

事例から学ぶ MLOps 実装 Tips

三輪 麻衣子

プロフェッショナルサービス本部 シニアAI/MLコンサルタント
アマゾン ウェブ サービス ジャパン合同会社

自己紹介

三輪 麻衣子 (AI/ML Consultant)



担当エリア

- データ基盤の整備、可視化・分析、AI/ML活用など、お客様のデータ活用をプロジェクト型でご支援

好きなAWSサービス

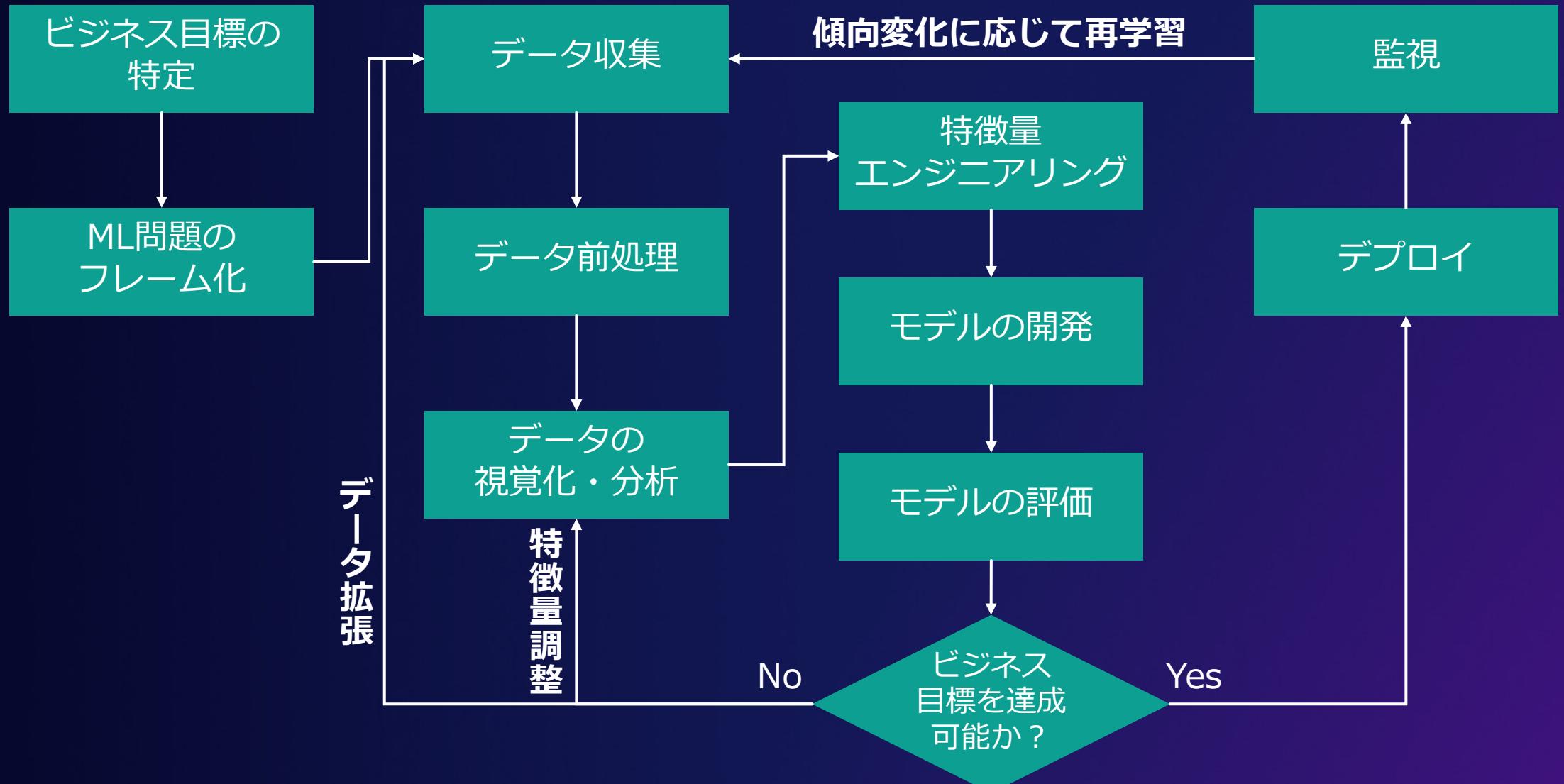
- AWS Glue、Amazon SageMaker

Agenda

1. MLOps とは
2. MLOps の成功に必要なもの
3. アーキテクチャ例と実装Tips
4. まとめ

MLOps とは

機械学習ワークフロー



機械学習ワークフローの特徴

モデル開発の試行錯誤に伴い、データ処理を繰り返す

新しいデータを利用した継続的なモデル改善が必要

管理対象が多い上、機密度が高いデータが含まれることも

手作業をなるべく排除して運用性・信頼性・セキュリティ・パフォーマンス・コスト効率を高めたい

MLOps

MLOps のポイント

データ

- データ加工処理をジョブ化し自動実行可能にする
- データの来歴（Data Lineage）を明らかにする
- どのデータでモデルを学習したか追跡可能にする

モデル

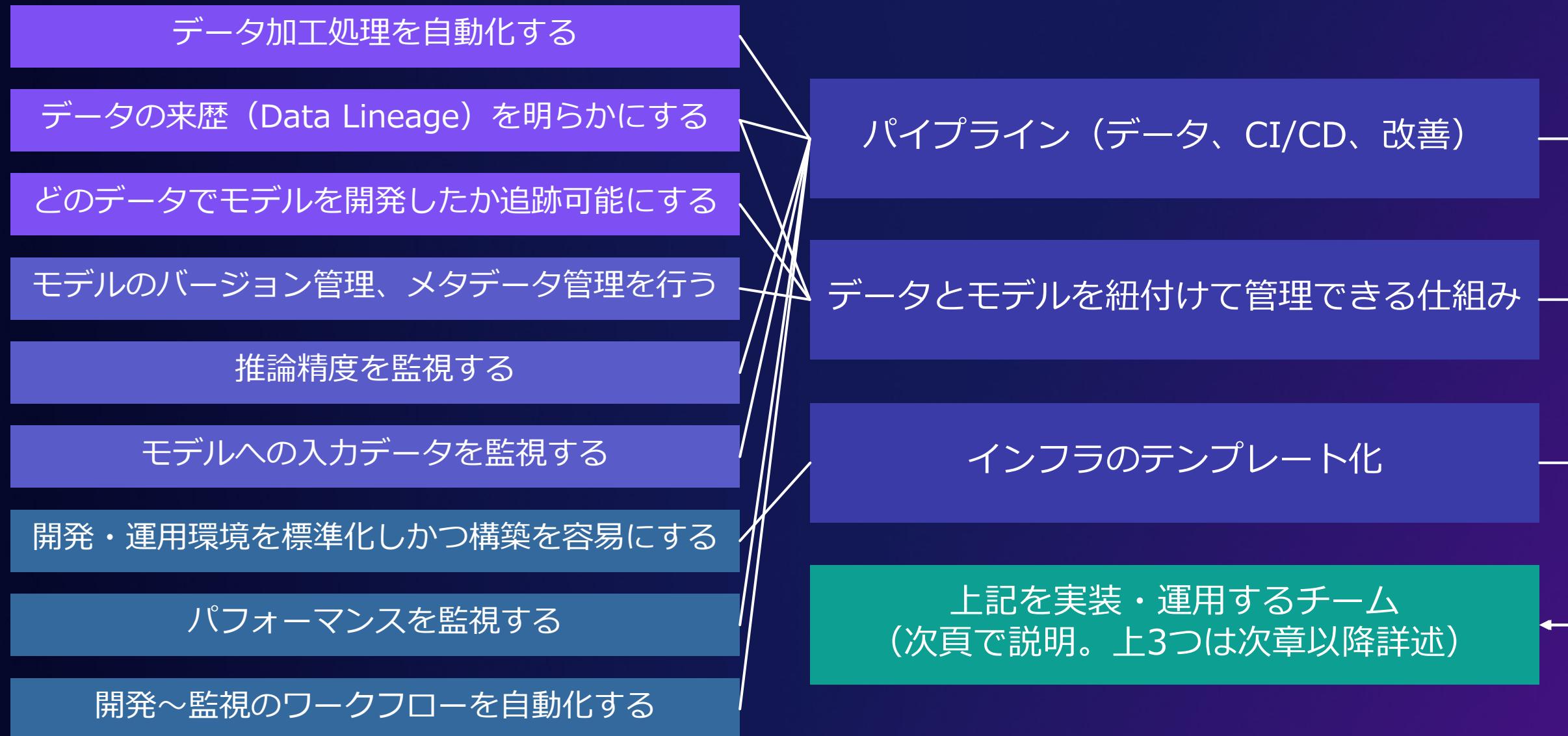
- モデルのバージョン管理、メタデータ管理を行う
- 推論出力・精度を監視する
- モデルへの入力データを監視する

