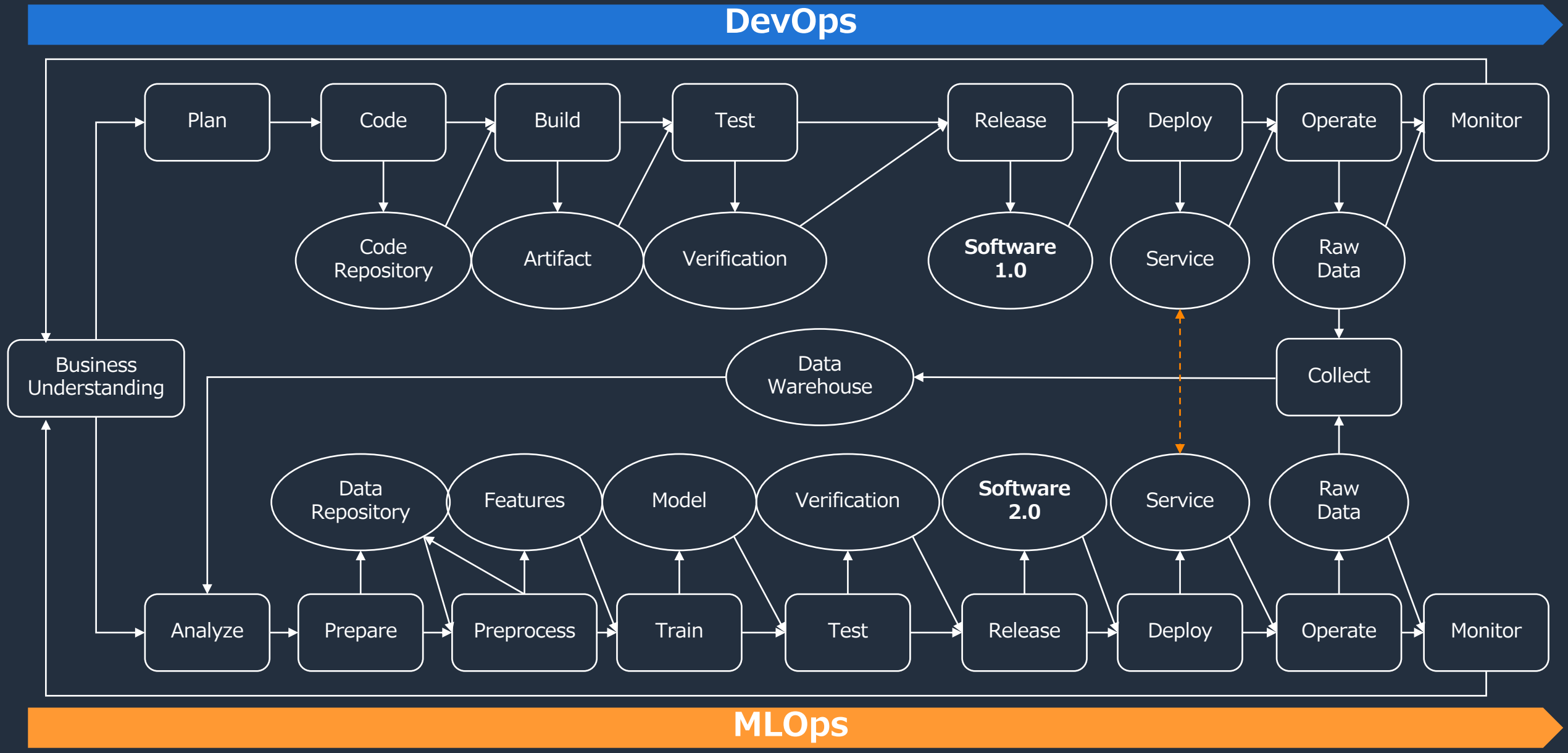
ML Enablement Series 【ML-Dark-02】

Amazon SageMaker による実験管理

機械学習ソリューションアーキテクト
伊藤 芳幸

DevOps & MLOps フロー



DevOps & MLOps を実現するロールマップ

Architect ソフトウェア開発に必要なソフトウェアアーキテクチャ全体を設計する。

DevOps Engineer ソフトウェアの開発・運用プロセスを自動化する。

Product Manager
実装すべきソフトウェア機能を定義する。

Software Engineer
ソフトウェアの開発を行う。

Operator

System Admin

機械学習のモデル構築にあたって、
試行錯誤の再現と振り返りのための
「実験管理」について機能を紹介

Business Analyst
解決すべきビジネス上の問題を定義する。

Data Analyst
データの可視化と分析で問題を定量的に特定する。

IT Auditor

Data architect データを管理する基

Domain Expert
あるべき挙動をデータを用いて定義する。
評価尺度を定義する。

Data Engineer
機械学習モデルに入力可能なデータと特徴を作成する。

Data Scientist
機械学習モデルを構築する

ML Engineer
機械学習モデルを本番環境にデプロイ可能な形式に変換する。

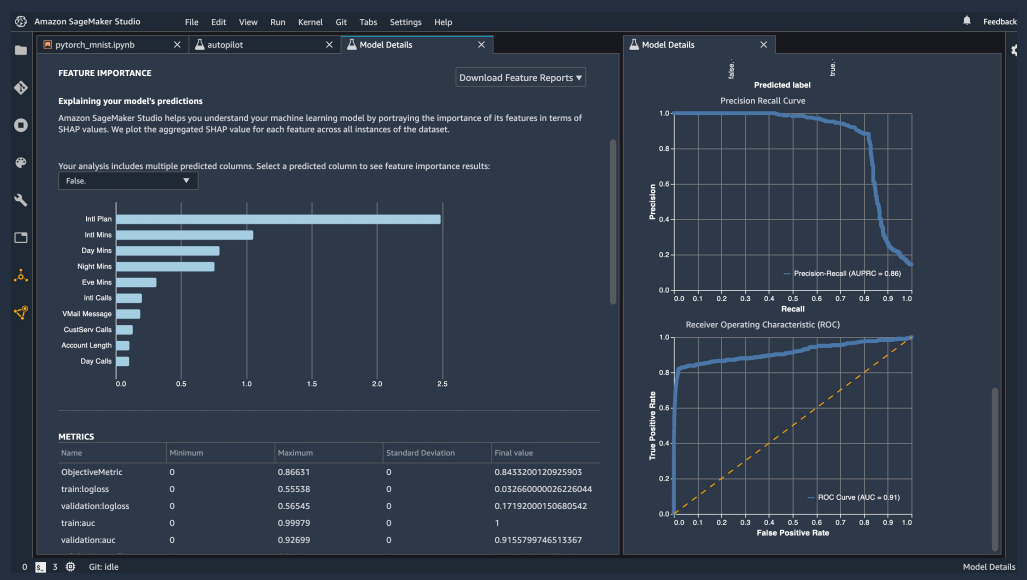
ML Operator
推論結果に基づき業務を行いつつ、推論結果にフィードバックを与える。

