INNOVATE

AI/ML EDITION

24 February 2022

Amazon SageMaker でつくる MLOps
– 効率的な機械学習の開発・運用環境 –

宇都宮 聖子
アマゾン ウェブ サービス ジャパン合同会社
シニア機械学習ソリューションアーキテクト

aws

宇都宮聖子, Ph.D.

アマゾン ウェブ サービス ジャパン合同会社
シニア機械学習ソリューションアーキテクト

2018年よりAWSにジョイン
AWS 機械学習・量子コンピュータを担当

前職は、自動車 OEM にて機械学習を用いた自動運転開発
前々職は、国立情報学研究所にて量子コンピュータの研究者

好きな AWS サービス: Amazon SageMaker, Amazon Braket

Twitter: shokout

はじめに

- 対象：機械学習のモデル開発者、これから機械学習運用を目指す方
- 概要： Amazon SageMaker を中心に AWS 上で 効率的な機械学習の開発運用環境 (MLOps) を実現するための方法を設計原則やベストプラクティスを交えてご紹介

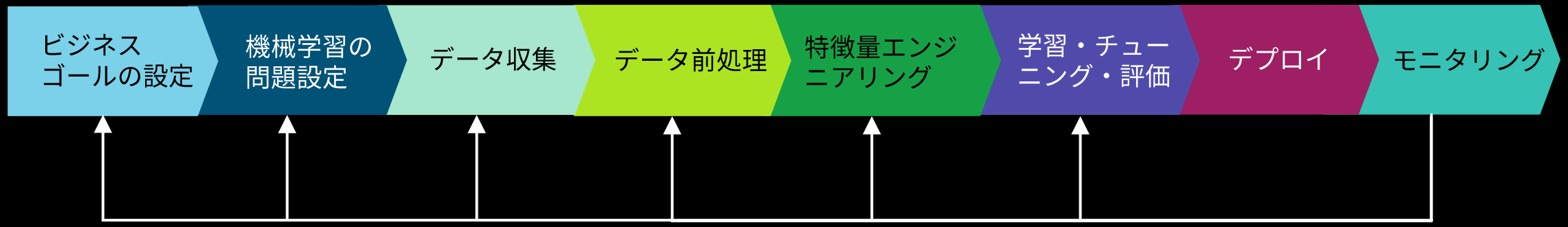
本セッションでお話すること

- MLOps を考える上で必要な要素
- MLOps 構築のためのツール・Amazon SageMaker の MLOps 機能
- Amazon SageMaker を用いた MLOps ベストプラクティス

機械学習の開発・運用のパイプライン

https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/machine-learning-lens.html

高精度なMLモデルを継続的に提供し続ける

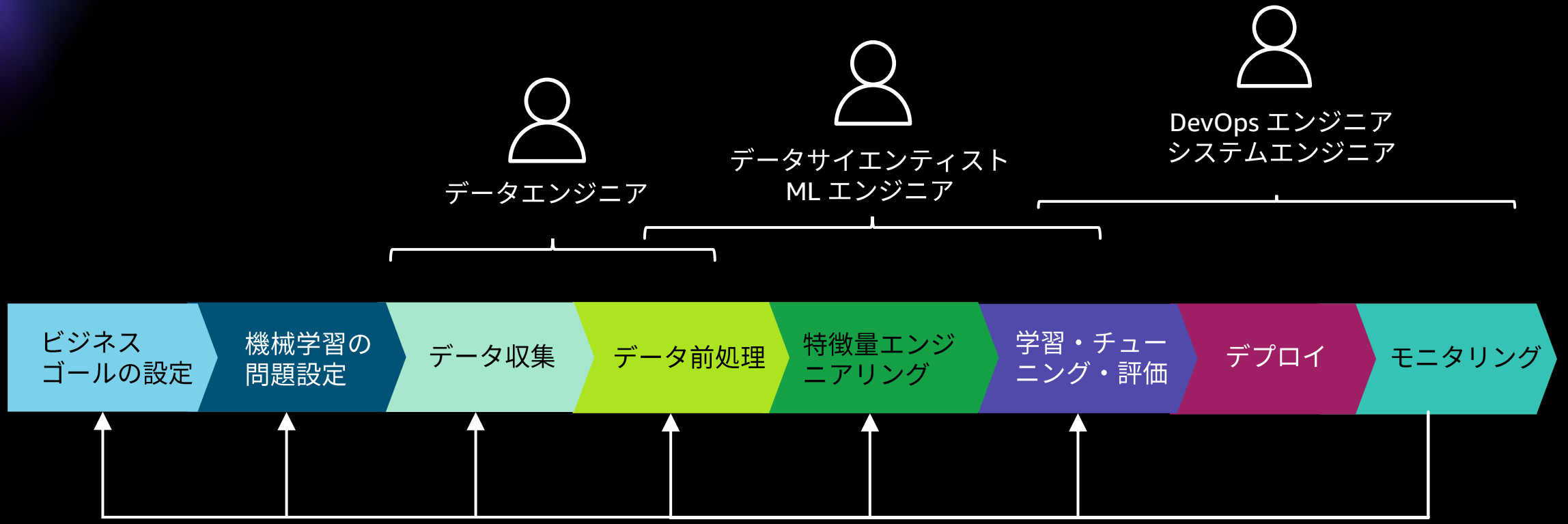


高精度な機械学習モデルを定常的に運用していくためのフィードバック

https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/machine-learning-lens.html

aws

ML開発運用のパイプラインとペルソナ

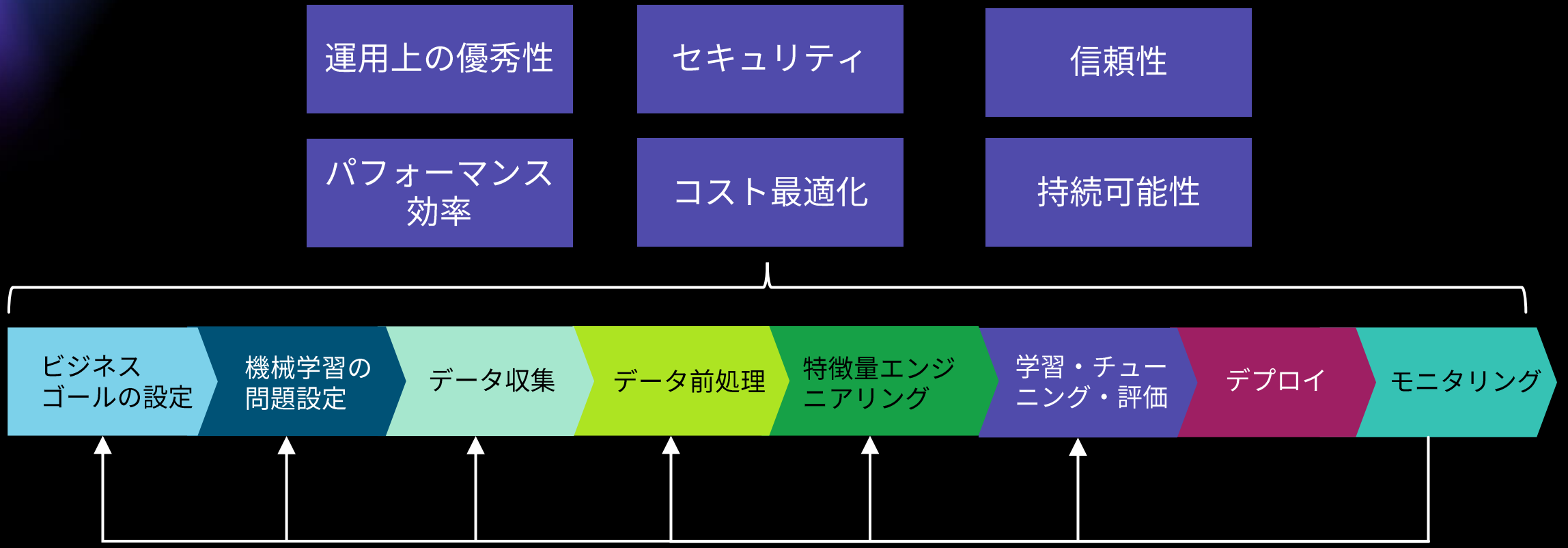


異なるペルソナ・チーム間で効率的なデータ・
モデルの受け渡しができる環境構築が重要

あなたの機械学習環境は？

- 運用基盤のコスト最適化
  - 学習・推論・運用基盤のコスト、最適化できていますか？
  - ML基盤の保守・運用が煩雑でメンテナンスに負荷がかかっていませんか？
- パイプライン
  - 機械学習の開発から運用まで、コード管理 ～ CI/CD パイプライン どうしていますか？
  - 異なるチーム間で成果物（データ・モデル等）をどう受け渡していますか？
- データ・リソース管理
  - 増え続けるモデル・データ・特徴量、どこでどのように管理していますか？
  - トレーサビリティを担保できていますか？