インフラ

- 開発・運用環境を標準化しつつ構築を容易にする
- パフォーマンスを監視する
- 開発～監視のワークフローを自動化する

MLOps の成功に必要なもの

MLOps のポイント実現のために必要なもの



MLOps を実現するチーム



Data Engineer

- データパイプラインの構築
- データの管理



Data Scientist

- データの分析
- モデルの開発



ML Engineer

- モデルの本番化、デプロイ



MLOps Engineer

- 開発・運用環境の設計、構築、管理



System Administrator

- 必要なITリソースの用意
- インフラのテンプレート化
- システム監視



Security/Compliance Administrator

- システムの安全性の管理
- 監査対応

必要なスキルセットが多岐に渡るため、最初の組織整備が重要

アーキテクチャ例と実装Tips

アーキテクチャの考え方

1. 要件を明確にする

2. 理想形と現状の差分を確認し、ロードマップを描く

3. ツールの選定と設計を行う

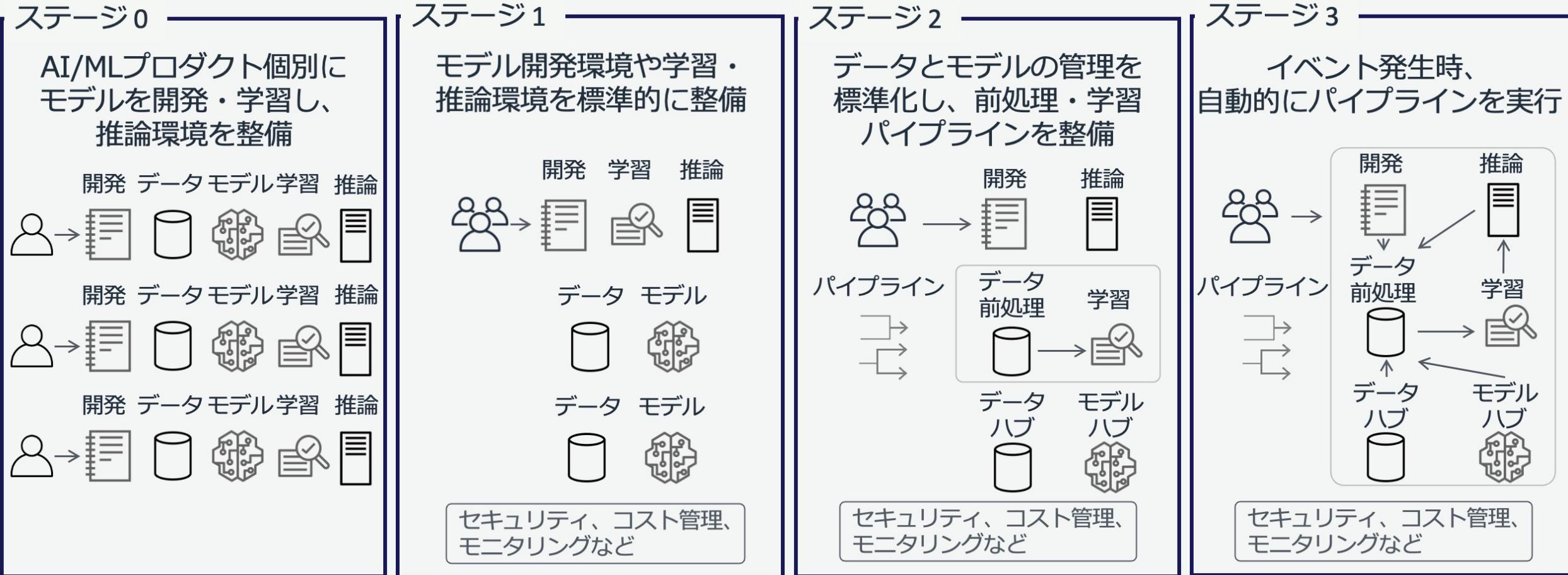
アーキテクチャの考え方

1. 要件を明確にする

- データ関連
処理が必要な量、頻度、アクセス制御の粒度、セキュリティ制約
- モデル関連
環境制約、デプロイ方式、承認フロー、改善フロー
- インフラ関連
性能要件、可用性、拡張性、自動化範囲、監視対象、運用フロー

