Categories', 'Delivery Methods', and 'Solutions'. Below this, a sidebar lists 'All Categories' (Infrastructure Software (28), Business Software (25), Machine Learning (212), IoT (10)), 'Filters' (Clear all filters), 'Vendors' (RocketML (23), Perception Health (17), Sensifai (16), Cloudwick (11), Figure Eight (11), Persistent Systems (10), Twinword Inc. (8), Modjoul (8), Amazon Web Services (8), Intel® AI (7)), 'Software Pricing Plans' (Free (67), Hourly (145)), and 'Software Free Trial' (Free Trial (92)). The main content area shows 'All Categories (212 results) showing 1 - 10'. The first three results are:

- Cortexica BodyParts Localiser (CPU)**: Sold by Cortexica. Detect different body parts within an image.
- Named Entity Recognition**: Sold by Haptik. Chatbot NER: Named Entity Recognition for chatbots.
- Celebrity Recognition in Videos**: Sold by Sensifai. Celebrity and Famous People Recognition in Videos.

Other visible results include **Botnet Attack Detection** (Sold by Cloudwick) and **Background Noise Classifier (GPU)** (Sold by Figure Eight).

# ノートブックから EMR への接続

- SageMaker のノートブックから既存の EMR に接続することで、大規模データに対する前処理を高速に実行可能
- ノートブックインスタンス起動時に、EMR クラスタがあるのと同じ VPC およびサブネットを指定する必要
- ノートブックとやり取りするために、EMR クラスタには Livy のインストールをして、適切なポートをあけておく

**Notebook instance settings**

Notebook instance name  
  
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

IAM role  
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

VPC - *optional*  
Notebook instances will have internet access independent of your VPC setting.

Subnet - *optional*

Security group(s) - *optional*

Encryption key - *optional*  
An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

# Airflow Operator サポート

- Airflow 1.10.1 より，SageMaker Operator をサポート
- 既存の Airflow 環境から，SageMaker を読んでパイプラインを構築することがより簡単に

```
train_op = SageMakerTrainingOperator(  
    task_id='training',  
    config=train_config,  
    wait_for_completion=True,  
    dag=dag)  
  
transform_op = SageMakerTransformOperator(  
    task_id='transform',  
    config=trans_config,  
    wait_for_completion=True,  
    dag=dag)
```

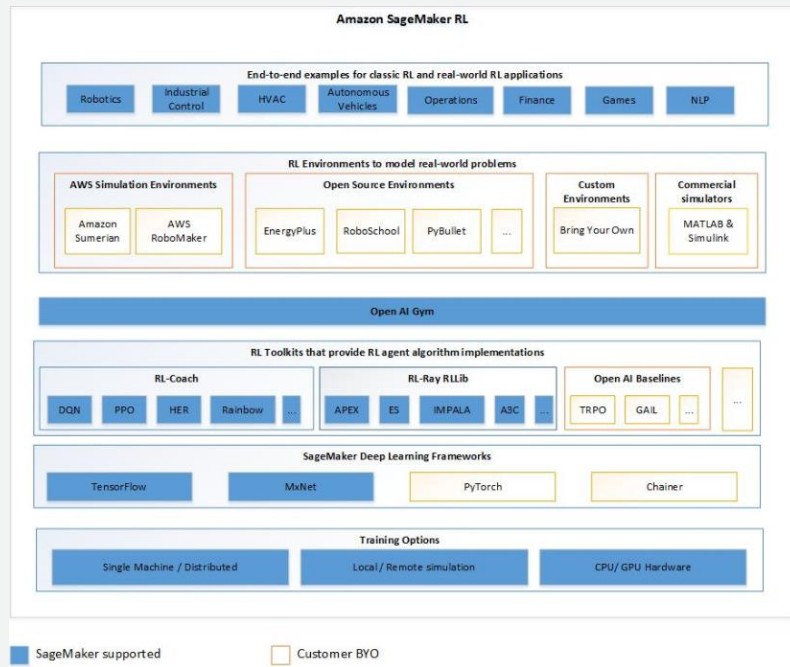
```
# train_config specifies SageMaker training configuration  
train_config = training_config(estimator=estimator,  
                               inputs=your_training_data_s3_uri)  
  
# trans_config specifies SageMaker batch transform configuration  
trans_config = transform_config_from_estimator(estimator=estimator,  
                                               instance_count=1,  
                                               instance_type='ml.m4.xlarge',  
                                               data=your_transform_data_s3_uri,  
                                               content_type='text/csv')
```

<https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/workflow/README.rst>



# 強化学習

- RLEstimator を用いることで、強化学習にも対応
- Open AI Gym / Intel Coach / Berkeley Ray RLLib などを含んだ形で、Tensorflow / MXNet のコンテナを利用することが可能
- また TensorFlow や StableBaselines のような強化学習ライブラリを活用して、自分自身の環境を作成することも可能
- 以下のようなツール群と連携
  - シミュレーター
    - AWS が提供: AWS RoboMaker, Amazon Sumerian
    - 他社提供: MATLAB and Simulink (ライセンスは別途必要)
  - 環境: OpenAI Gym, Gym インタフェースを使った環境 (Roboschool, EnergyPlus など)