Machine Learning Well Architected Framework - MLLens

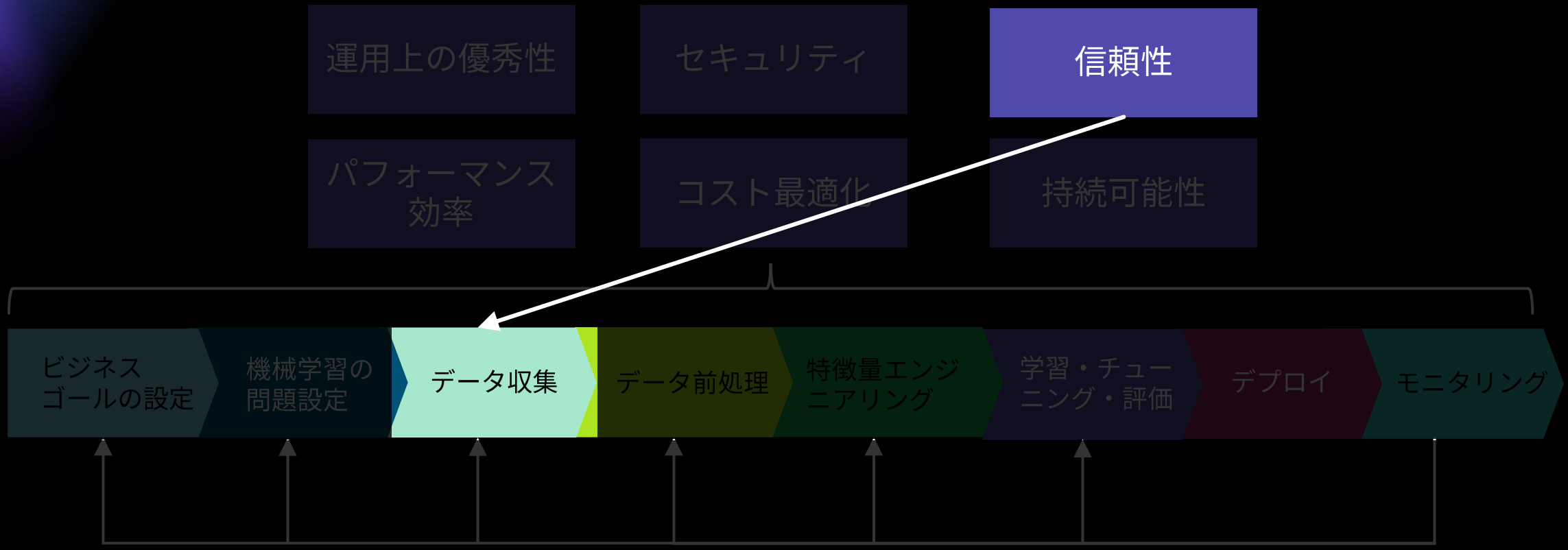


あなたのML開発運用環境を6つの柱について診断！

https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/well-architected-machine-learning-lifecycle.html

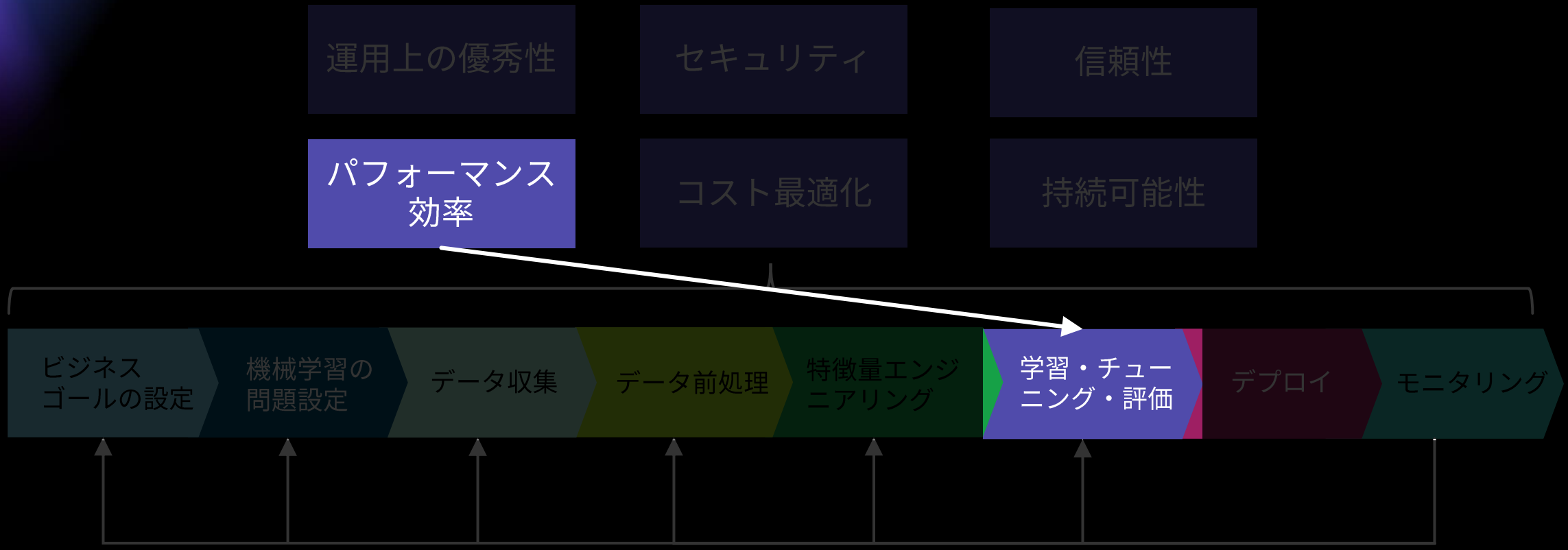
aws

信頼性 x データ収集



障害が発生した際に、モデルの正確なバージョンを再現できるように、
学習に使ったデータとモデル等が紐付いて管理されていますか？

パフォーマンス効率 x 学習・チューニング



ワークロードに応じてパフォーマンスを最適化するGPUやCPUスペックを選択できていますか？マシンリソースを最適化できていますか？

効率的な機械学習環境を構築するために考えるべきこと

機械学習パイプラインの効率的な組み方

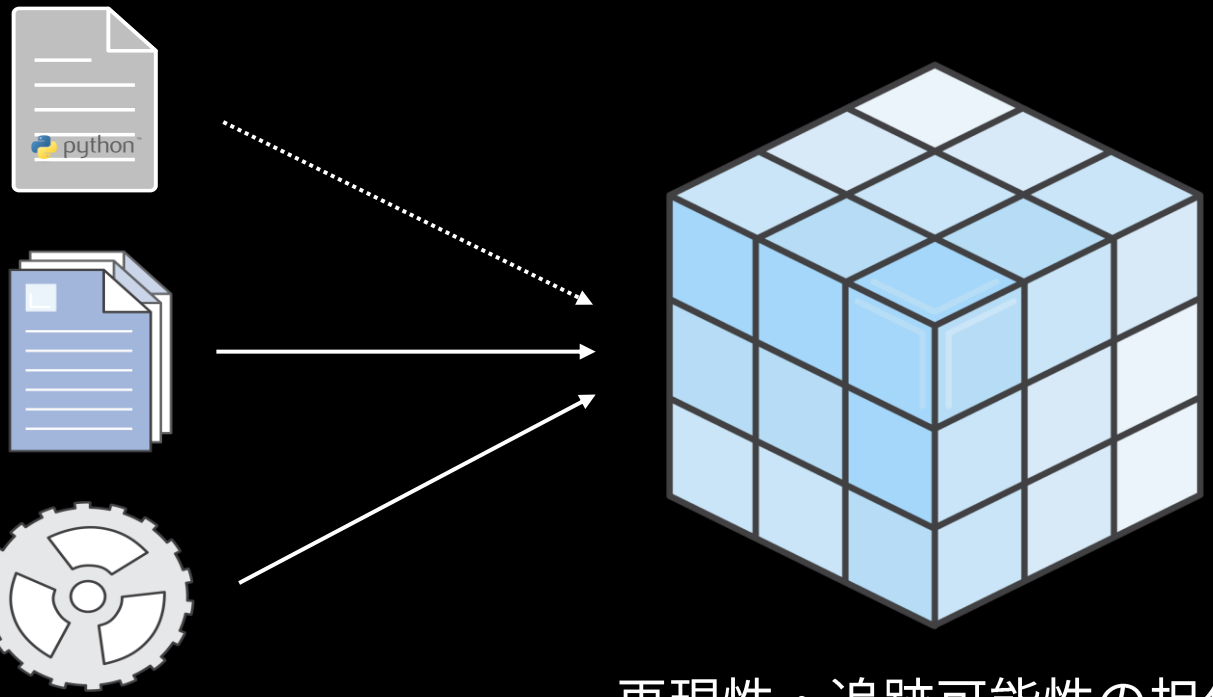
- コンテナを活用
  - コンテナによる仮想環境で環境を共通化、繰り返しの設定コストを減らし、ムラを無くす
  - チーム内外の環境統一化により開発効率を上げる
- CI/CD パイプライン自動化
  - コードレポジトリから開発デプロイまでを継続的に
  - 自動化できるところを探し、繰り返し作業のコストを削減
- データ・モデル・特徴量等を管理
  - 学習に使ったデータ・特徴量・モデル等を管理、再利用可能にする
  - 学習時のアーティファクトを紐付け管理、トレーサビリティを担保、再現可能にする