アーキテクチャの考え方

2. 理想形と現状の差分を確認し、ロードマップを描く



アーキテクチャの考え方

3. ツールの選定と設計を行う

どこにでも適合するベストプラクティスは無い
ツールの特性を理解して、自社の要件に合うものを選定する

次ページから、選定と設計のポイント on AWS を解説

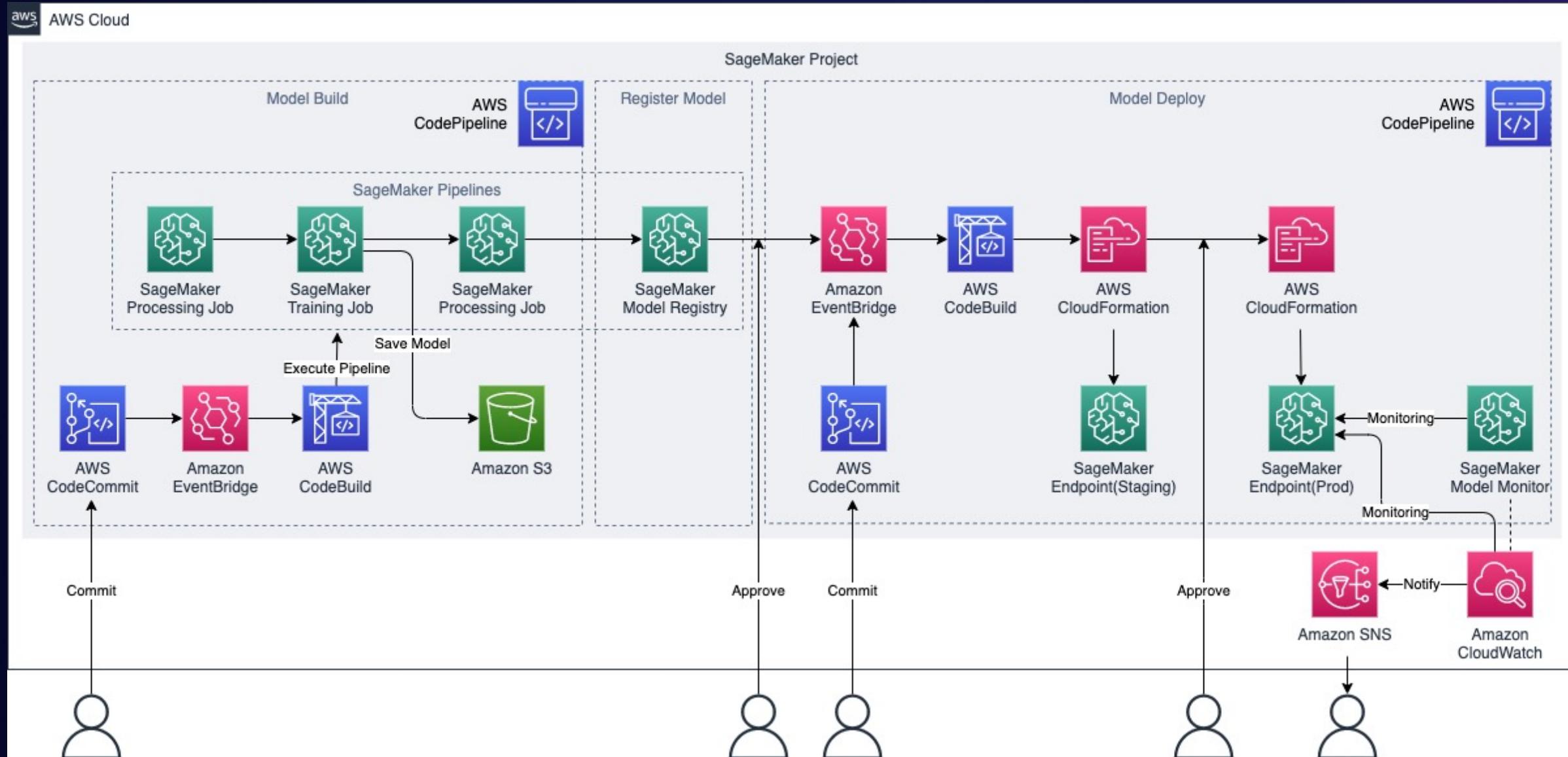
MLパイプライン実装選択肢

AWSサービス/Software	メリット	考慮点
Amazon SageMaker Pipelines 	<ul style="list-style-type: none">ほぼ SageMaker Studio UIで完結できるSageMaker Projectを利用すると、AWS CodeBuildによるモデルデプロイまで含めたパイプラインを 簡単に作成可能キャッシュ機能をONにするとエラー発生箇所のみの再実行が可能	<ul style="list-style-type: none">SageMaker SDKに慣れが必要
AWS Step Functions 	<ul style="list-style-type: none">GUIで開発できるサーバーレス	<ul style="list-style-type: none">メタデータ管理を作り込む必要がある部分再実行をするには工夫が必要データとモデルの管理の仕組みは別途考える必要がある
Amazon Managed Workflows for Apache Airflow 	<ul style="list-style-type: none">管理画面への情報集約度が高いPythonで柔軟に開発できるAWSサービス以外との連携も容易部分実行が可能	<ul style="list-style-type: none">環境起動中ずっと課金されるデータとモデルの管理の仕組みは別途考える必要がある
Kubeflow 	<ul style="list-style-type: none">OSSで完結させることも、AWSサービスと組み合わせて運用コストを下げることもできる	<ul style="list-style-type: none">部分再実行ができないコンテナ稼働環境の構築・運用が必要

MLデータ処理実装選択肢

AWSサービス/Software	メリット	考慮点
Amazon SageMaker Processing 	<ul style="list-style-type: none">ほぼ <u>SageMaker Studio UIで完結できる</u>SageMaker Data Wranglerを使い<u>GUIで開発できる</u>Sparkコンテナを使えば<u>データの大規模分散処理も可能</u>	<ul style="list-style-type: none">SageMaker SDKに慣れが必要
Amazon Managed Workflows for Apache Airflow 	<ul style="list-style-type: none">ワークフローをMWAAで作成している場合、<u>手軽に処理を組み込む</u>れる	<ul style="list-style-type: none">環境起動中ずっと課金される大規模データ（TB単位）の処理には向かない
AWS Glue Job 	<ul style="list-style-type: none"><u>GUIで開発できる</u><u>データの大規模分散処理が可能</u>	<ul style="list-style-type: none">GUIだけで実装できる処理がSageMaker Data Wranglerに比べると少ない
AWS Lambda 	<ul style="list-style-type: none">軽い処理を<u>スピーディーに低成本</u>で実行できる	<ul style="list-style-type: none">大規模データ（TB単位）の処理には向かない
Kubeflow 	<ul style="list-style-type: none">OSSで完結させることも、AWSサービスと組み合わせて運用コストを下げることもできる	<ul style="list-style-type: none">コンテナ稼働環境の構築・運用が必要

アーキテクチャ例① テンプレートで簡単立ち上げ



アーキテクチャ例① テンプレートで簡単立ち上げ

マッチする状況

- 既存の基盤やパイプラインがない状態から、なるべく手をかけずに典型的な基盤を構築したい（次頁参照）
- 学習が必要なAWSサービスを絞りたい

実装Tips

- 大規模なデータの前処理（TB単位）が必要な場合はProcessing JobにSparkを使用する
- SageMakerで開発した既存コードをパイプラインに移植する際は、パイプラインのステップが繋がる書き方に変更することを忘れずに（次々頁参照）

SageMaker Pipelines の構築

Amazon SageMaker Studio

Components and registries
Select the component or registry to view.

