# Sagemakerの特徴：

1. インターフェース SageMakerでは、Jupyter Notebookを使用して、「インスタンスの作成」、「モデル構築」、「トレーニング」、「デプロイ」までのフローを実行することができる。対話型のブラウザで実行することで、実行結果を分かりやすい形で監視することができる。
2. 主要な機械学習フレームワーク Tesnsorflowやsklearnなどの機械学習モデルをDockerコンテナ上で実行するように構成されている。
3. 一般的な機械学習アルゴリズムが事前にインストールされている。 (他の機械学習サービスと比較して、高いパフォーマンスでアルゴリズムを実行できる可能性が高い)
4. ワンクリックトレーニング SageMakerのコンソールからワンクリックで高度なトレーニング他チューニングを実行できるようになっている。
5. フルマネージメントスケーリング機能 インフラ部分については自動的に管理されてスケーリング(規模の増減、面積の拡大縮小）されるので、ペタバイト規模のモデルトレーニングが簡単にスケーリングできる。

# 基本的な準備:

1. S3バケットの作成 名前の中で「sagemaker」の文字列を含める必要がある。
2. Amazon SageMakerのトップページから、ノートブックを作成する。 ノートブック作成の際にインスタンスを指定する。
3. IAMロールの作成

# SageMakerのセットアップ

まずはコンソールログインする。

ノートブックインスタンスを選択


# AWSサービスの概要

## データ分析とデータ習得

AWSでデータ管理に使うことのできるサービス：

- Amazon SageMaker Ground Truth
    - データのラベル付けを簡単に行うことができるデータラベリングサービス
    - Amazon SageMaker Ground Truth では手動ラベリングと自動ラベリングを選択することができる。
        - 手動ラベリング：
          3 種類のチームを選択することができる。

          1. 社員などのプライベートのチーム
          2. Amazon Machanical Turk を使用したパブリックチーム
          3. ベンダーなどのサードパーティーのチーム

          チームのメンバーは Amazon SageMaker Ground Truth が用意した UI を通してラベリング作業を行う。
    - 自動ラベリング：
      手動ラベリングに追加して有効化でき、ビルトインアルゴリズムを利用した機械学習のモデル学習を行う。
    - ラベリングされたデータは安価で高耐久なオブジェクトストレージである Amazon S3 のバケットに格納されます。
- Amazon Athena
    - インタラクティブなクエリサービス. Amazon S3 内のデータを標準 SQL を使用してクエリを取得できる。
    - サーバーレスでインフラストラクチャの管理かつ実行したクエリに対してのみ料金が発生。
- Amazon Redshift

    - データウェアハウスサービスです。データウェアハウス（DWH）というのは、さまざまなデータ源からデータを収集・統合・蓄積し、分析のため保管しておくシステムのこと。伝統的なRDBMSとは違って、継続的な書き込みや更新には向いておらず、一括でデータを書き込み分析のため大容量データを読み出すという処理に最適化されている。その結果として、たとえばRDB設計における正規化はデータウェアハウスでは重視されず、読み出しの高速化のためにあえて正規化しないでデータを格納することもある。Amazon Redshiftでは、並列コンピューティングをサポートしており、大量のデータを短時間で読み出し・分析することが可能です。インターフェイスとしては、BIツールやPostgreSQLクライアントから操作することが可能。

---

## 前処理

- Amazon SageMaker Processing Amazon SageMaker SDK を通して、コンテナイメージ、前処理のコード、入力と出力の Amazon S3 バケットを指定することで、自動でインフラの構築と処理を実行する。Amazon EC2 インスタンスの起動や停止、ライブラリのインストールなどを考える必要がない。
- AWS Batch Amazon SageMakerで一気通貫に行う必要がない場合に利用できる。
- AWS Glue より複雑なデータの変換などをしたい場合、Apache Sparkで大規模に処理したい場合などに利用すると良い。