コンテナによる機械学習環境の統一化

機械学習スクリプト
train.py, inference.py, preprocess.py

Deep Learning Framework
TensorFlow / mxnet PYTORCH Chainer
GLUON / Keras HOROVOD

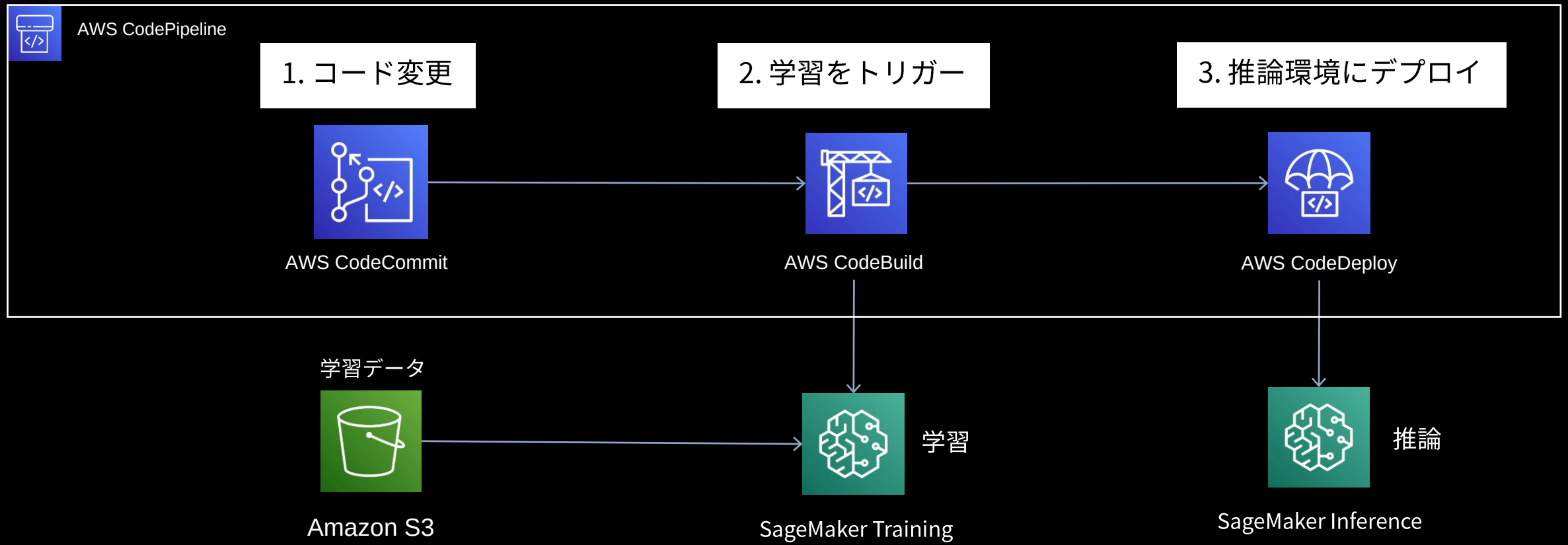
ランタイム/エンジン
Python, CUDA, cuDNN



再現性・追跡可能性の担保

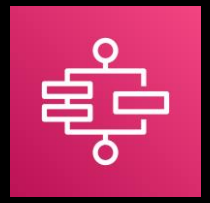
機械学習の CI/CD パイプライン

Git サービスやビルドサービスを機械学習ワークフローと連携させることで、
コード変更に対して、モデルの評価やデプロイへのテストなどが一括管理、自動化できる



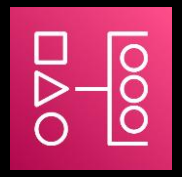
Pipeline as Code

パイプラインをコードで記述することにより属人化したデプロイプロセスを効率化



AWS Step Functions

- サーバーレスオーケストレーションサービス
- ETLやMLプロセス、マイクロサービスのアプリ実行などのワークフローをステートマシンで記述・制御
- JSON ベースの構造化言語 Amazon State Language でステートマシンを定義
- データサイエンティスト向けに Python で記述できる AWS Step Functions Data Science SDK



Amazon Managed Workflows for Apache Airflow (MWAA)

- Apache Airflow によるワークフローのマネージドサービス
- Python で記述された Directed Acyclic Graphs (DAG) のデプロイをすぐに開始
- OSS の Apache Airflow を大規模に実行する手間を省き、ビルトインセキュリティで Airflow を実行
- Athena、Batch、Cloudwatch、DynamoDB、DataSync、EMR、ECS / Fargate、EKS、Firehose、Glue、Lambda、Redshift、SQS、SNS、Sagemaker、S3 などのワークフローに必要な AWS またはオンプレミスのリソースと接続可能

Amazon SageMaker Pipelines

- Amazon SageMaker の中で機械学習ワークロードの CI/CD を実現する
- 機械学習ワークフローのデータロードや学習処理などの一連の処理ステップを任意のタイミングや所定の時間に実行できる
- 各ステップの処理結果は SageMaker Experiments で記録され、モデルの出来映えや学習パラメータなどを視覚化できる

AWS の ML CI/CD ~ パイプライン自動化ツール

開発者

デプロイパイプライン

学習・テスト　デプロイ

プラン　モニタリング

フィードバックループ

顧客

コード管理・CI/CD

AWS CodeCommit

AWS CodeBuild

AWS CodeDeploy

git

GitLab

実験管理

Amazon SageMaker Experiments

mlflow

TensorBoard

BENTOML

パイプライン自動化

AWS Step Functions

Amazon Managed Workflows for Apache Airflow

Amazon SageMaker Pipelines

Amazon EventBridge

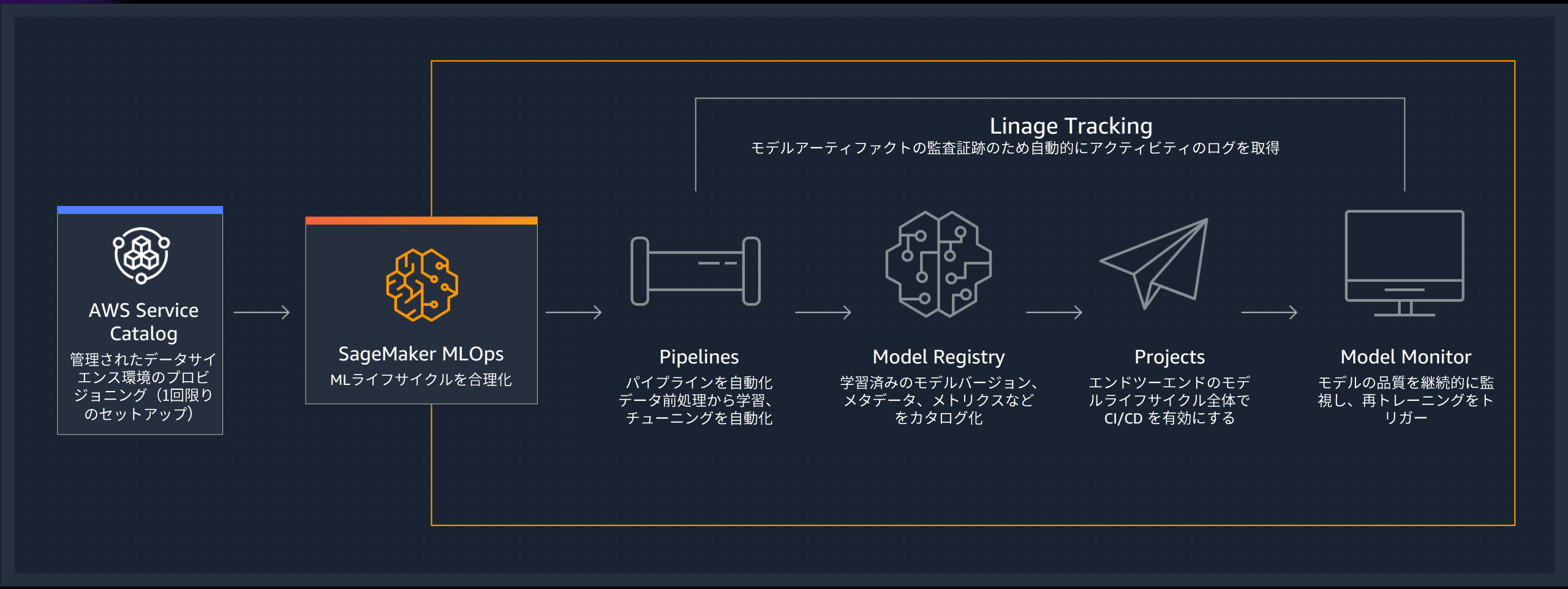
AWS CodePipeline

Apache Airflow

Amazon SageMaker の MLOps 機能

Amazon SageMaker の MLOps 機能

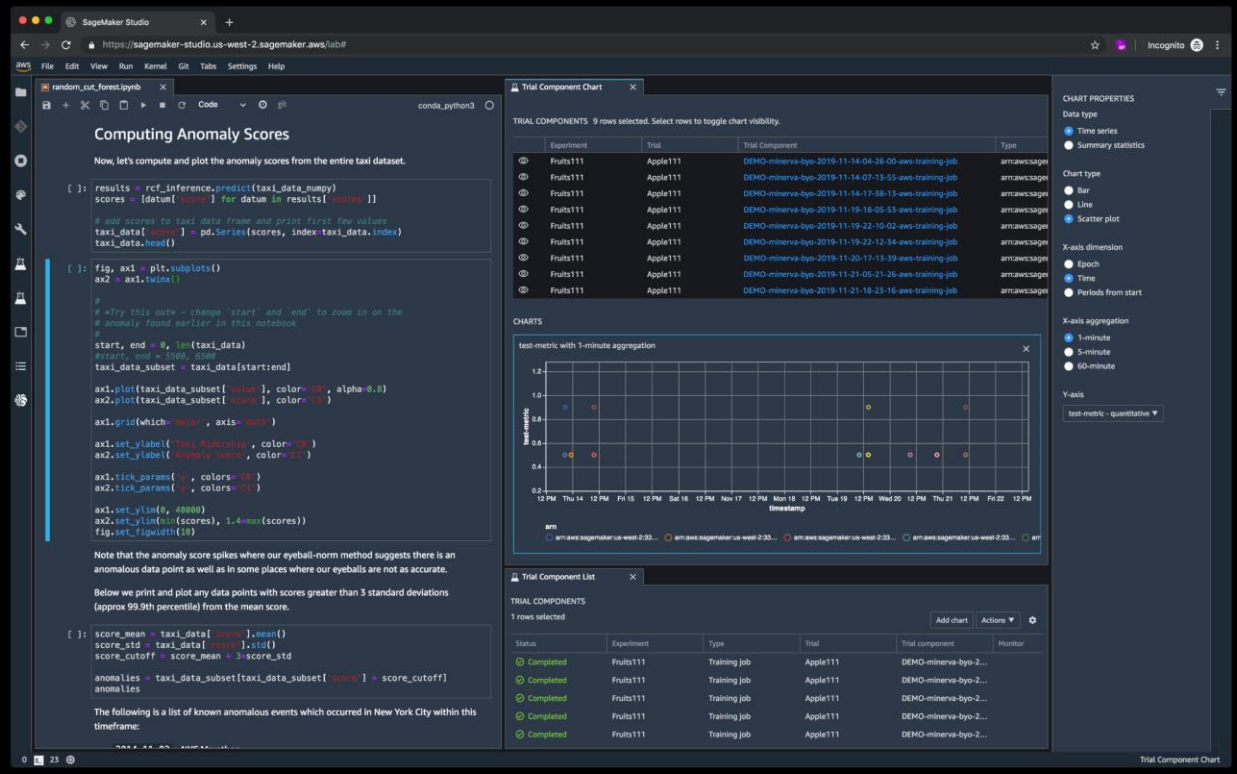
SageMaker 内で ML ライフサイクルを合理化するための機能



Amazon SageMaker の基本機能

機械学習の開発運用を効率化するコンテナオーケストレーションサービス

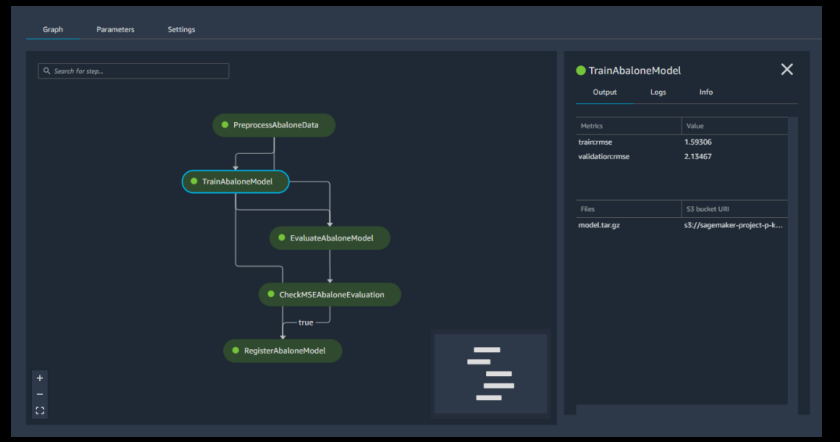
- 機械学習に必要なライブラリ等が入った SageMaker のデフォルトコンテナ
- マネージドにリソース（GPU，CPU）を立ち上げ、学習が終了するとリソース削除
- 前処理・後処理には SageMaker Processing container で環境構築
- 本番用の推論環境をクイックにデプロイ