Projects

Search
Create project

Name
project-includemonitoring
project-maxtemplate
End of the list

Create project

Group related SageMaker components, and resources such as code repositories, pipelines, experiments, model groups, and endpoints into a project. You can also automate model building, and deployment by choosing a project template.

Choose project template Enter project details

SageMaker project templates

Name Description

- MLOps template for model building, training, deployment and monitoring
- MLOps template for image building, model building, and model deployment
- MLOps template for model building, training, and deployment with third-party dependencies
- MLOps template for model building, training, and deployment
- MLOps template for model deployment
- MLOps template for model building and training
- MLOps template for model building, training, and deployment with third-party dependencies

Organization templates SageMaker templates

CloudFormation > スタック > SC-315047899393-pp-pvtexehete3v2u

SC-315047899393-pp-pvtexehete3v2u 削除 更新 スタックアクション ▾ スタックの作成 ▾

アクティブ▼ ネスト表示

CREATE_COMPLETE

裏でCloudFormationスタックが作成され、必要なAWSリソースが生成されるという仕組み

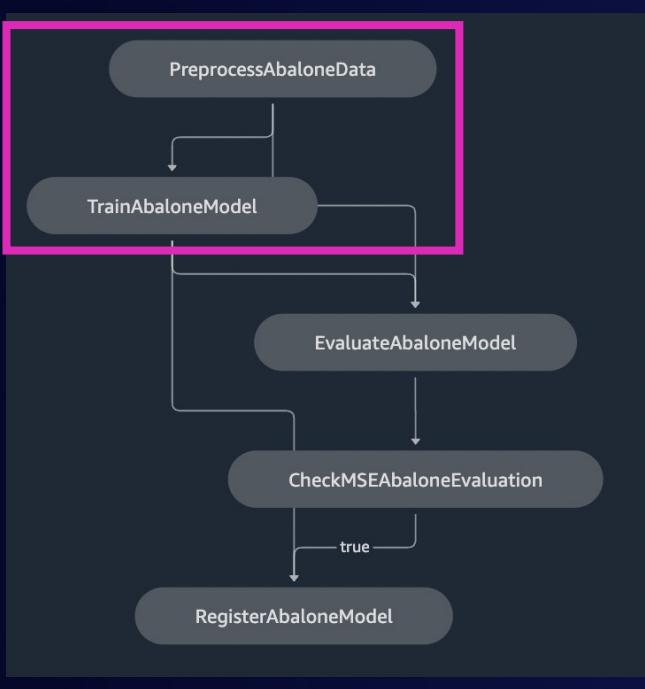


SageMaker Studioで「Project」を作成すると、選択したテンプレートに応じたパイプラインが自動的に作成されるので、それをベースに自社環境に合わせて編集をする（独自に作成したテンプレートを登録することも可能）

リソース ID	物理 ID	タイプ	ステータス	状況の理由	モジュール
GetBaselinesAndConfigs	sagemaker-project-includemonitoring-p-llscuzknkqdi-modelmonitor	AWS::CodeBuild::Project	CREATE_COMPLETE	-	-
InServiceEndpointEventRule	sagemaker-project-includemonitoring-p-llscuzknkqdi-ep	AWS::Events::Rule	CREATE_COMPLETE	-	-
MIOpsArtifactsBucket	sagemaker-project-p-llscuzknkqdi	AWS::S3::Bucket	CREATE_COMPLETE	-	-
ModelBuildCodeCommitEventRule	sagemaker-project-includemonitoring-p-llscuzknkqdi-	AWS::Events::Rule	CREATE_COMPLETE	-	-

SageMaker Pipelines の実装コード例

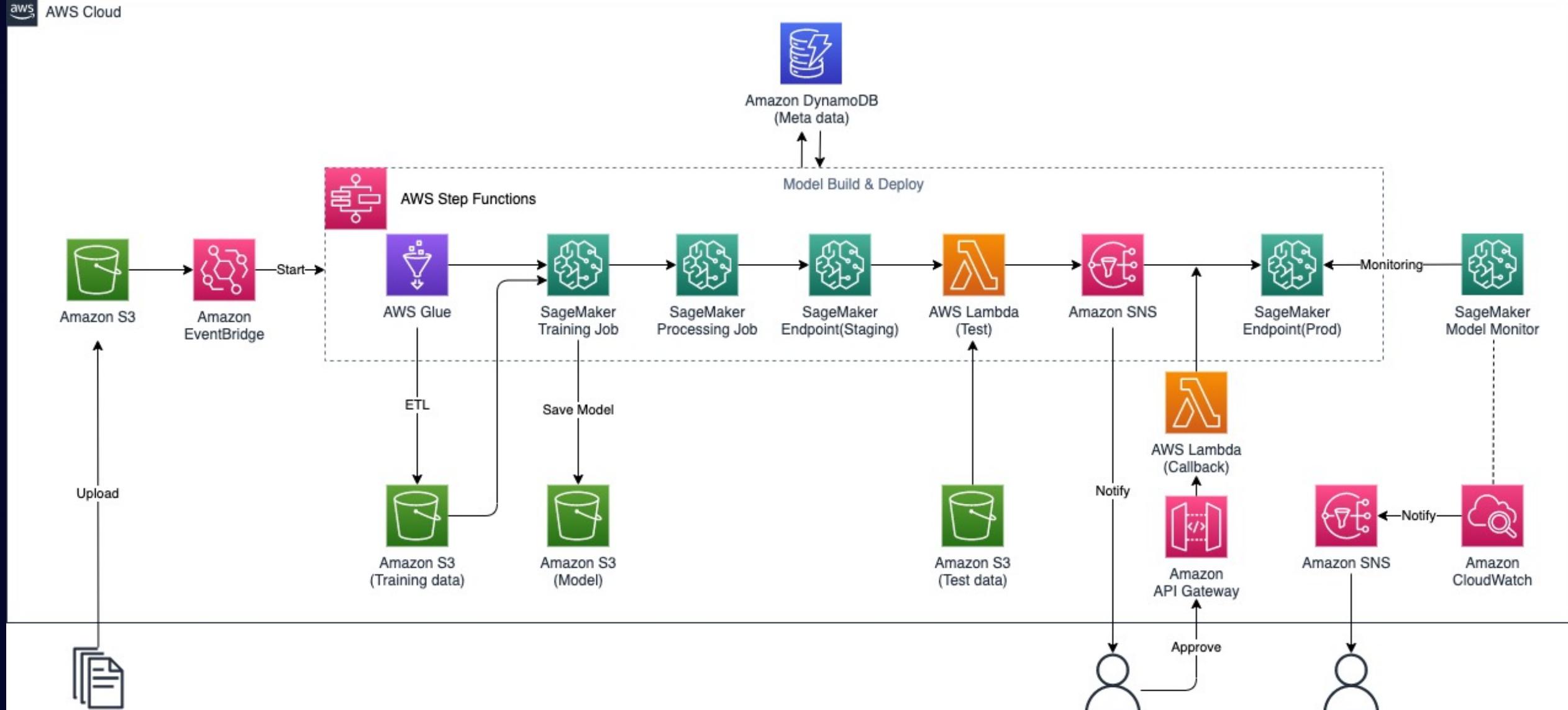
```
120 step_process = ProcessingStep(  
121     name="PreprocessAbaloneData",  
122     processor=sklearn_processor,  
123     outputs=[  
124         ProcessingOutput(output_name="train", source="/opt/ml/processing/train"),  
125         ProcessingOutput(output_name="validation", source="/opt/ml/processing/validation"),  
126         ProcessingOutput(output_name="test", source="/opt/ml/processing/test"),  
127     ],  
128     code=os.path.join(BASE_DIR, "preprocess.py"),  
129     job_arguments=["--input-data". input_data]  
130 )
```