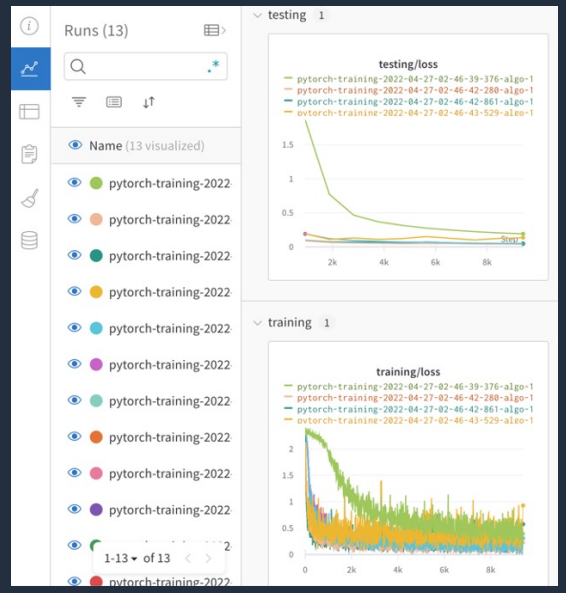
Model risk Manager
機械学習のサービスの挙動を監視する。

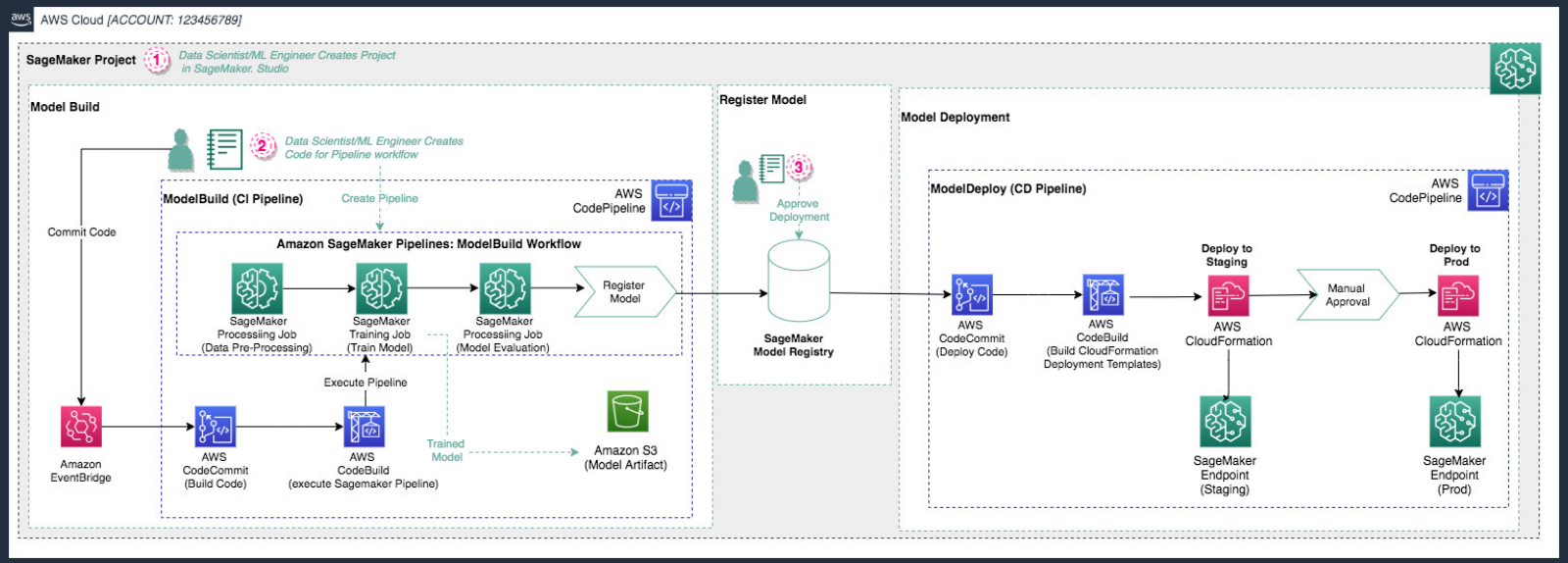
MLOps Engineer 機械学習モデルの開発・運用プロセスを自動化する。

AI/ML Architect 機械学習に必要なアーキテクチャ全体を設計する。

この動画の対象者と得られること







データサイエンティスト向け

- 実験管理の必要性おさらい
- SageMakerでできる実験管理
- SageMakerと使い慣れた実験管理ツールとの連携

MLOpsエンジニア向け

- 実験管理の必要性おさらい
- パイプラインによるガバナンスの効いた実験管理の仕組み

この動画で解説すること3つ

1. なぜ実験管理が必要なのか
2. 再現性のためのソリューション
3. 振り返りのためのソリューション

補足：ソリューション構築手順紹介

なぜ実験管理が必要なのか

モデル開発には試行錯誤が伴う

データ観察

評価基準構築
（ベースモデル構築）

モデル高度化
（特徴量生成・他モデル）

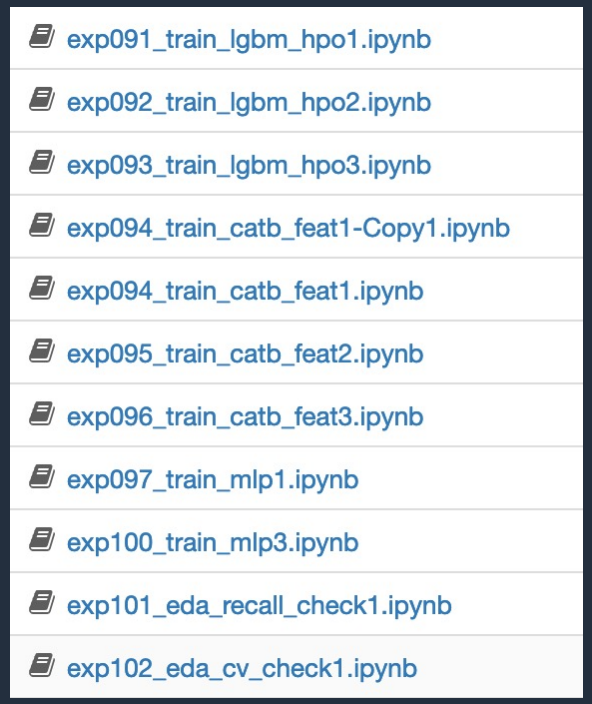
パラメータチューニング

- データを変えてみる
- マシンを変えてみる
- ライブラリを変えてみる
- 前処理ロジックを変えてみる
- 評価指標を変えてみる
- CVの方法を変えてみる
- 特徴量を変えてみる
- アルゴリズムを変えてみる
- ハイパーパラメータを変えてみる
- 複数の実験結果を比較する

数百の実験をすることも！

ついやってしまう管理（管理ではない）

ローカルPCにノートブック形式で保存



✖ データ保存

・PCが壊れたり、担当データサイエンティストが退職して全てを失うケースも…

✖ 振り返り

・コード差分がわかりずらく、実験の把握に時間がかかる
・セル出力を保存していない場合、ノートブックを再実行する必要がある

✖ 再現性

・実行順に記載されておらず、エラーが発生
・異なる環境でエラーが発生

ローカルPCに表計算ソフトで記録

	A	B	C
1	notebook	Cvscore	TESTscore
2	exp035	0.66421478	0.12398777
3	exp037	0.4996275	0.40424265
4	exp040	0.58709581	0.36761254
5	exp041	0.88862392	0.96609412
6	exp042	0.73541102	0.36604913
7	exp043	0.76872504	0.46775784
8	exp044	0.81449647	0.7869291
9	exp045	0.74482335	0.01155465
10	exp048	0.97364383	0.54052217
11	exp049	0.54851112	0.79621083