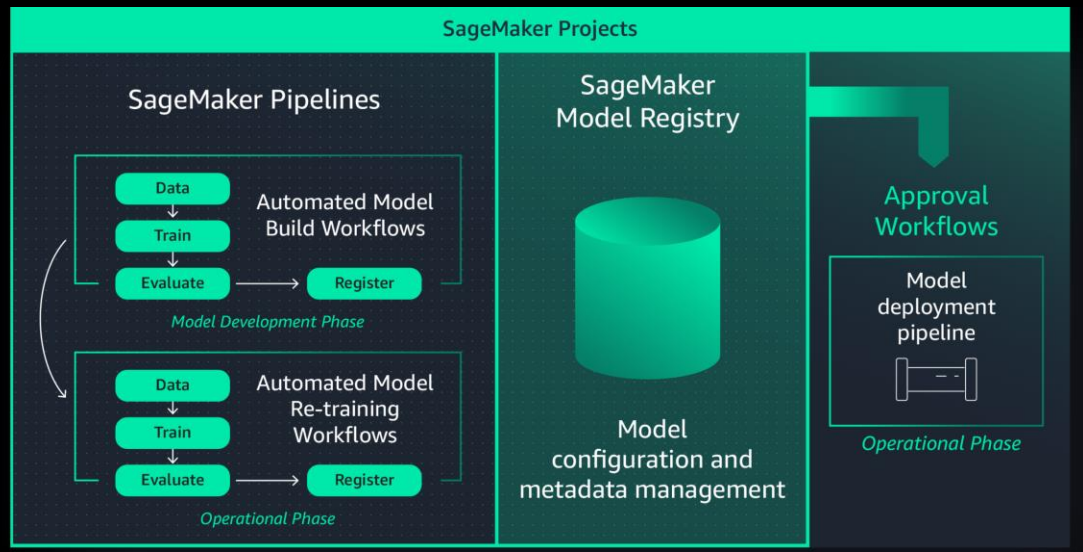
統合開発環境 SageMaker Studio Notebook

SageMaker Pipelines によるパイプライン管理

- 機械学習の CI/CD パイプライン を実現
- 機械学習ワークフローの作成
  - データロードや学習処理、評価など一連のワークフローを JSONパイプラインとして定義
  - DAG を使用してエンコード、再利用可能
- SageMaker の各種機能と連携
  - モデルレジストリから最適なモデルを選択
  - ML モデルの自動追跡
  - デプロイ管理のための認証機能



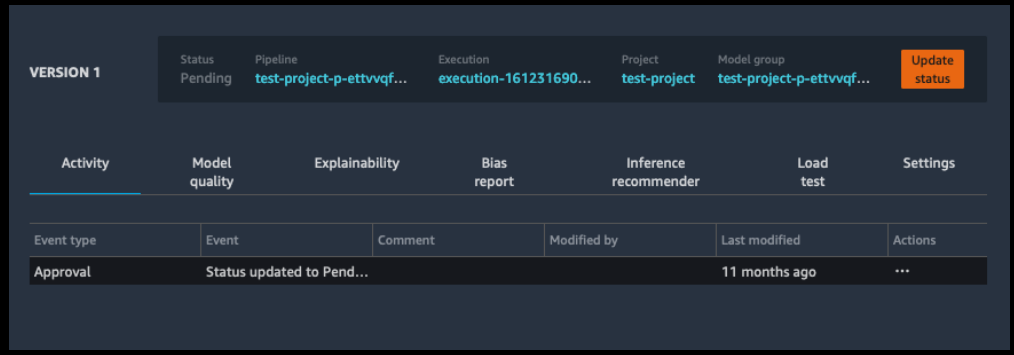
Studio 内での Pipelines 管理画面



Amazon SageMaker の学習モデル管理

Amazon SageMaker Model Registry

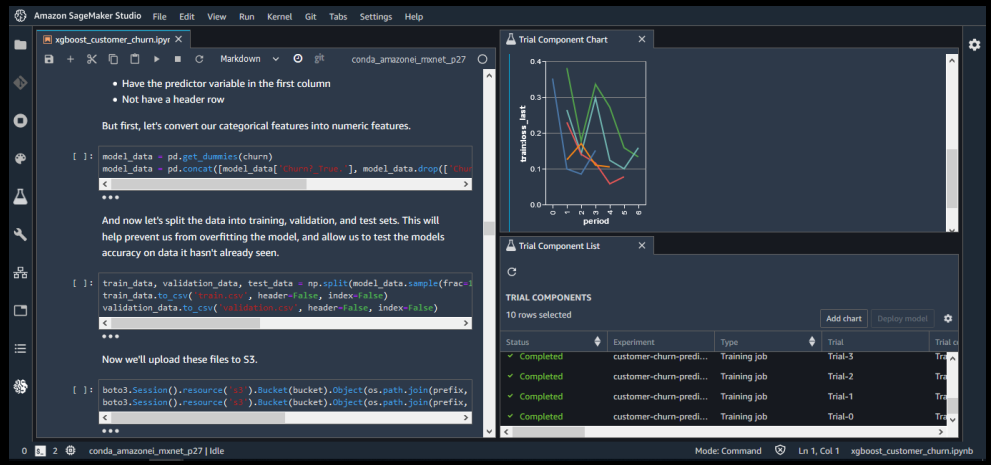
- モデルパッケージグループを登録
- 学習によるモデルのバージョニングが可能に
- モデルパッケージバージョンごとに承認・拒否・デプロイを実行
- クロスアカウントのモデルバージョン管理も可能



モデルのバージョンごとにメトリクス管理

Amazon SageMaker Experiments

- 学習を改善するための試行錯誤 (実験) の結果を管理する機能
- 複数の異なる学習を1つの実験と紐付け、メトリクスなどを一元管理
- 実験の結果を、pandas や matplotlib を使って分析可視化



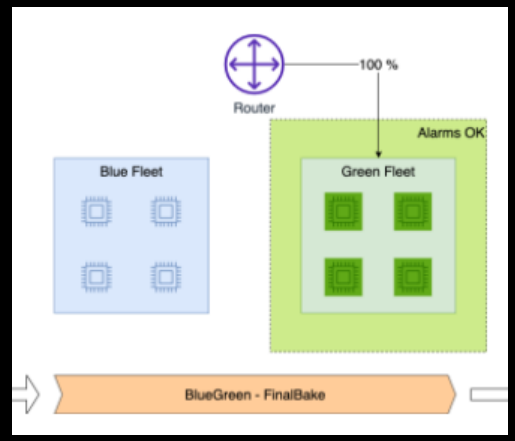
実験結果を可視化し最適なモデルを選択

Amazon SageMaker で使える様々な推論タイプ

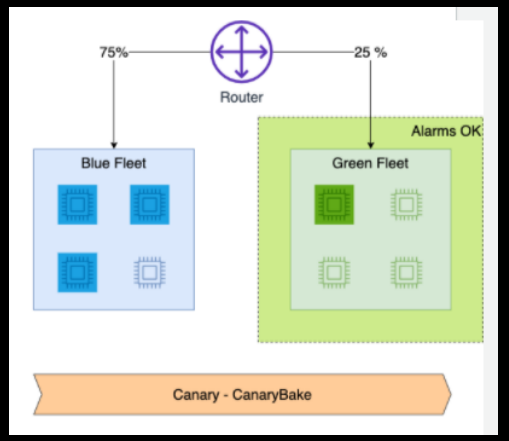
推論タイプ	用途	メリット・注意点	Quotas
リアルタイム推論 .deploy()	• アプリ上での推論処理などリアルタイムな推論が求められる場合	• 推論負荷に応じてオートスケーリング • 推論インスタンスを常時立ち上げるため、コストに注意	エンドポイント呼び出しの推論タイムアウトは1分
バッチ推論 .transform()	• 定期的なバッチ推論など、リアルタイム性が必要ない場合 • 都度推論実行の場合	• 推論処理が終わると推論インスタンスが自動終了するためインスタンス管理が不要	推論インスタンスは最大４まで
非同期推論	• 大きなサイズのペイロードを持つか、長い処理時間を必要とする推論向け	• リアルタイム推論ではタイムアウトしてしまう重めの処理をニアリアルタイムに実行	推論ペイロードが最大1GB、15分の推論まで対応
サーバレス推論 (preview)	• 断続的なトラフィック、または予測できないワークロード向け • コールドスタートが許容できる場合	• サーバーレス推論にも対応！ • 現在は preview 中 • アイドル時間の課金不要	エンドポイントメモリサイズは1~6GBまで選択

Amazon SageMaker で Blue/Green デプロイ

- SageMaker で Blue/Green デプロイのステージコントロールからエンドポイントの更新が可能に
- エンドポイントを一度に切り替えるモード(All At Once)、段階ごとにグリーンフリートで異常検知し問題がない場合にグリーンフリートにシフトするカナリヤモード(Canary)と段階的にトラフィックをシフトする線形モード(Linear)を提供