ProcessingStep (前処理) の Outputs を TrainingStep (学習)
で呼び出す書き方をするとステップが繋がる

```
160     step_train = TrainingStep(  
161         name="TrainAbaloneModel",  
162         estimator=xgb_train,  
163         inputs={  
164             "train": TrainingInput(  
165                 s3_data=step_process.properties.ProcessingOutputConfig.Outputs[  
166                     "train"  
167                 ].S3Output.S3Uri,  
168                 content_type="text/csv",  
169             ),  
170             "validation": TrainingInput(  
171                 s3_data=step_process.properties.ProcessingOutputConfig.Outputs[  
172                     "validation"  
173                 ].S3Output.S3Uri,  
174                 content_type="text/csv",  
175             ),  
176         },  
177     )
```

アーキテクチャ例② GUIワークフロー + 大規模データ



アーキテクチャ例② GUIワークフロー + 大規模データ

マッチする状況

- GUIでパイプラインを作成したい or Step Functionsに習熟している
- 前処理対象データが多く（TBレベル）、分散処理を行いたい

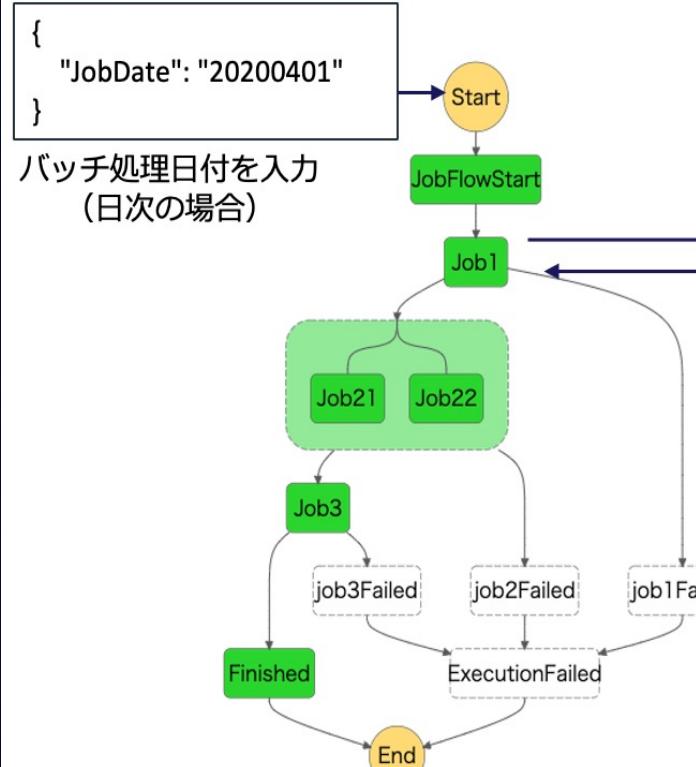
実装Tips

- メタデータの受け渡しを作り込む必要がある。データセットとモデルの紐付
き管理もここに含めるか、SageMaker Model Registryを使う
- 問題発生時に途中からリトライしたい場合は工夫が必要（次頁参照）