✖ データ保存

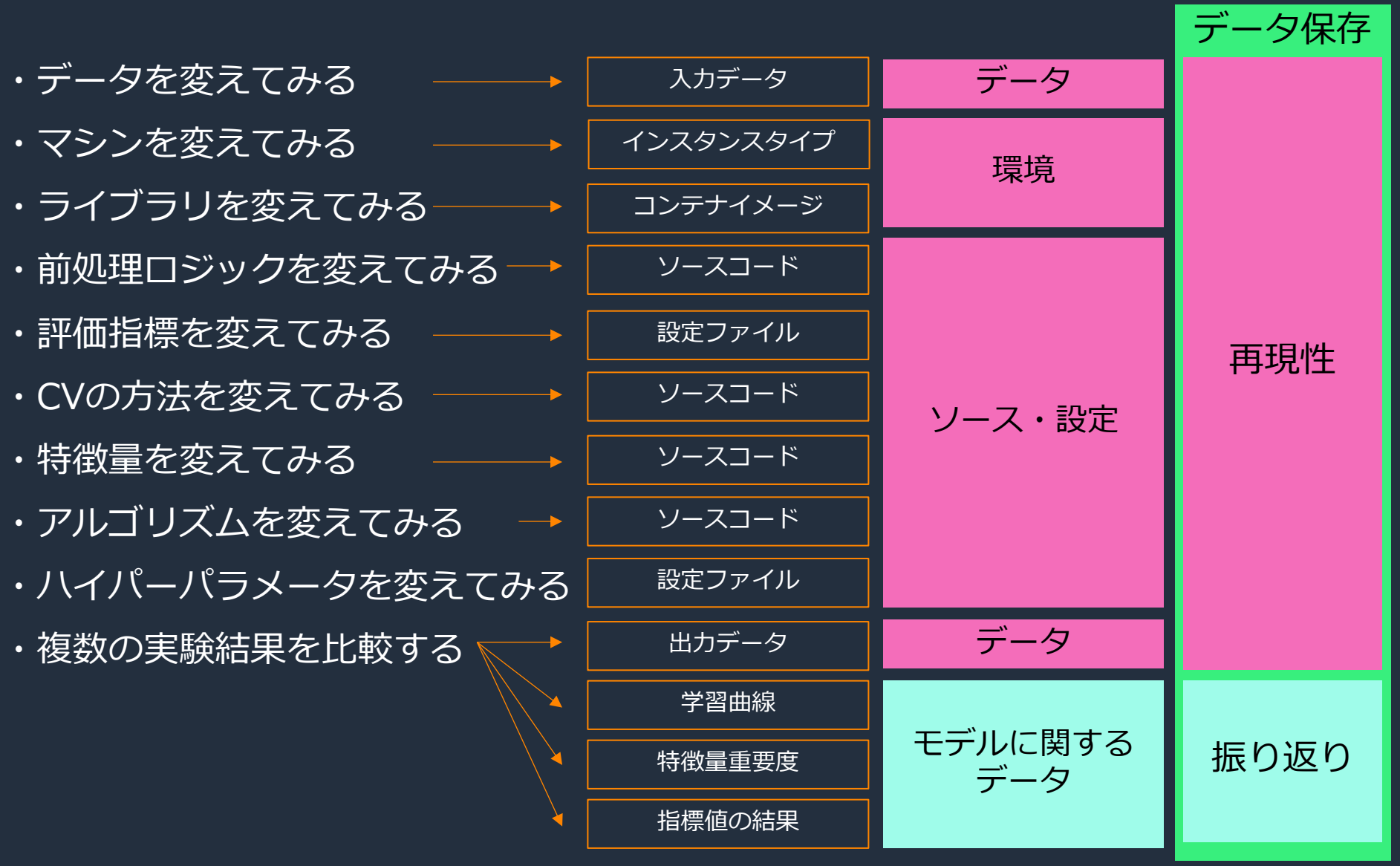
・PCが壊れたり、担当データサイエンティストが退職して全てを失うケースも…

✖ 振り返り

・記録の抜け漏れがある
・属人的（人の善意が頼り）

属人性が高く、組織として開発するには向かない仕組み

再現性と振り返りのために保存すべきデータ

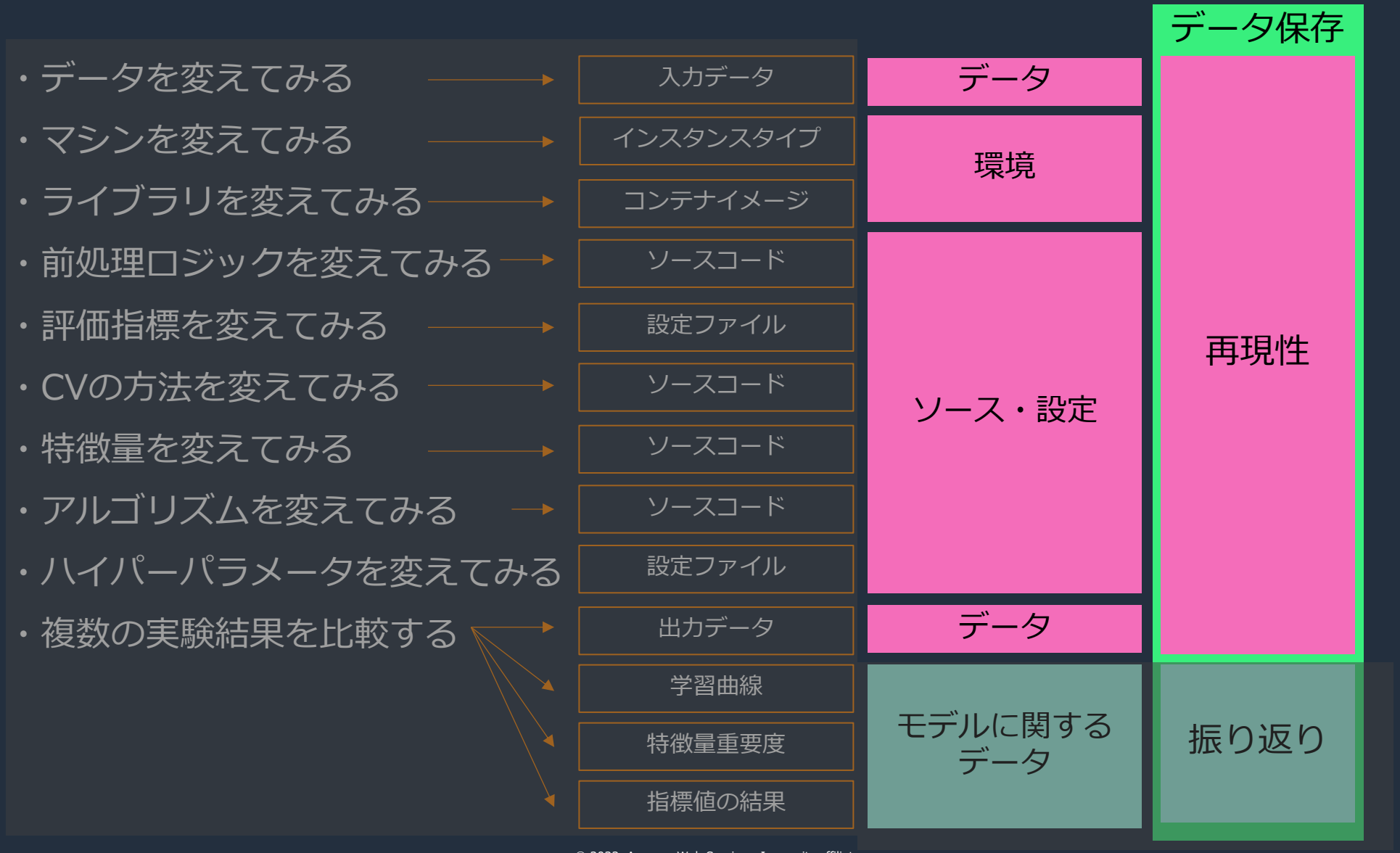


ロールごとの、実験管理に求めること

	データサイエンティスト 管理は最低限にして、多く実験がしたい。勝手に記録してくれて比較しやすい機能があると嬉しい。	MLOpsエンジニア 開発されたモデルがスムーズにプロダクトに組み込める仕組みを作りたい。モデル構築にガバナンスを効かせたい。
データ保存	・実験記録が消失しないこと ・自動で保存されること	・実験記録が消失しないこと ・自動で保存されること
振り返り	比較しやすい実験管理ツールが利用できること	データサイエンティストが効率的に振り返りができること
再現性	他メンバや過去の自分の実験が再現できること	データサイエンティストがいなくても実験を再現できること