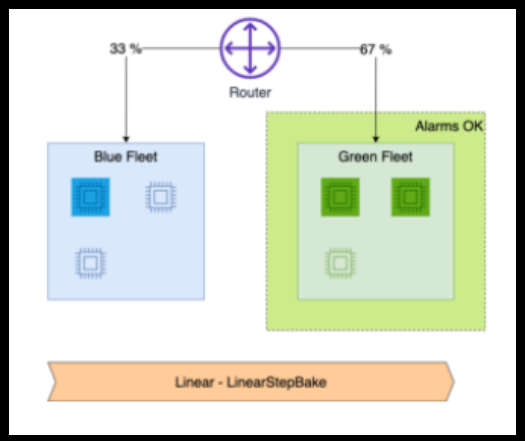


All at once



Canary

（グリーンステージで検証後シフト）



Linear

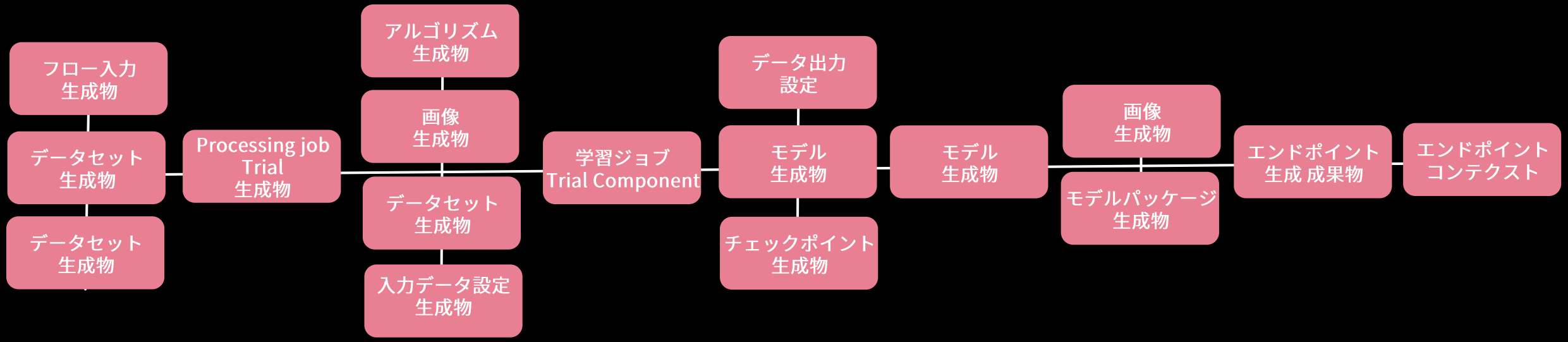
（シフトステップを線形増加）

段階的に新しいエンドポイントへリクエストを振り分け

https://docs.aws.amazon.com/sagemaker/latest/dg/deployment-guardrails-blue-green-canary.html

Amazon SageMaker データリネージュ

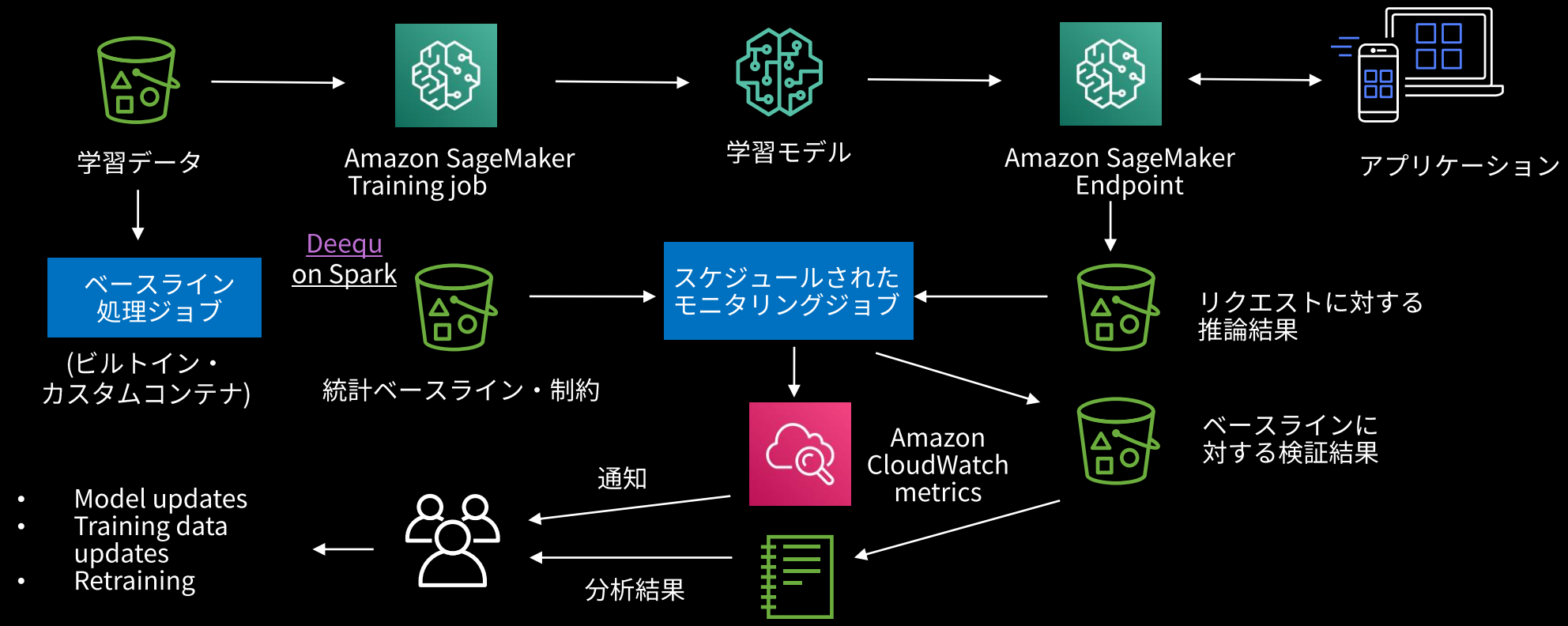
- ML ワークフローのすべてのステップのログを関連づけて記録・追跡
- 学習データ、プラットフォーム構成、モデルパラメータ、学習勾配などのモデル生成物の監査証跡を作成
- 監査証跡を使用してモデルを再作成可能、潜在的な問題をデバッグ
- クロスアカウントにも対応、コンプライアンス要件をサポート



ML ワークフロー時系列とデータリネージュで追跡可能な生成物の例

Amazon SageMaker Model Monitor エンドポイント監視

学習モデル作成時のデータの統計量をもとに予測モデルの妥当性を評価
ドリフト（不正確な予測）を自動的に検出してアラート



https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html
https://aws.amazon.com/blogs/big-data/test-data-quality-at-scale-with-deequ/

Amazon SageMaker を活用した MLOps ベストプラクティス

Amazon SageMaker を使って MLOps をはじめよう

STEP1

- SageMaker を使った基本構成（データ前処理・学習・推論）

STEP2

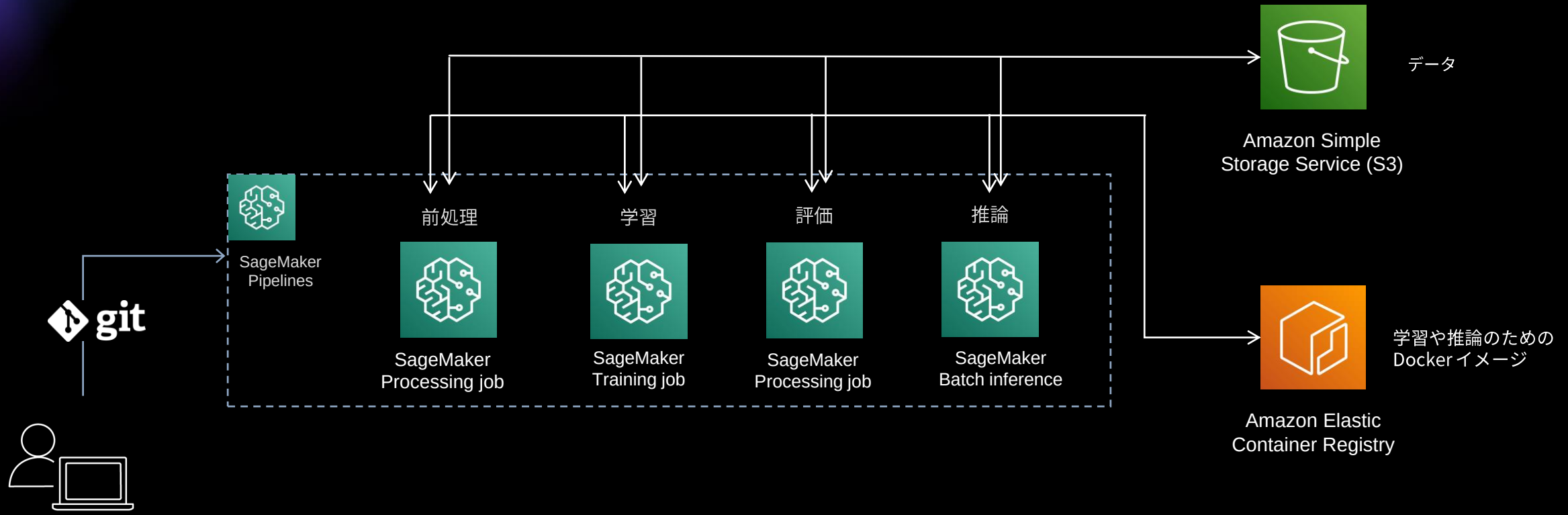
- Amazon SageMaker Pipelines を使った CI/CD パイプライン
- データ処理基盤や ETL 等と機械学習パイプラインの連携

STEP3

- 研究フェーズと開発運用フェーズの使い分け
- 本番環境のアプリ開発などの推論環境との連携・パイプライン自動化
- 既存のリソース（既存のデータ処理パイプライン、CI/CD ツール）との連携を考慮