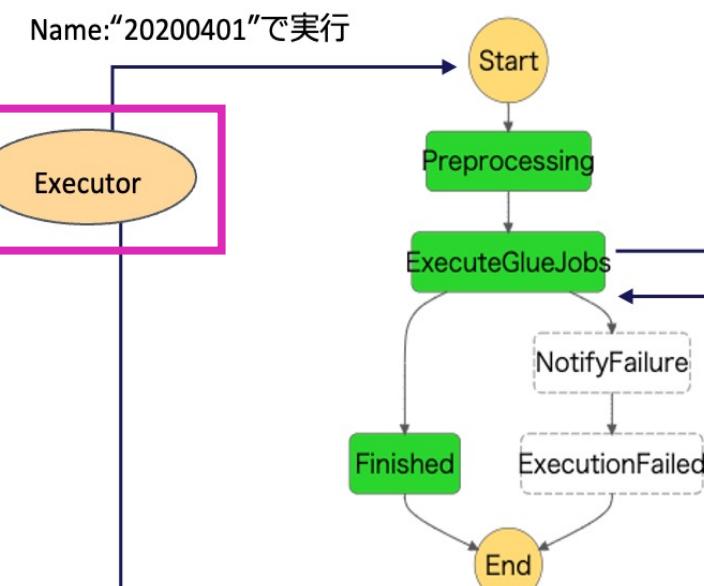
AWS Step Functions でのリトライ実装

中間に单機能のステートマシン「Executor」を介在させる

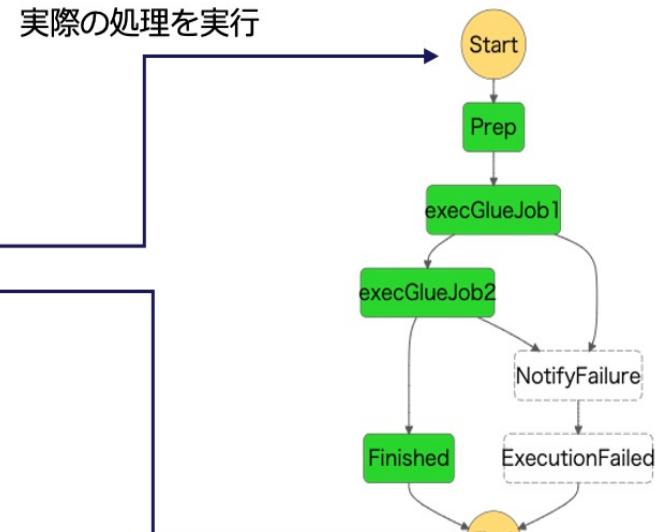
ジョブフローを定義する
ステートマシン



リランの単位となる
ステートマシン



実際の処理を記述する
ステートマシン

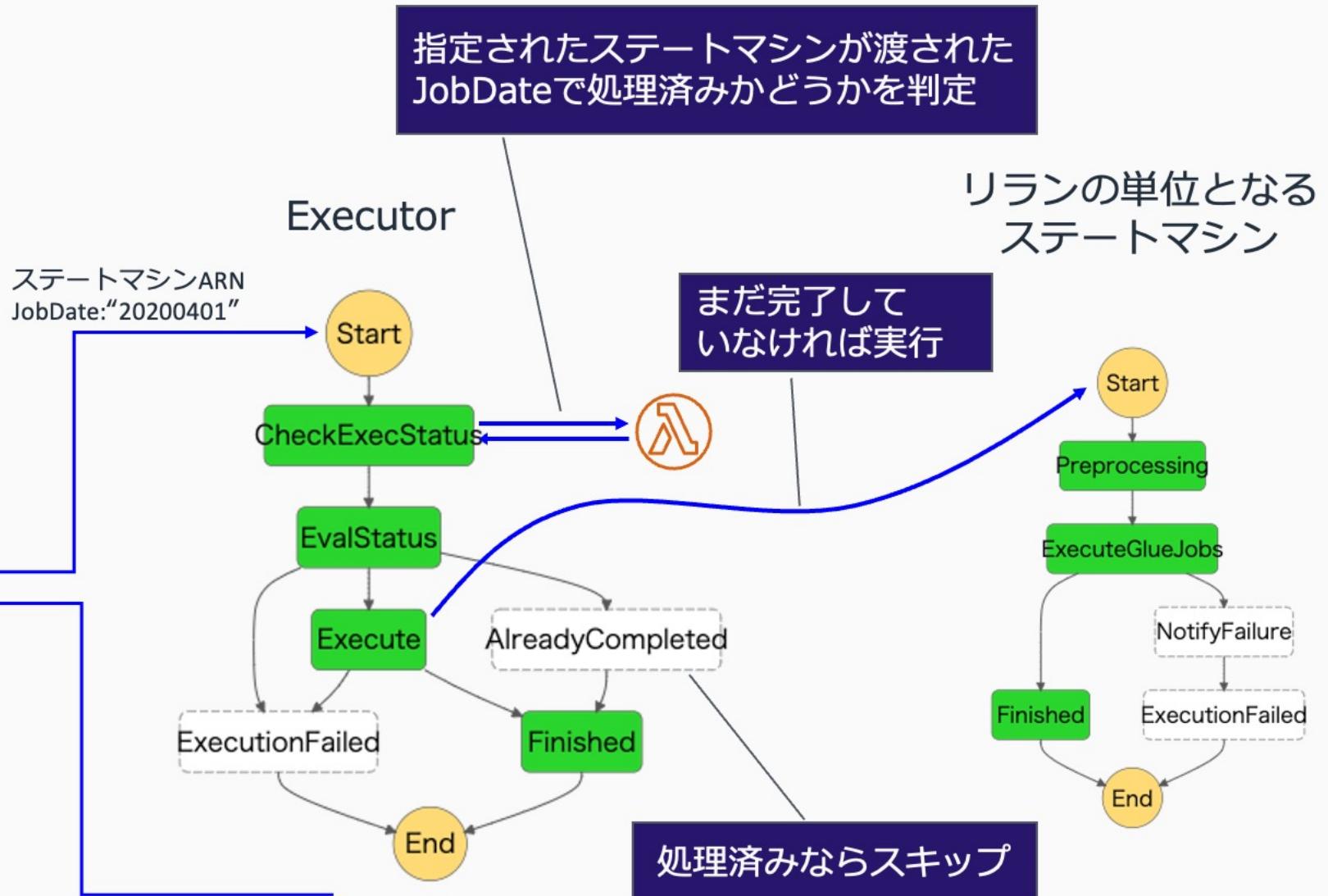
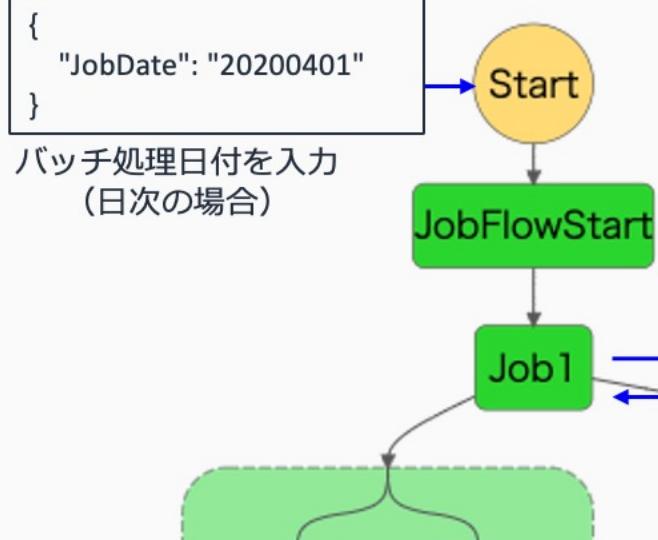