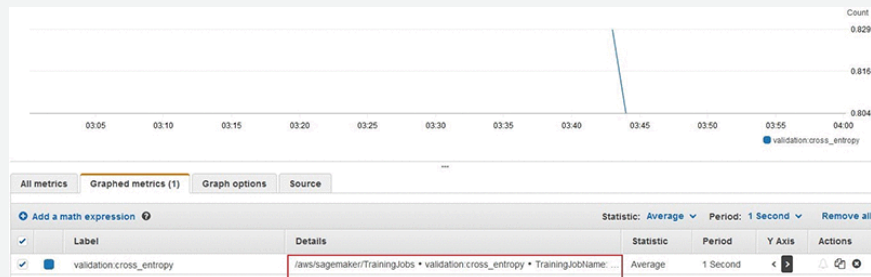


<https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/rl>  
<https://docs.aws.amazon.com/sagemaker/latest/dg/reinforcement-learning.html>

# 学習ジョブの評価の可視化

- CloudWatch Metrics 経由で、学習ジョブの指標を可視化することが可能
- **CreateTraininJob API** 実行時に、標準出力で出されるログに対する正規表現を指定することで、任意のメトリクスをログから取得して可視化することが可能
- ビルトインアルゴリズムは、初めから validation:cross\_entropy のようなメトリクスに対応している

Name	Regex
validation:cross_entropy	#quality_metric: host={S+}, epoch={S+}, validation cross_entropy <loss>={S+}
train:accuracy	#quality_metric: host={S+}, epoch={S+}, train accuracy <score>={S+}
train:progress	#progress_metric: host={S+}, completed {S+} %
test:cross_entropy	#quality_metric: host={S+}, test cross_entropy <loss>={S+}
test:accuracy	#quality_metric: host={S+}, test accuracy <score>={S+}
test:mean_squared_error	#quality_metric: host={S+}, test mean_squared_error <loss>={S+}
train:mean_squared_error	#quality_metric: host={S+}, epoch={S+}, train mean_squared_error <loss>={S+}
validation:accuracy	#quality_metric: host={S+}, epoch={S+}, validation accuracy <score>={S+}
train:cross_entropy	#quality_metric: host={S+}, epoch={S+}, train cross_entropy <loss>={S+}
validation:mean_squared_error	#quality_metric: host={S+}, epoch={S+}, validation mean_squared_error <loss>={S+}
train:throughput	#throughput_metric: host={S+}, train throughput={S+} records/second



<https://aws.amazon.com/jp/blogs/machine-learning/easily-monitor-and-visualize-metrics-while-training-models-on-amazon-sagemaker/>

# 学習ジョブの評価

- 学習ジョブの実行時のログは、すべて CloudWatch Logs に出力される
- 自作コンテナを使う場合は、標準出力に出したものがそのまま CloudWatch Logs に送られる
- モデル評価等は、すべてログに出力して、あとから集計

CloudWatch > Log Groups > Streams for /aws/sagemaker/TrainingJobs

Search Log Group Create Log Stream Delete Log Stream

Filter: Log Stream Name Prefix x

Log Streams

- xgboost-single-machine-regression-2018-01-10-05-24-53/algo-1-1515562124
- xgboost-single-machine-regression-2018-01-10-05-24-52/algo-1-1515562109
- xgboost-single-machine-regression-2018-01-10-05-21-21/algo-1-1515561899
- tensorboard-example-2018-01-16-13-38-16-318/algo-2-1516110122
- tensorboard-example-2018-01-16-13-38-16-318/algo-1-1516110122
- sagemaker-tensorflow-py2-cpu-2018-01-17-02-01-20-582/algo-1-1516154695
- sagemaker-tensorflow-py2-cpu-2018-01-17-01-56-00-319/algo-1-1516154380
- sagemaker-tensorflow-py2-cpu-2018-01-17-01-24-43-028/algo-1-1516152503
- sagemaker-mxnet-py2-gpu-2018-01-10-02-00-42-508/algo-1-1515549902
- sagemaker-mxnet-py2-gpu-2017-12-15-01-06-31-749/algo-1-1513300180

Filter events

Time (UTC +00:00)	Message
2018-01-10	No older events found at the moment. <a href="#">Retr...</a>
▶ 05:30:11	Arguments: train
▶ 05:30:12	[2018-01-10:05:30:11:INFO] Running standalone xgboost training.
▶ 05:30:12	[2018-01-10:05:30:11:INFO] File size need to be processed in the nod
▶ 05:30:12	[05:30:11] S3DistributionType set as FullyReplicated
▶ 05:30:12	[05:30:11] 2923x9 matrix with 23384 entries loaded from /opt/ml/input
▶ 05:30:12	[05:30:11] S3DistributionType set as FullyReplicated
▶ 05:30:12	[05:30:11] 626x9 matrix with 5008 entries loaded from /opt/ml/input/d
▶ 05:30:12	[05:30:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30
▶ 05:30:12	[0]#011train-rmse:8.12873#011validation-rmse:7.89999
▶ 05:30:12	[05:30:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32
▶ 05:30:12	[1]#011train-rmse:6.6447#011validation-rmse:6.42765
▶ 05:30:12	[05:30:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 44
▶ 05:30:12	[2]#011train-rmse:5.48553#011validation-rmse:5.27792
▶ 05:30:12	[05:30:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30
▶ 05:30:12	[3]#011train-rmse:4.59102#011validation-rmse:4.3968
▶ 05:30:12	[05:30:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 48
▶ 05:30:12	[4]#011train-rmse:3.89811#011validation-rmse:3.71958
▶ 05:30:12	[05:30:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 48
▶ 05:30:12	[5]#011train-rmse:3.35896#011validation-rmse:3.19781