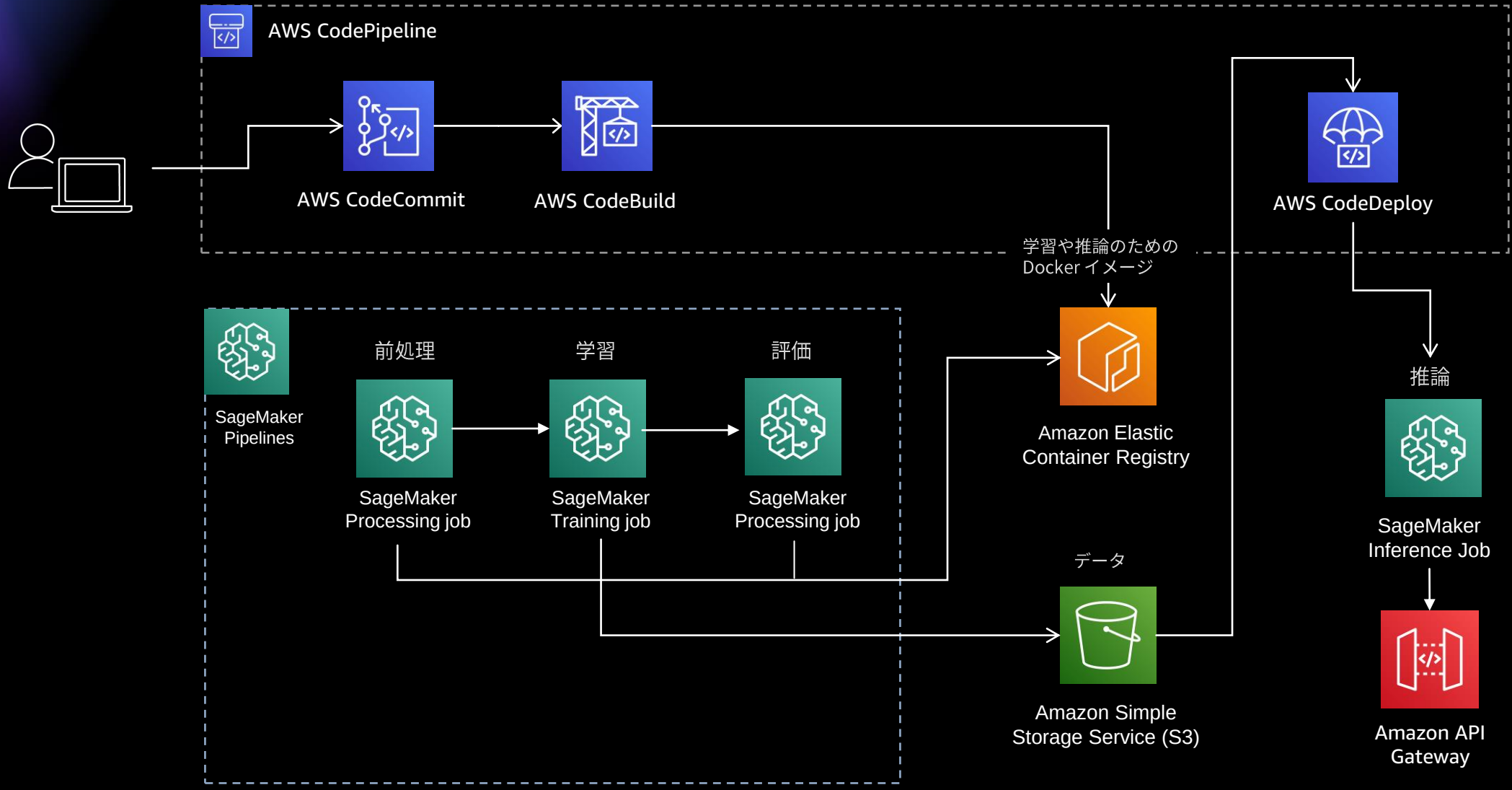
aws

Amazon SageMaker を使った MLOps 基本形

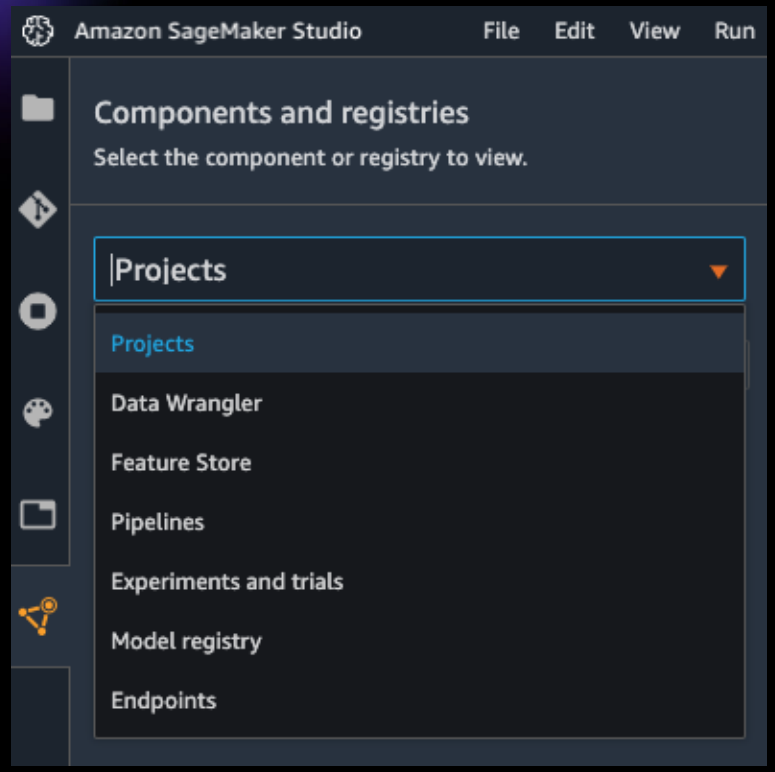


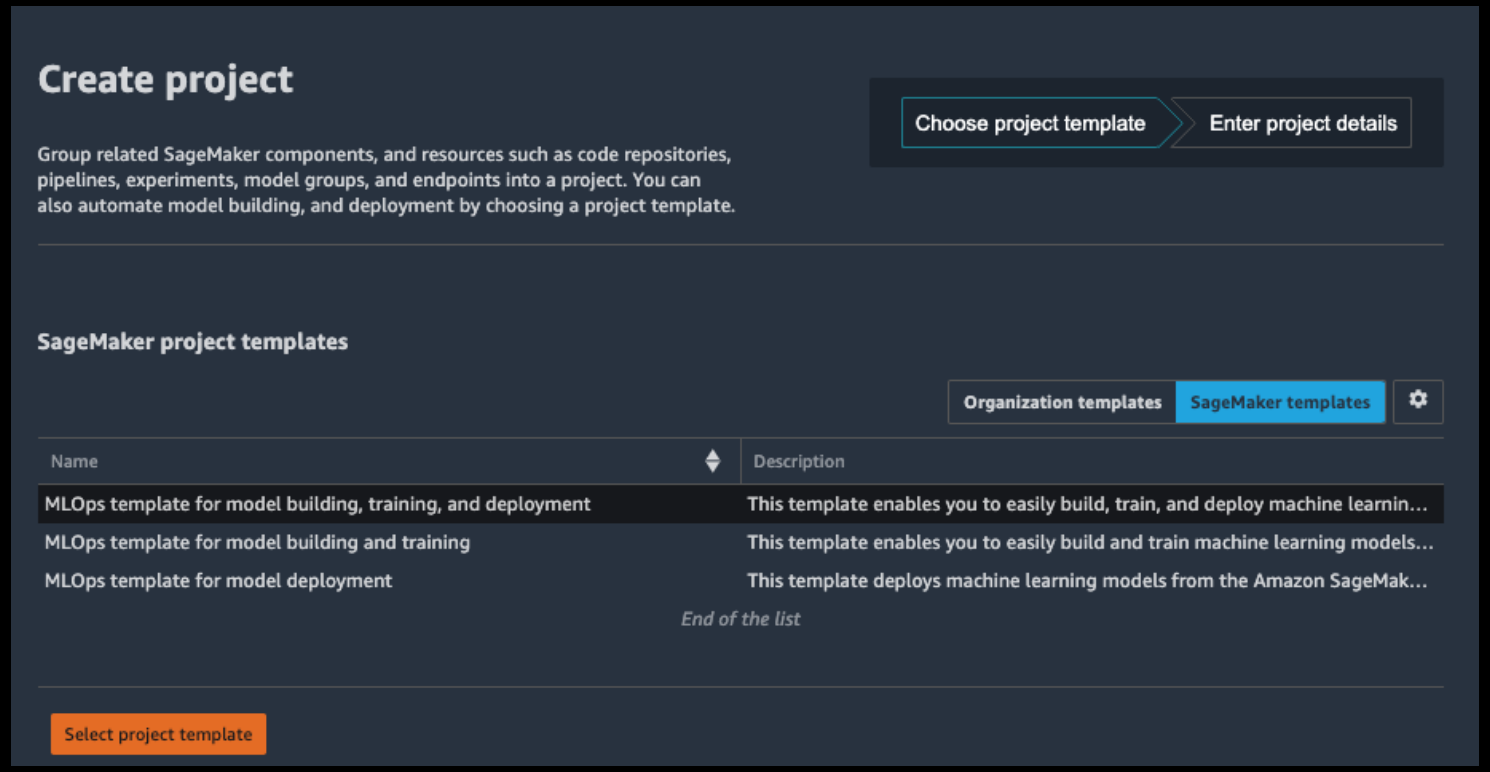
- 前処理・学習・評価・推論それぞれの用途に応じてコンテナを使い分ける
- マネージドサービスを使い、ML 基盤の保守・運用を効率化