リラン時の再処理を
許容できる範囲で定義する

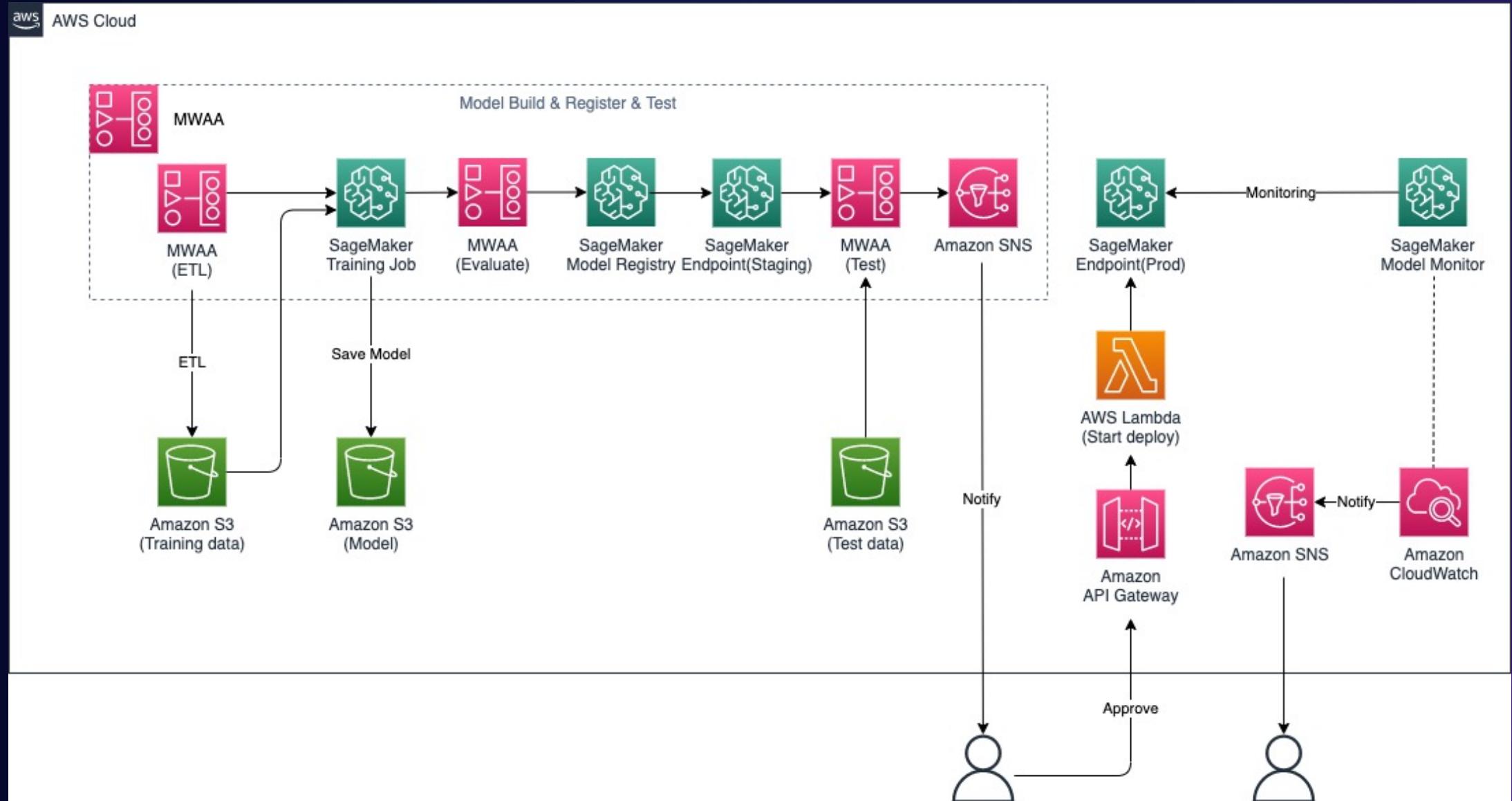
実処理としてまとめ
る範囲で定義する

AWS Step Functions でのリトライ実装

ジョブフローを定義する
ステートマシン



アーキテクチャ例③ 柔軟・管理しやすいワークフロー



アーキテクチャ例③ 柔軟・管理しやすいワークフロー

マッチする状況

- ・ ワークフローをPythonで自由度高く開発したい or Airflowに習熟している
- ・ ワークフロー管理画面をなるべく集約し、複数のAWSサービスの管理画面を参照しなくて済むようにしたい（次頁参照）

実装Tips

- ・ 環境起動中は処理のない時も課金される。停止機能がないので、処理頻度が低い場合は環境作成/削除を自動化して必要時のみ起動してコスト節約するという工夫もあり。ただし環境を削除するとメタデータも消えるので、実行履歴を保持したい場合はS3にCSVとして退避するといった対処が必要

MWAA 管理画面

Airflow DAGs Security Browse Admin Docs

Triggered etl_glue_job, it should start any moment now.

DAGs

All 2 Active 1 Paused 1

Filter DAGs by tag Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
etl_athena_job	airflow/handson	1	*/60 * * * *	2022-03-09, 21:59:25	2022-03-09, 22:00:00	1
etl_glue_job	airflow/handson	3	*/60 * * * *	2022-03-09, 22:09:14	2022-03-09, 22:00:00	4

Showing 1-2 of 2 DAGs

Airflow DAGs Security Browse Admin Docs

Task Instance: athena_create_input_table at: 2022-03-09, 21:59:25 UTC

Instance Details Rendered Log All Instances Filter Upstream

Download Log (by attempts): 1

Task Actions

Ignore All Deps Ignore Task State Ignore Task Deps Run

Past Future Upstream Downstream Recursive Failed Clear

Past Future Upstream Downstream Mark Failed

Past Future Upstream Downstream Mark Success

Close

Airflow DAGs Security Browse Admin Docs

DAG: etl_athena_job etl athena DAG

running Schedule: */60 * * * * Next Run: 2022-03-09, 21:00:00

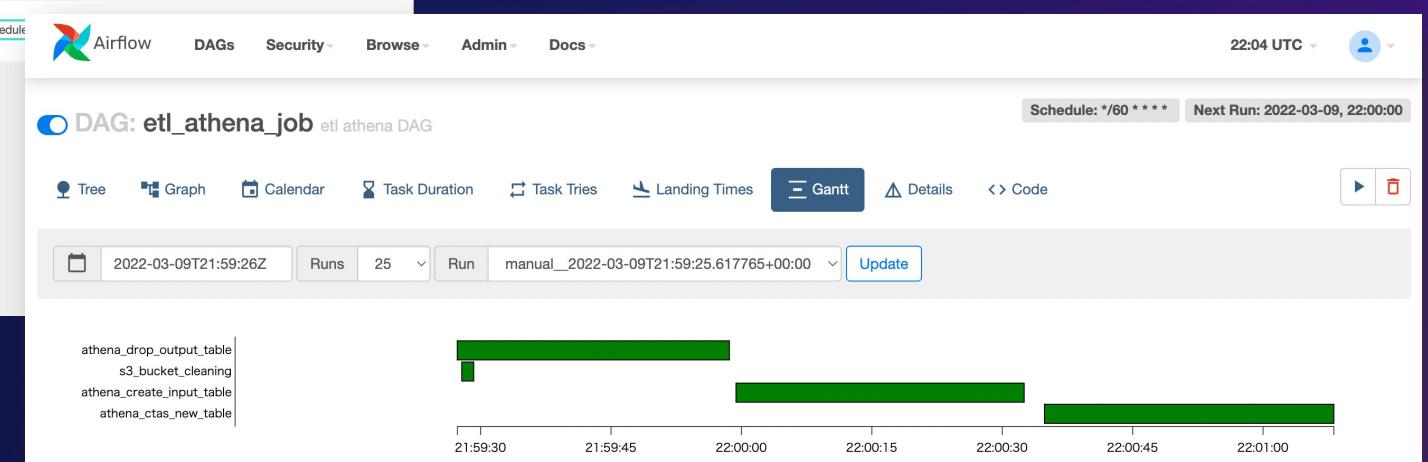
Tree Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code

2022-03-09T21:59:26Z Runs 25 Run manual_2022-03-09T21:59:25.617765+00:00 Layout Left > Right Update Find Task...