---

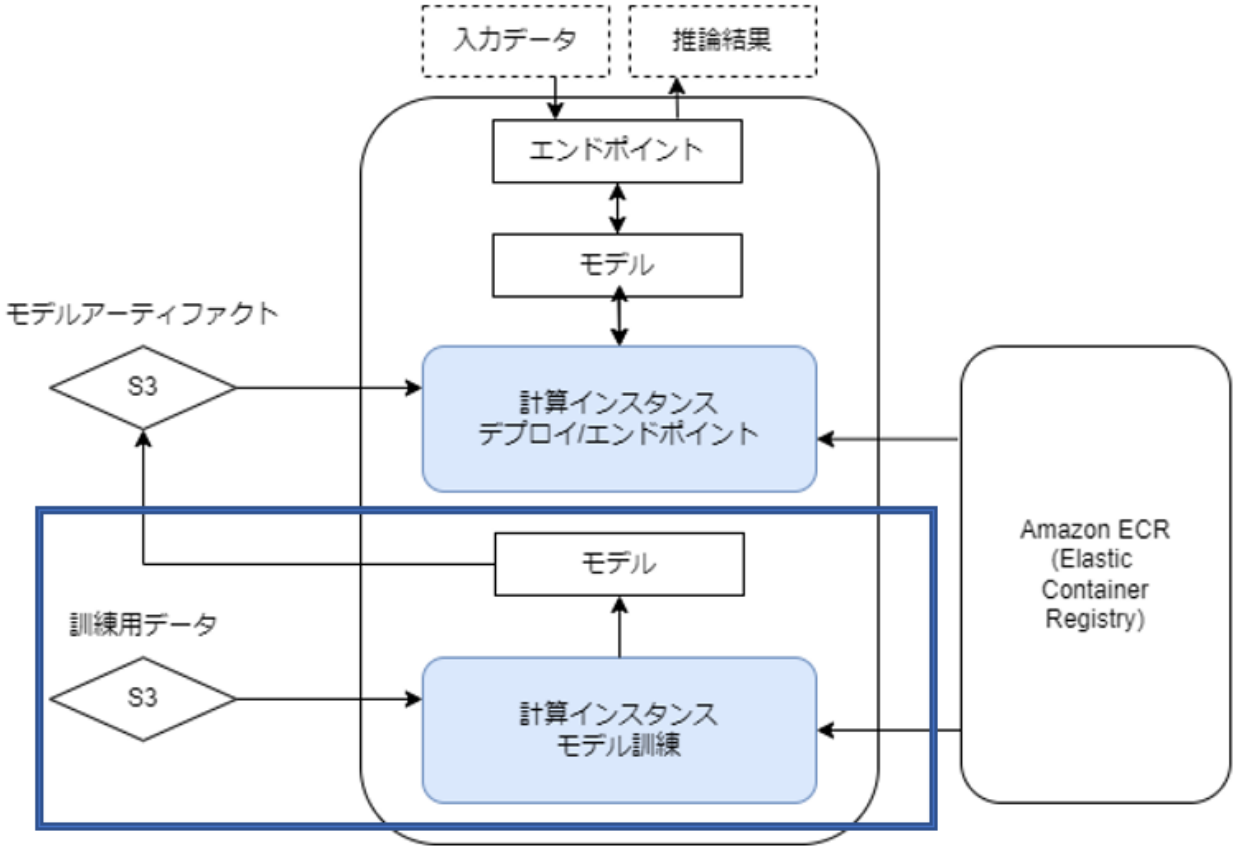
## モデルの構築・学習

- SageMaker
    - Sagemakerでは組み込みのアルゴリズムに加えて以下のようなフレームワークを利用できるコンテナを提供している。
        - TensorFlow
        - Pytorch
        - Apache MXNet
        - XGBoost
        - scikit-learn

# 機械学習モデルの作成

---

## Sagemakerによる機械学習モデルの作成




## 学習時と推論時の環境の一致について

Sagemakerでは、基本的に学習用と推論用のコンテナを分けて運用することが前提となっており、それぞれについてコンテナイメージを用意する必要がある。また、SageMakerでは組み込みのコンテナを用いてモデルを学習させる、もしくは独自のモデルを含むコンテナを用意する必要があり、その際には学習用と推論用のコンテナイメージに含まれる環境をできる限り統一する必要がある。

## 推論器の稼働


SageMakerを利用することで、推論用の環境を独自で用意する必要はなくなる。基本的はモデルのデプロイ時に推論用のエンドポイントを作成して、そのエンドポイントを通じて推論用のサーバーに入力値を送信する。

- 大量のモデルを管理したい場合 SageMakerでは複数のモデルについていつでも本番環境で利用できるようにマルチモデルエンドポイントを利用することができる。また必要に応じて、1 つのエンドポイントに複数のトレーニング済みモデルをデプロイし、単一のサービングコンテナを使用して稼働させることでコストの削減を行うことができる。エンドポイントを呼び出す際には、ターゲットのモデル名を指定することで、特定のモデルに簡単にアクセスできる。

- 新しくモデルを追加する場合
  S3 ではモデルのどの階層でもエンドポイントを定義でき、マルチモデルエンドポイントにモデルを追加するには、新しいトレーニング済モデルのアーティファクトを S3 に追加して呼び出せば良い。

- すでに使用中のモデルを更新する場合
  S3 に新しい名前のモデルを追加して、新しいモデル名でエンドポイントの呼び出しを開始する。マルチモデルエンドポイントでデプロイされたモデルの使用を中止するには、モデルの呼び出しを停止し、S3 からモデルを削除すればよい。