再現性のためのソリューション

再現性と振り返りのために保存すべきデータ

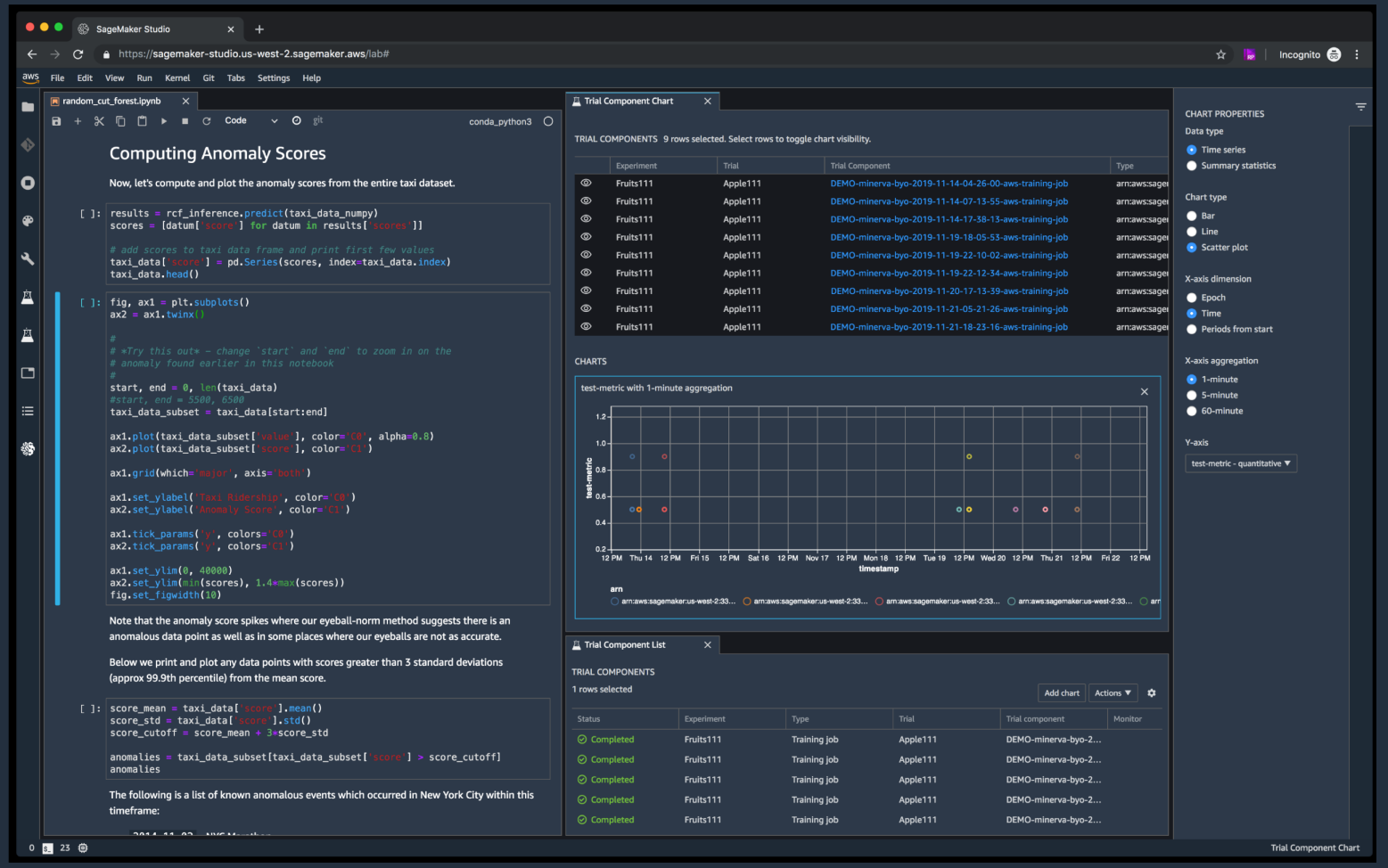


とはいえ、ノートブックの柔軟性は捨てがたい…

クラウドで実施しましょう

統合開発環境（IDE）：Amazon SageMaker Studio

Web ブラウザから利用可能なクラウドの Jupyter notebook 環境



データ

Jupyter上のファイルはユーザー毎のEFSに保管されている

環境

記録されない。実行インスタンスタイプやコンテナはノートブックなどに記載しておく必要あり

ソース・設定

ノートブックとして記録されているのみ

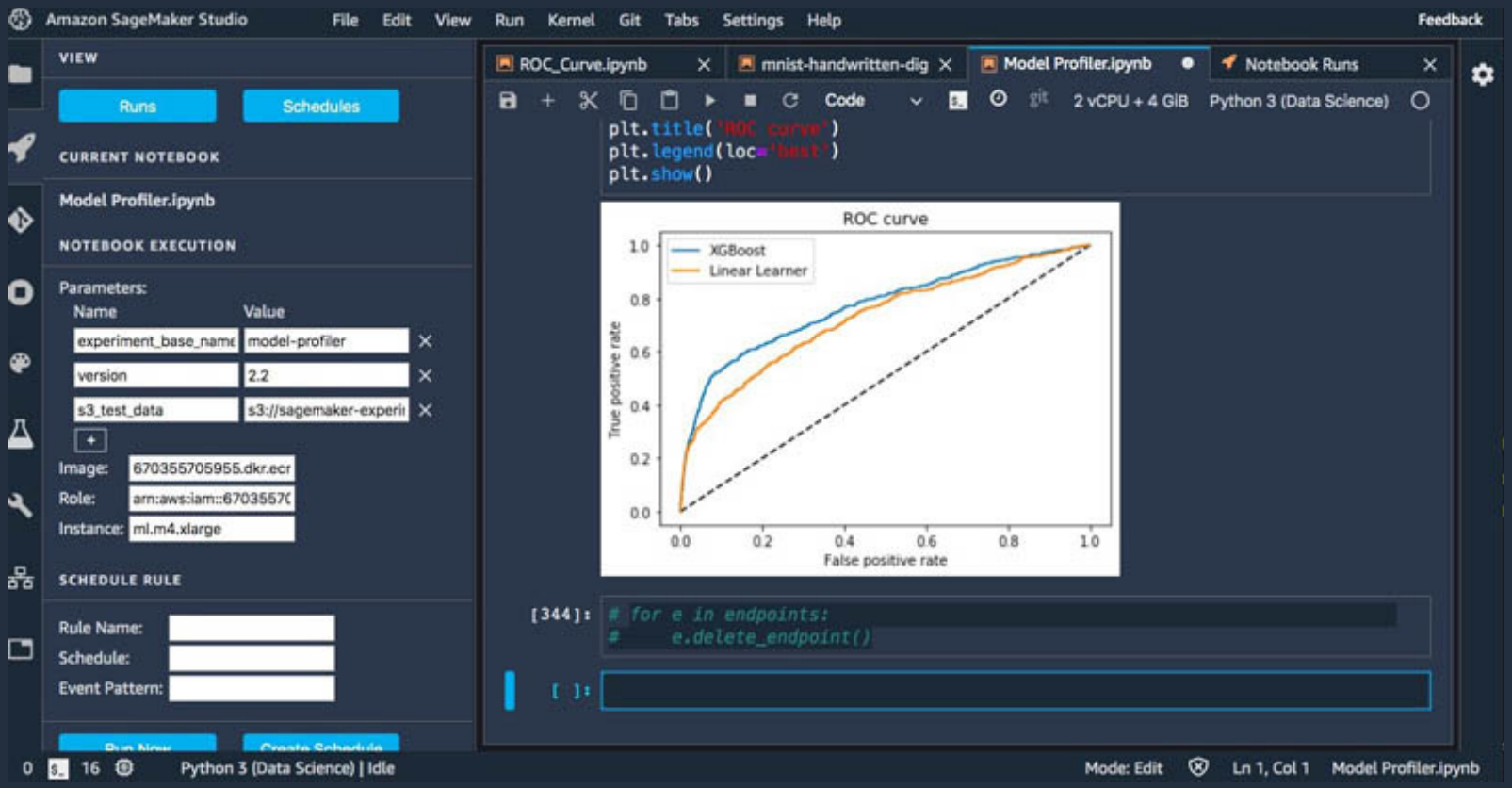
振り返り

ノートブックを開いて、セル出力結果を都度確認する

SageMaker-run-notebook

使い方は動画後半で

Papermill で Jupyter ノートブックのままバッチ実行が可能



データ

事前にS3に保存する。ノートブックではS3のファイルをロードする。

環境

実行インスタンスタイプとコンテナURLが記録される

ソース・設定

実行前ノートブックと、実行後ノートブックがS3に保存される

振り返り

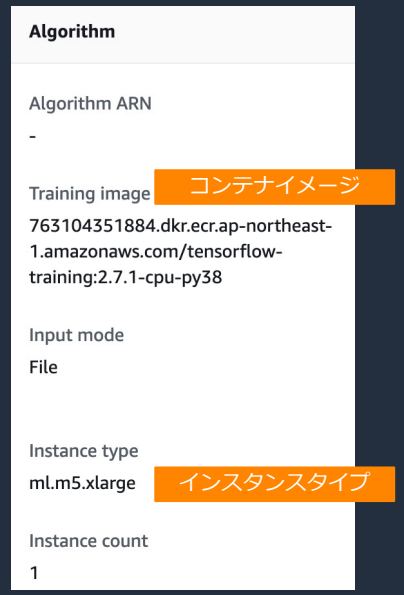