AWSAthenaOperator PythonOperator

queued running success failed up_for_retry up_for_reschedule

```
graph LR; athena_drop_output_table --> athena_create_input_table; athena_create_input_table --> athena_ctas_new_table; s3_bucket_cleaning --> athena_create_input_table
```



MWAA 管理画面

DAG: `etl_athena_job` etl athena DAG

failed Schedule: */60 * * * *

Tree Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code

2022-03-09T21:52:41Z Runs 25 Run manual__2022-03-09T21:52:40.616703+00:00 Layout Left > Right Update

AWSAthenaOperator PythonOperator

queued running success failed up_for_retry up_for_reschedule upstream_failed skip

```
graph LR; athena_drop_output_table[athena_drop_output_table] --> athena_create_input_table[athena_create_input_table]; s3_bucket_cleaning[s3_bucket_cleaning] --> athena_create_input_table; athena_create_input_table --> athena_ctas_new_table[athena_ctas_new_table]
```

21:57 UTC

Schedule: */60 * * * *

DAG: etl_athena_job etl athena DAG

Tree Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code

Task Instance: s3_bucket_cleaning at 2022-03-09, 20:00:00

Task Instance Details Rendered Template Log XCom

Log by attempts

1 2

Jump To End Toggle Wrap

```
** Reading remote log from Cloudwatch log_group: airflow-MyAirflowEnvironment-Task log_stream: etl_athena_job/s3_bucket_cleaning/2022-03-09T20_00_00+00_00/2.log.
[2022-03-09, 21:54:46 UTC] {{taskinstance.py:1035}} INFO - Dependencies all met for <TaskInstance: etl_athena_job.s3_bucket_cleaning scheduled_2022-03-09T20:00:00>
[2022-03-09, 21:54:46 UTC] {{taskinstance.py:1035}} INFO - Dependencies all met for <TaskInstance: etl_athena_job.s3_bucket_cleaning scheduled_2022-03-09T20:00:00>
[2022-03-09, 21:54:46 UTC] {{taskinstance.py:1241}} INFO -
```

```
[2022-03-09, 21:54:46 UTC] {{taskinstance.py:1242}} INFO - Starting attempt 2 of 2
[2022-03-09, 21:54:46 UTC] {{taskinstance.py:1243}} INFO -
```

```
[2022-03-09, 21:54:46 UTC] {{taskinstance.py:1262}} INFO - Executing <Task(PythonOperator): s3_bucket_cleaning> on 2022-03-09 20:00:00+00:00
[2022-03-09, 21:54:47 UTC] {{standard_task_runner.py:52}} INFO - Started process 46 to run task
[2022-03-09, 21:54:47 UTC] {{standard_task_runner.py:52}} INFO - Started process 46 to run task
[2022-03-09, 21:54:47 UTC] {{standard_task_runner.py:77}} INFO - Job 7: Subtask s3_bucket_cleaning
[2022-03-09, 21:54:47 UTC] {{base_aws.py:401}} INFO - Airflow Connection: aws_conn_id=aws_default
[2022-03-09, 21:54:47 UTC] {{base_aws.py:190}} INFO - No credentials retrieved from Connection
[2022-03-09, 21:54:47 UTC] {{base_aws.py:93}} INFO - Creating session with aws_access_key_id=None region_name=ap-northeast-1
[2022-03-09, 21:54:47 UTC] {{base_aws.py:168}} INFO - role_arn is None
[2022-03-09, 21:54:47 UTC] {{logging_mixin.py:109}} INFO - Running <TaskInstance: etl_athena_job.s3_bucket_cleaning scheduled_2022-03-09T20:00:00> [running]>
[2022-03-09, 21:54:47 UTC] {{taskinstance.py:1429}} INFO - Exporting the following env vars:
```

DAG: etl_athena_job etl athena DAG

Schedule: */60 * * * *

Tree Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code

DAG Details

failed 3 success 9 upstream_failed 4

Schedule Interval	*/60 * * * *
Catchup	False
Started	True
End Date	None
Max Active Runs	0 / 16
Concurrency	10000
Default Args	{'email_on_failure': True, 'email_on_retry': True, 'owner': 'airflow', 'provide_context': False, 'retries': 1, 'retry_delay': 120, 'start_date': DateTime(2022, 3, 8, 0, 0, 0, tzinfo=Timezone('UTC'))}
Tasks Count	4
Task IDs	['athena_drop_output_table', 's3_bucket_cleaning', 'athena_create_input_table', 'athena_ctas_new_table']

List Task Instance

Search:

Action	State	Dag Id	Task Id
<input type="checkbox"/>	failed	etl_athena_job	s3_bucket_cleaning
<input type="checkbox"/>	failed	etl_athena_job	s3_bucket_cleaning
<input type="checkbox"/>	failed	etl_athena_job	athena_ctas_new_table

List Task Instance																				
Search +																				
Actions 																				
State	Dag Id	Task Id	Run Id	Execution Date	Operator	Start Date	End Date	Duration	Job Id	Hostname	Username	Priority	Weight	Queue	Queued Dttm	Try Number	Pool	Queued By Job Id	External Execution Id	Log Url
  	 etl_athena_job	s3_bucket_cleaning	Y	scheduled_2022-03-09T20:00:00+00:00	2022-03-09, 20:00:00	PythonOperator	2022-03-09, 21:54:46	2022-03-09, 21:54:47	0:00:01.226945	7	ip-10-192-21-66.ap-northeast-1.compute.internal	airflow	3	airflow celery 178319b... fe3d-487a-... b37c-... b1b83a467bb6d	2022-03-09, 21:54:45	3	default_pool	1	10004af0-8d45-4209-82f6-73688ba74166	
  	 etl_athena_job	s3_bucket_cleaning	Y	manual_2022-03-09T21:52:40.6161703j+00:00	2022-03-09, 21:52:40	PythonOperator	2022-03-09, 21:54:47	2022-03-09, 21:54:49	0:00:01.861719	8	ip-10-192-21-66.ap-northeast-1.compute.internal	airflow	3	airflow celery 178319b... fe3d-487a-... b37c-... b1b83a467bb6d	2022-03-09, 21:54:46	3	default_pool	1	10a2e44-1311-4253-99b6-17661ead848	
  	 etl_athena_job	athena_ctas_new_table	Y	scheduled_2022-03-09T21:00:00+00:00	2022-03-09, 21:00:00	AWSAthenaOperator	2022-03-09, 22:03:42	2022-03-09, 22:04:14	0:30:31.457272	17	ip-10-192-21-66.ap-northeast-1.compute.internal	airflow	1	airflow celery 178319b... fe3d-487a-... b37c-... b1b83a467bb6d	2022-03-09, 22:03:42	3	default_pool	2	b6a2185c-e5af-4857-9b62-330ba6a0d373	

まとめ

MLOps 実装の流れと本日触れた内容

実装準備ステップ

要件を明確にする

理想形と現状の差分を確認し、
ロードマップを描く

ツールの選定と設計を行う

本日の内容

- 以下の観点で確認するべきポイント
 - データ
 - モデル
 - インフラ
- ロードマップを描く際の考慮点
 - スキル、既存資産
 - ステップアップ例
- AWSで実装する際の代表的なツール（パイプラインとデータ処理を中心に）
 - メリット/考慮点
 - アーキテクチャ例
 - 実装時の参考情報

参考情報

- Machine Learning Lens
 - 機械学習のAWS Well-Architected Framework
 - <https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/machine-learning-lens.html>
- MLOps: Continuous Delivery for Machine Learning on AWS
 - MLOpsのWhitepaper
 - <https://d1.awsstatic.com/whitepapers/mlops-continuous-delivery-machine-learning-on-aws.pdf>
- Building, automating, managing, and scaling ML workflows using Amazon SageMaker Pipelines
 - SageMaker Pipelinesの使い方解説
 - <https://aws.amazon.com/jp/blogs/machine-learning/building-automating-managing-and-scaling-ml-workflows-using-amazon-sagemaker-pipelines/>

Thank you!



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.