- 具体的なユースケース

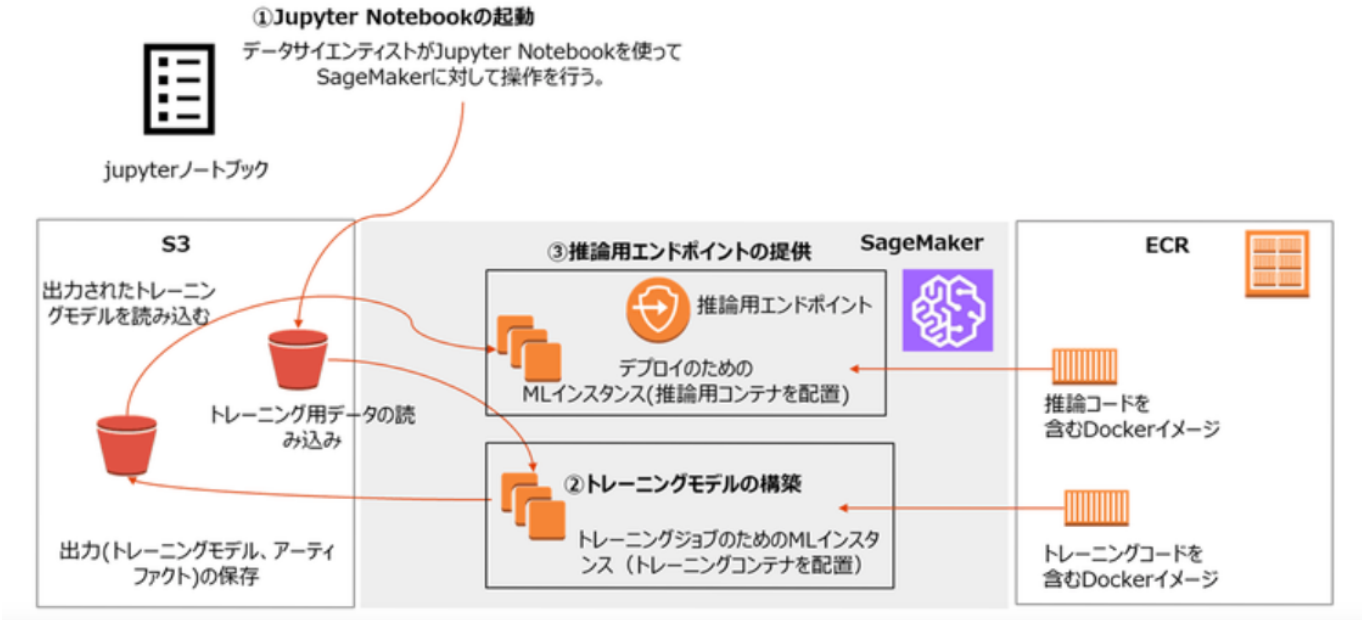
    - 法律関係のアプリケーション
      広範な規定上の管轄を完全にカバーする必要がありますが、それにはめったに使用されないモデルが多数含まれることになります。1 つのマルチモデルエンドポイントでは、使用頻度の低いモデルに対応し、コストの最適化と大量のモデルの管理を効率よく行うことができます。

また、SageMakerでは訓練用データ、出力先ファイルの他にモデルの学習に必要な環境変数（インスタンスの数など）を引き渡すことで、以下の訓練ジョブのフローを容易に行うことができる。

- 訓練ジョブの実行フロー
    1. S3からの入力データの読み込み
    2. データを利用したモデルの訓練
    3. モデルをS3に書き戻す

## モデルロードパターン

SageMakerのシステムは「AIエンジニアのための機械学習システムデザインパターン」の3章で紹介されているモデルロードパターンに近い。




基本的にSageMakerはJupyter Notebook、もしくはJupyter Labを使用して操作を行うが、学習時、推論時にそれぞれ実行用のインスタンスを用意して、そこにモデルのDocker イメージをダウンロードする形式を取る。Dockerのイメージは、ダウンロード先の環境で動くイメージを予めECRに登録しておき、インスタンス起動時に使用できるようにしておく。

# 推論システムの作成

# 推論システムの運用

| オプション | ユースケース |
| --- | --- |
| リアルタイム推論 | トラフィックが多く、低レイテンシーが求められ、そこまでペイロードが大きくない場合。 |
| 非同期推論 | ペイロードが大きく、処理時間も長い場合でコールドスタートが許容できる場合。 |
| バッチ変換ジョブ | バッチで推論を実行。 |
| Serverless Inference (preview) | トラフィックが断続的で予測できないが低レイテンシーが求められる場合。 |

- 運用について
  基本的に必要に応じてライブラリや学習済みモデルを外部からインストールするが、外部への依存度を考えると、これはセキュリティの観点や可用性あまり推奨されない。

  基本的には、これらのリソースをDocker イメージやS3バケットにコピーする形で利用する方が良い。

## 参考文献

https://www.acrovision.jp/service/aws/?p=1237

### 独自のコンテナを利用する方法

https://corp.logly.co.jp/blog/417

https://www.inoue-kobo.com/aws/sagemaker-with-mycontainer/index.html

## Redshiftについての解説

https://techblog.nhn-techorus.com/archives/8232

## SageMakerについての公式説明：

マルチエンドポイントについて https://aws.amazon.com/jp/blogs/news/save-on-inference-costs-by-using-amazon-sagemaker-multi-model-endpoints/

エンドポイントを使ったリリースについて https://aws.amazon.com/jp/blogs/news/load-test-and-optimize-an-amazon-sagemaker-endpoint-using-automatic-scaling/

## SageMakerの仕組み

https://www.accenture.com/jp-ja/blogs/cloud-diaries/amazon-sage-maker