実行後ノートブックを確認

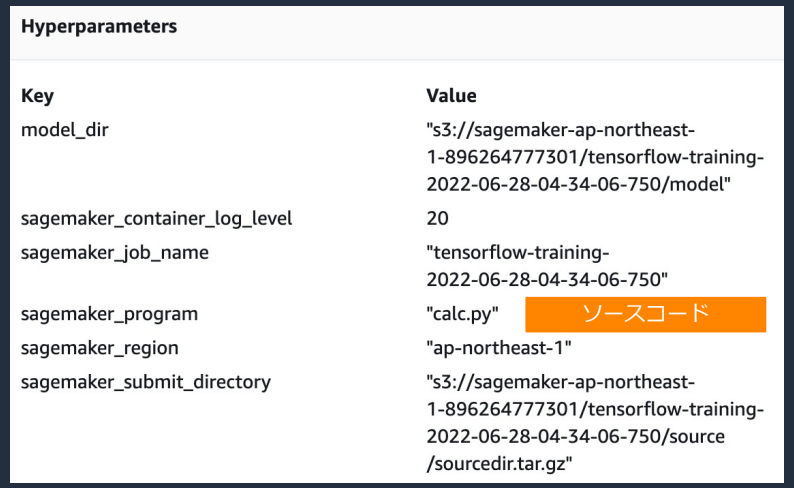
AWSブログ：https://aws.amazon.com/jp/blogs/news/scheduling-jupyter-notebooks-on-sagemaker-ephemeral-instances/
GitHub：https://github.com/aws-samples/sagemaker-run-notebook

SageMaker ジョブをノートブックから実行

再現に必要なデータを自動で記録

```
# トレーニングジョブの実行
from sagemaker.tensorflow import TensorFlow
estimator = TensorFlow(
    entry_point='./src/1-2-1/calc.py',
    py_version='py38',
    framework_version='2.7.1',
    instance_count=1,
    instance_type='ml.m5.xlarge',
    role=sagemaker.get_execution_role()
)
estimator.fit(input_s3_uri)
```