Amazon SageMaker Pipelines とCI/CD



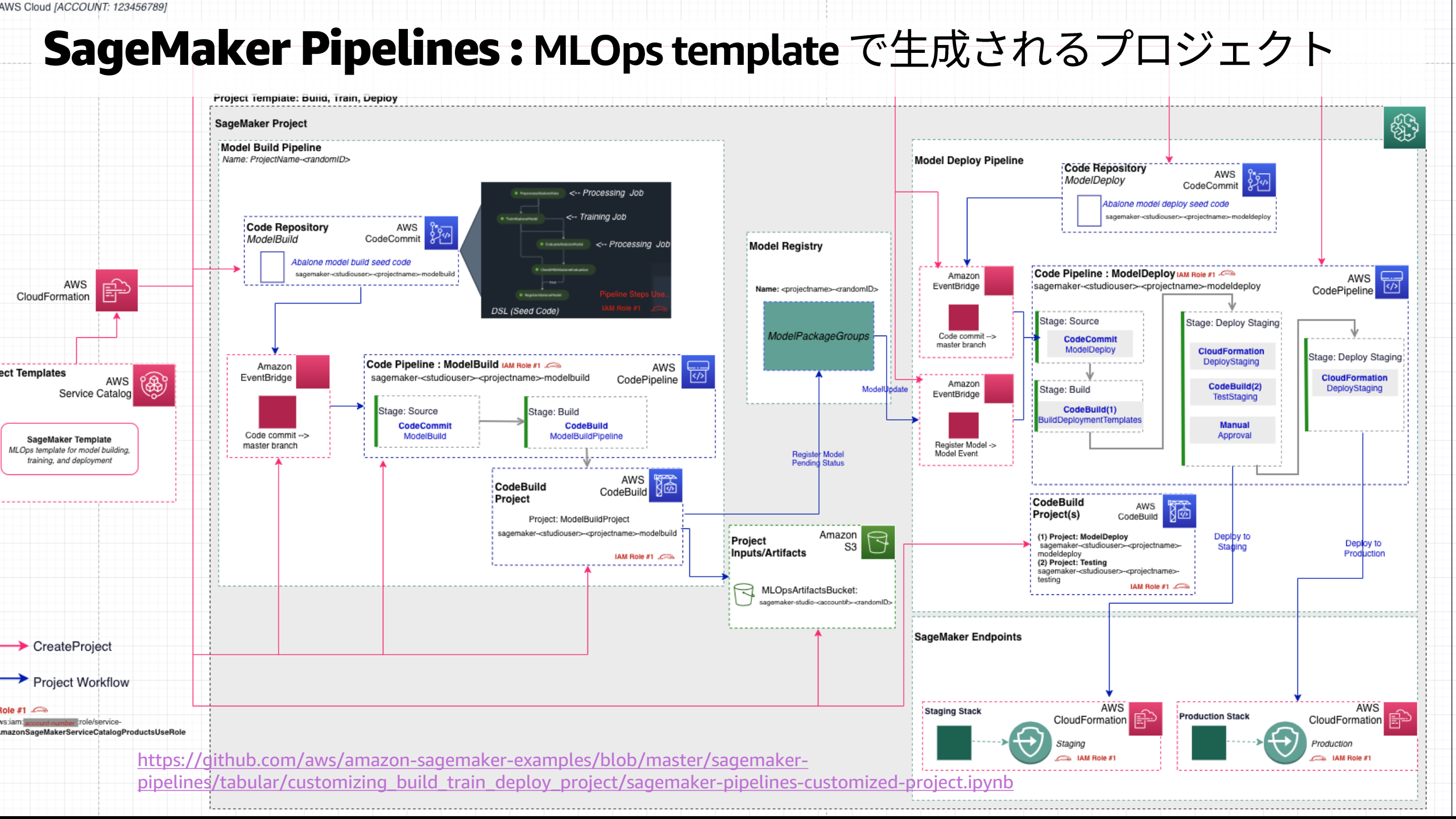
Amazon SageMaker Pipelines MLOps テンプレート





Amazon SageMaker Pipelinesでサンプルプロジェクトを選択し、Pipelines テンプレートを立ち上げる

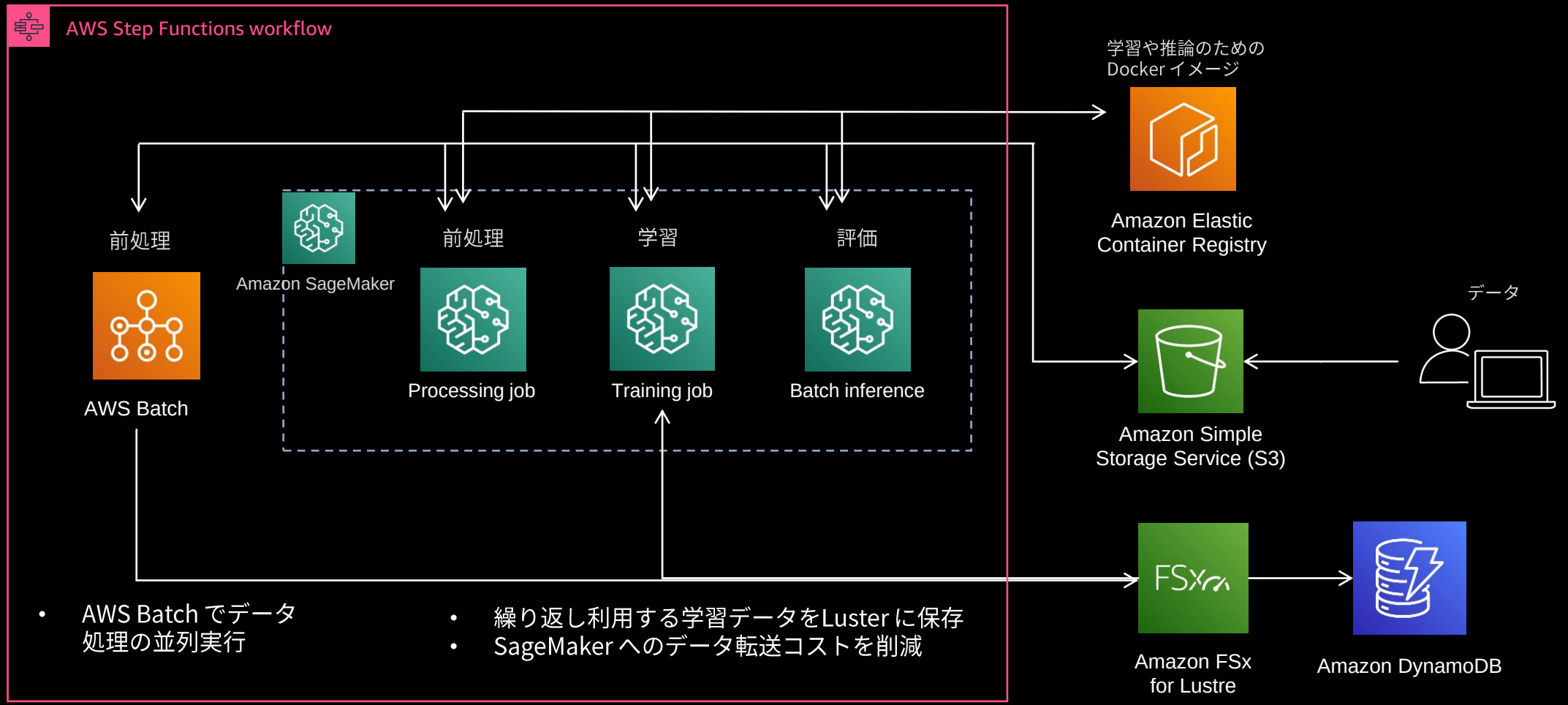
SageMaker Pipelines : MLOps template で生成されるプロジェクト



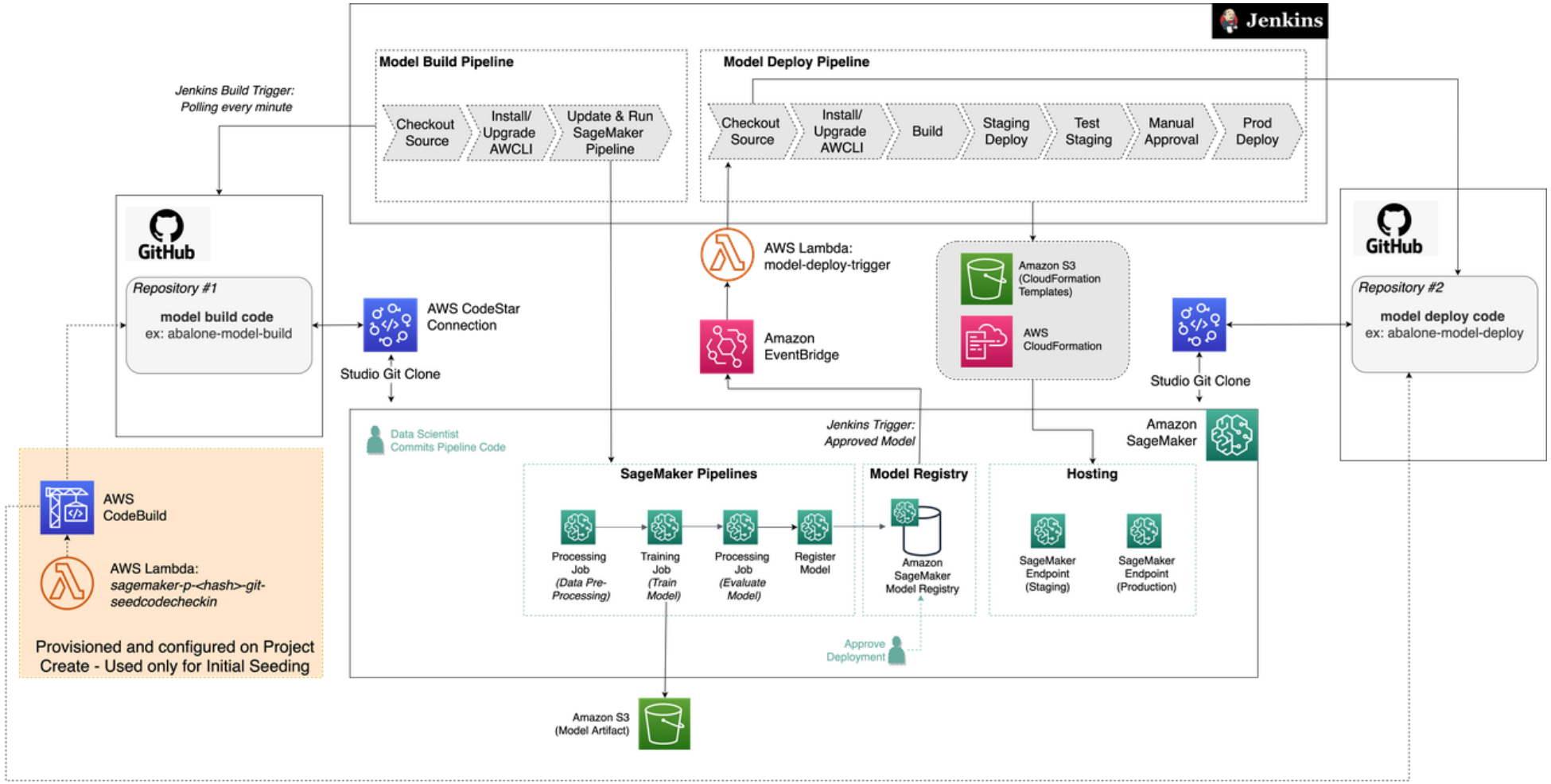
https://github.com/aws/amazon-sagemaker-examples/blob/master/sagemaker-pipelines/tabular/customizing_build_train_deploy_project/sagemaker-pipelines-customized-project.ipynb