# PIPE モードによるデータの高速な読み込み

- 学習時のデータ読み込み方法は、以下の 2 種類がある
  - FILE: 学習用のデータをすべて学習インスタンスにコピー
  - PIPE: 学習用のデータを、必要なタイミングで必要なぶんだけ S3 API 経由でストリームとして取得
- 以下の場合には、PIPE モードを使うことでパフォーマンスアップが期待できる
  - Tensorflow フレームワークで TFRecord フォーマットデータを扱うとき
  - MXNet フレームワークで RecordIO フォーマットデータを扱うとき
- Chainer および PyTorch には、上記のような入出力専用フォーマットが存在しないため、PIPE モードによる速度向上の恩恵を受けることはできない

# SageMaker Neo によるモデルのコンパイル

- Neo により，SageMaker で学習したモデルを，推論環境に最適化された形でコンパイルすることが可能に
- 現状の対応リージョンはバージニア北部，オレゴン，アイルランドのみ
- コンパイルジョブ自体の利用料金は無料

対応フレームワーク	対応プラットフォーム
<ul style="list-style-type: none"><li>• TensorFlow</li><li>• Apache MXNet</li><li>• PyTorch</li><li>• ONNX</li><li>• XGBoost</li></ul>	<ul style="list-style-type: none"><li>• EC2 インスタンス (c4/5, m4/5, p2/3)</li><li>• Jetson TX1/2</li><li>• DeepLens</li><li>• Raspberry Pi 3 Model</li></ul>

<https://aws.amazon.com/jp/blogs/news/amazon-sagemaker-neo-train-your-machine-learning-models-once-run-them-anywhere/>  
<https://docs.aws.amazon.com/sagemaker/latest/dg/neo.html>

# セキュリティ: 暗号化とコンプライアンス

- 学習と推論のジョブにおいて、オプションパラメータとして KMS key ID を指定することで、SSE-KMS を利用可能
  - `CreateTrainingJob` /
  - `CreateEndpointConfig`
- 以下のものをすべて暗号化可能
  - 学習時の入出力データ
  - 学習用インスタンス, およびエンドポイントインスタンスのストレージ
  - バッチ推論時の入出力データ
- Cloudtrail に対応済み
- PCI DSS および HIPAA に対応済み

<https://aws.amazon.com/about-aws/whats-new/2018/01/aws-kms-based-encryption-is-now-available-in-amazon-sagemaker-training-and-hosting/>

<https://aws.amazon.com/about-aws/whats-new/2018/01/aws-cloudtrail-integration-is-now-available-in-amazon-sagemaker/>

<https://aws.amazon.com/about-aws/whats-new/2018/01/amazon-sagemaker-achieves-pci-dss-compliance/>

<https://aws.amazon.com/about-aws/whats-new/2018/04/access-amazon-vpc-resources-for-training-and-hosting-with-amazon-sageMaker/>

<https://aws.amazon.com/about-aws/whats-new/2018/05/Amazon-SageMaker-Achieves-HIPAA-Eligibility/>

<https://aws.amazon.com/jp/about-aws/whats-new/2018/06/amazon-sagemaker-inference-calls-are-supported-on-aws-privatelink/>

# セキュリティ: 閉域網での通信

- SageMaker と S3 のデータ通信は、すべて S3 VPC エンドポイント経由で行うことが可能
  - 学習ジョブの入出力における S3 アクセス
  - 学習済モデルをデプロイする際の S3 アクセス
- SageMaker の API は、すべて PrivateLink 経由で行うことが可能
  - SageMaker Notebook Endpoint
  - SageMaker Service API
  - SageMaker Runtime API

<https://aws.amazon.com/blogs/machine-learning/direct-access-to-amazon-sagemaker-notebooks-from-amazon-vpc-by-using-an-aws-privatelink-endpoint/>  
<https://aws.amazon.com/about-aws/whats-new/2018/04/access-amazon-vpc-resources-for-training-and-hosting-with-amazon-sageMaker/>  
<https://aws.amazon.com/jp/about-aws/whats-new/2018/06/amazon-sagemaker-inference-calls-are-supported-on-aws-privatelink/>  
<https://aws.amazon.com/about-aws/whats-new/2018/08/amazon-sagemaker-apis-supported-on-aws-privatelink/>