その他 入力データ 出力データ など も記録

データ

- S3に事前に配置する
- 実行ではS3の場所を指定

環境

コンテナイメージの場所、インスタンスタイプが記録される

ソース・設定

- 利用したソースコードをS3に保存
- 入力パラメータが記録される

振り返り

- マネジメントコンソールで記録を確認
- metric_definitionsで指標値を記録

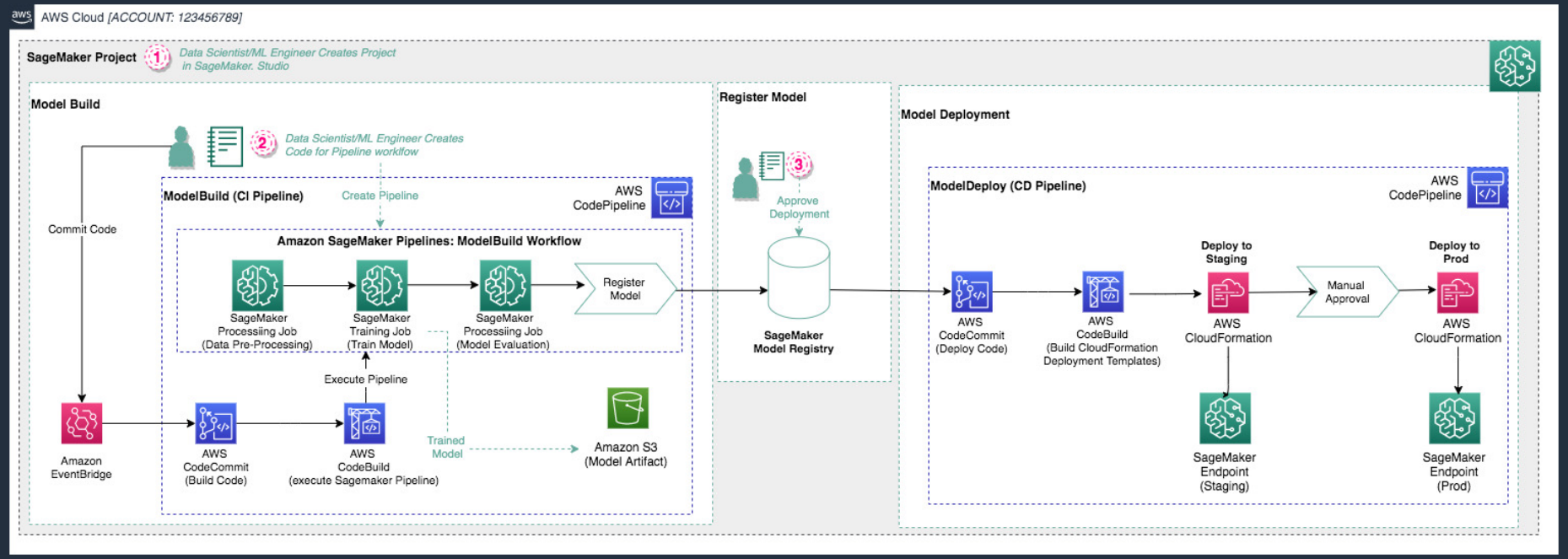
(SageMaker 学習ジョブ)

https://github.com/aws-samples/aws-ml-jp/blob/main/sagemaker/sagemaker-traning/tutorial/1_hello_sagemaker_training.ipynb

SageMaker Pipelines

使い方は動画後半で

SageMaker Studio から数クリックで構築できるパイプライン



データ

SageMakerジョブと同等

環境

SageMakerジョブと同等

ソース・設定

リポジトリに保存される

振り返り

SageMakerジョブと同等

・MLOpsエンジニアはパイプラインを構築・管理する

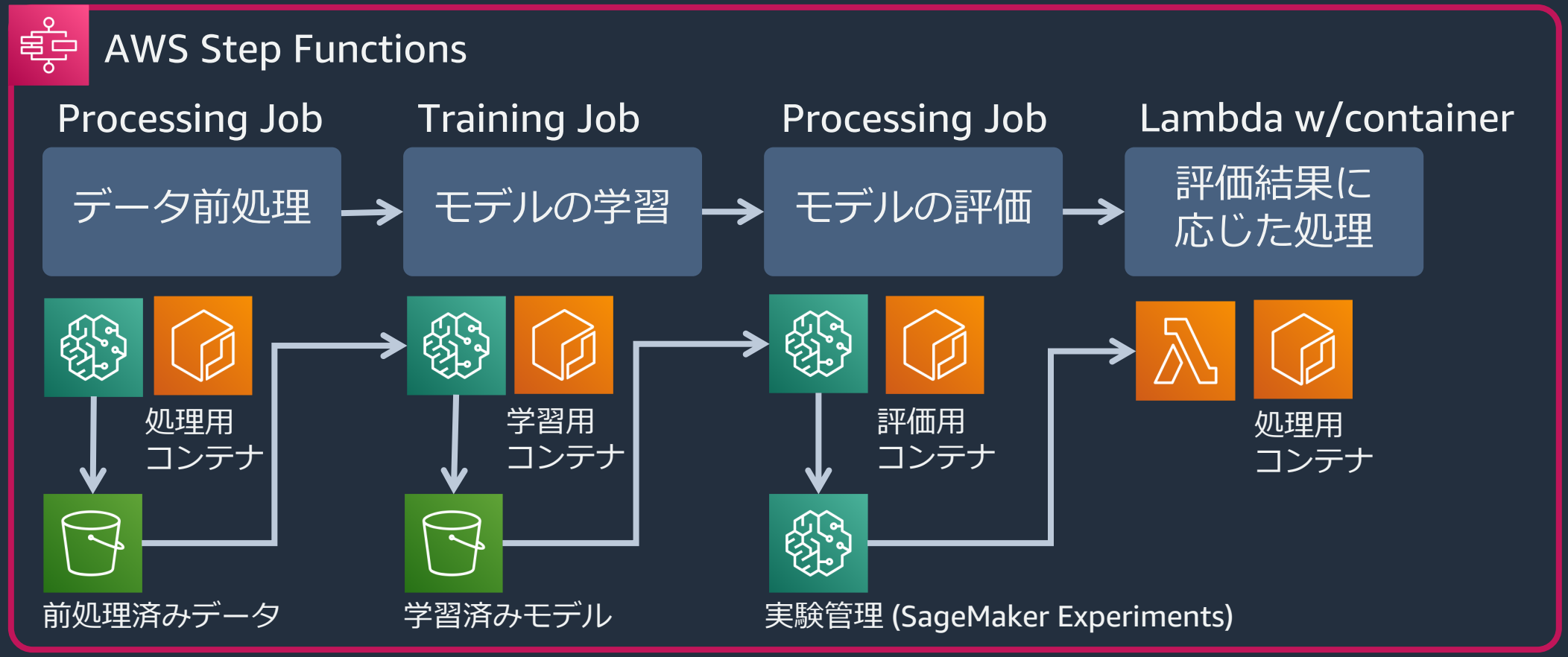
・成果物をリポジトリに提出するルールとすることで、ガバナンスを確立

・データサイエンティストはコーディング、リポジトリへのpushを担当し、モデル構築に注力する

https://aws.amazon.com/blogs/machine-learning/building-automating-managing-and-scaling-ml-workflows-using-amazon-sagemaker-pipelines/