ETL・データ基盤とMLプロセスの融合

日次のバッチ処理で学習を数時間以内に完了するパイプラインの設計例

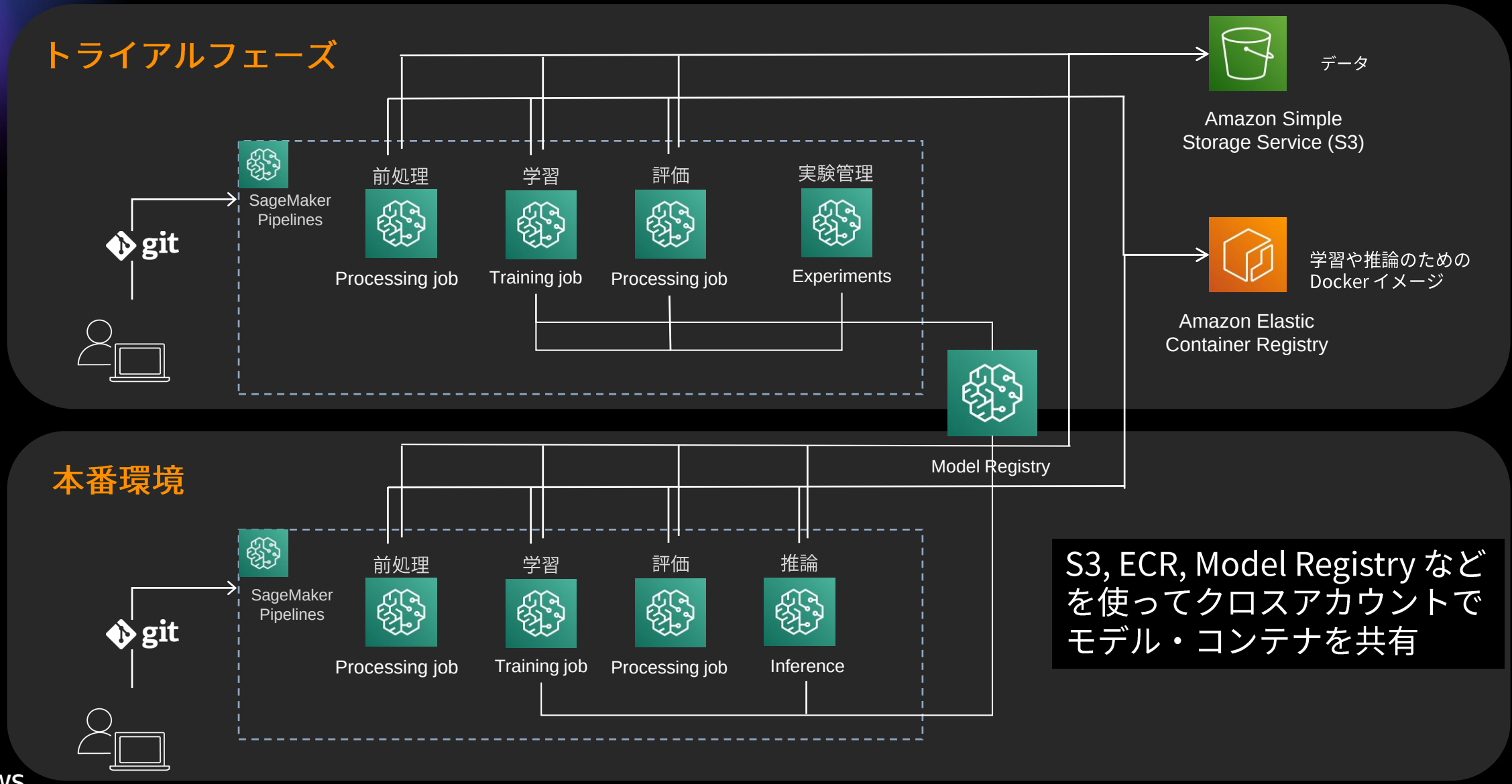


外部の CI/CD ツールと SageMaker Pipelines の連携



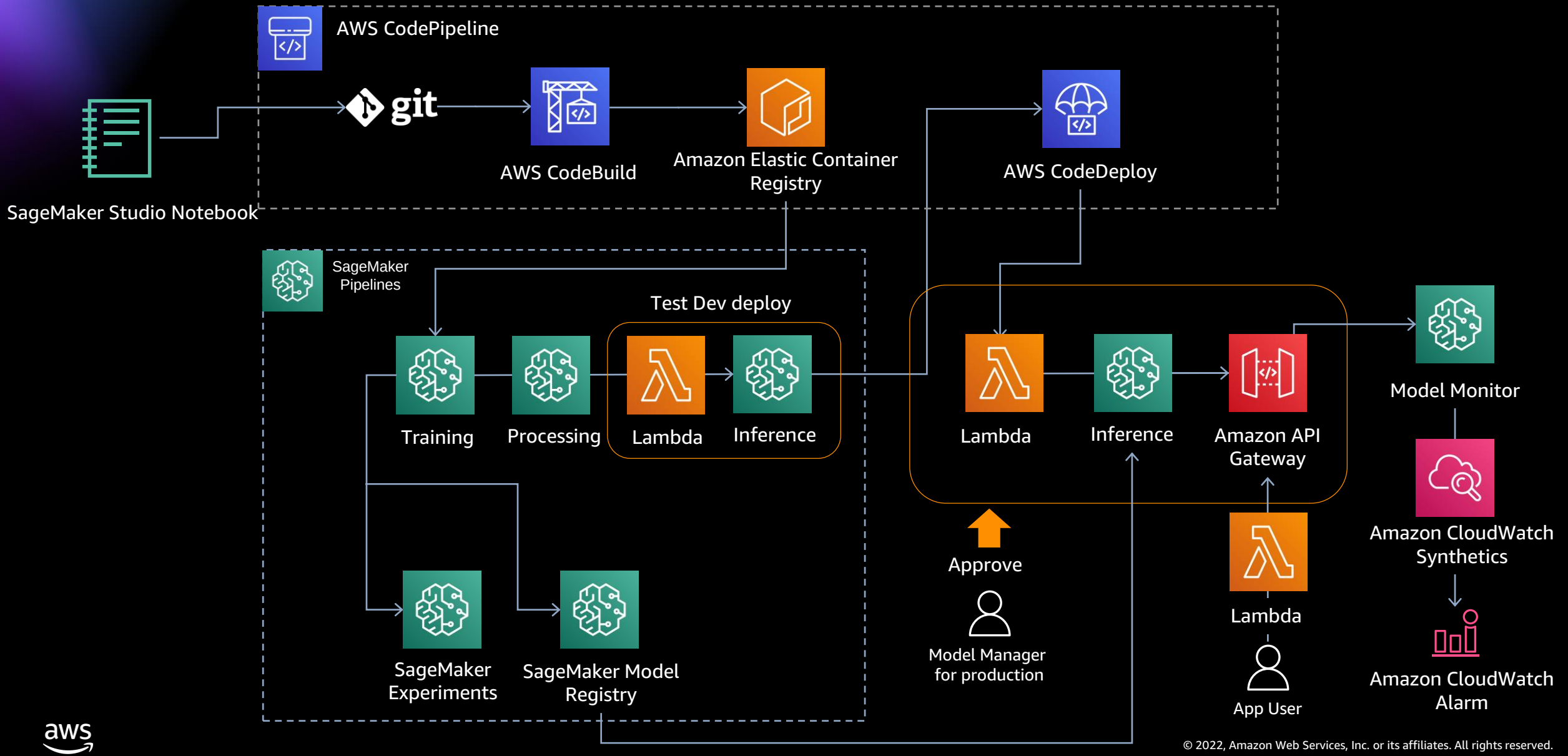
https://aws.amazon.com/jp/blogs/machine-learning/create-amazon-sagemaker-projects-using-third-party-source-control-and-jenkins/

フェーズに応じて開発環境を分ける



S3, ECR, Model Registry などを使ってクロスアカウントでモデル・コンテナを共有

本番環境へのリリースを考慮した安全なCI/CD



まとめ

- 運用基盤のコスト最適化 → Amazon SageMaker コンテナ 基本機能の活用
  - 学習・推論・運用基盤のコスト、最適化できていますか？
  - ML基盤の保守・運用が煩雑でメンテナンスに負荷がかかっていませんか？
- パイプライン自動化 → コードレポジトリとSageMaker Pipelines, SFn 等を連携
  - 機械学習ワークフロー、開発から運用までのコード管理～CI/CD どうしていますか？
  - 異なるチーム間で成果物（データ・モデル等）をどう受け渡していますか？
- データ・リソース管理 → SageMaker Model Registry, Projects, Experiments, Linage
  - 増え続けるモデル・データ・特徴量の管理、どこでどうしていますか？
  - トレーサビリティを担保できていますか？

まとめ

- Amazon SageMaker を軸に MLOps を構築
  - MLOps をマネージドに構築できる Amazon SageMaker を活用
  - Amazon SageMaker Pipelines を使ってスモールスタート
  - ML 以外の周辺条件・パイプラインに応じて最適なCI/CD ツールを選択
- Machine Learning Well-Architected Framework MLLens
  - 自身の機械学習環境をMLLensを用いてヘルスチェック、改善すべき要素を見つけて段階的に MLOps を育てよう

ご清聴ありがとうございました

Shoko Utsunomiya

Twitter: shokout