AWS StepFunctions パイプライン

共通部分をパイプラインにして自動化し、
使用するデータや学習スクリプトを変えながらより良いモデルを効率的に探索



https://aws.amazon.com/jp/builders-flash/202111/nyantech-ml-ops/?awsf.filter-name=*all

再現性のためのソリューションまとめ

・自社のユースケースや組織、どこまで再現性を要求するかをもとにソリューションを選択する
・SageMakerジョブで前処理や学習を行なっておけば、高い再現性を確保できる

ソリューション	再現性	データサイエンティストの学習コスト	MLOpsエンジニアが実施すること
ローカルノートブック	DS退職やPC壊れたら終了	特になし	特になし
SageMaker Studio	ノートブックをEFSに保存	特になし	SageMaker Studio ドメイン構築
SageMaker-run-notebook	ノートブック・実行環境をS3に保存（実行前・後）	S3でのデータの扱い方	Run-notebookコンテナ構築
SageMakerジョブ（ノートブックから）	データ・環境・ソースコードを保存（S3）	（上記に加え） SageMaker SDKの使い方	特になし
パイプライン (SageMaker Pipelines / AWS StepFunctions など)	ソースコードをリポジトリ(CodeCommit / GitHubなど) に保存	（上記に加え） ・リポジトリの使い方 ・パイプラインの理解	パイプライン構築

振り返りのためのソリューション

実験の振り返り、比較に手間がかかってしまう

ローカルPCにノートブック形式で保存

- exp091_train_lgbm_hpo1.ipynb
- exp092_train_lgbm_hpo2.ipynb
- exp093_train_lgbm_hpo3.ipynb
- exp094_train_catb_feat1-Copy1.ipynb
- exp094_train_catb_feat1.ipynb
- exp095_train_catb_feat2.ipynb
- exp096_train_catb_feat3.ipynb
- exp097_train_mlp1.ipynb
- exp100_train_mlp3.ipynb
- exp101_eda_recall_check1.ipynb
- exp102_eda_cv_check1.ipynb

✖ データ保存

・PCが壊れたり、担当データサイエンティストが退職して全てを失うケースも…

✖ 振り返り

・コード差分がわかりずらく、実験の把握に時間がかかる
・セル出力を保存していない場合、ノートブックを再実行する必要がある

✖ 再現性

・実行順に記載されておらず、エラーが発生
・異なる環境でエラーが発生

ひとつずつ開いて確認

ローカルPCに表計算ソフトで記録

	A	B	C
1	notebook	Cvscore	TESTscore
2	exp035	0.66421478	0.12398777
3	exp037	0.4996275	0.40424265
4	exp040	0.58709581	0.36761254
5	exp041	0.88862392	0.96609412
6	exp042	0.73541102	0.36604913
7	exp043	0.76872504	0.46775784
8	exp044	0.81449647	0.7869291
9	exp045	0.74482335	0.01155465
10	exp048	0.97364383	0.54052217
11	exp049	0.54851112	0.79621083

✖ データ保存

・PCが壊れたり、担当データサイエンティストが退職して全てを失うケースも…

✖ 振り返り

・記録の抜け漏れがある
・属人的（人の善意が頼り）

記録に手間がかかる

ダッシュボードで振り返り、パイプラインで再現性担保

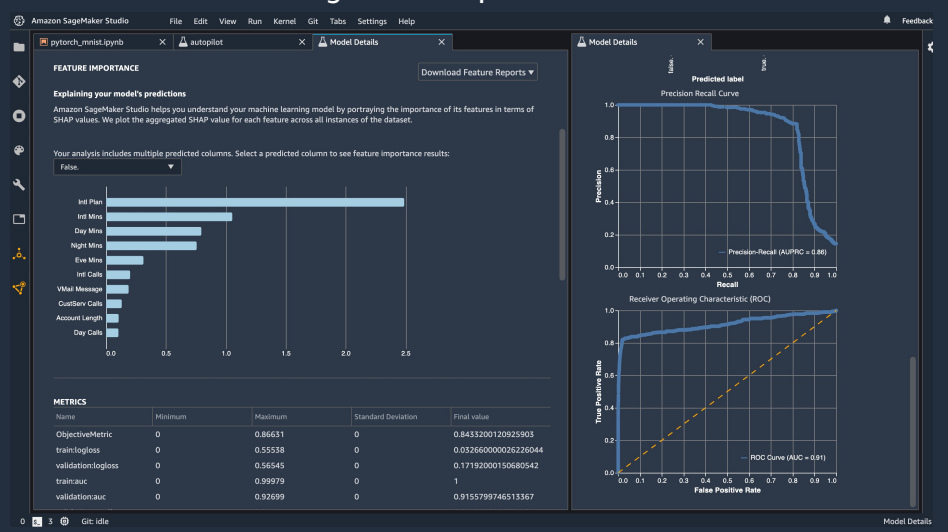
ひとつずつ開いて確認

比較しやすいダッシュボード

- 学習曲線
- 特徴量重要度
- 指標値の結果

→ 振り返り

SageMaker Experiments



サードパーティツール

記録に手間がかかる

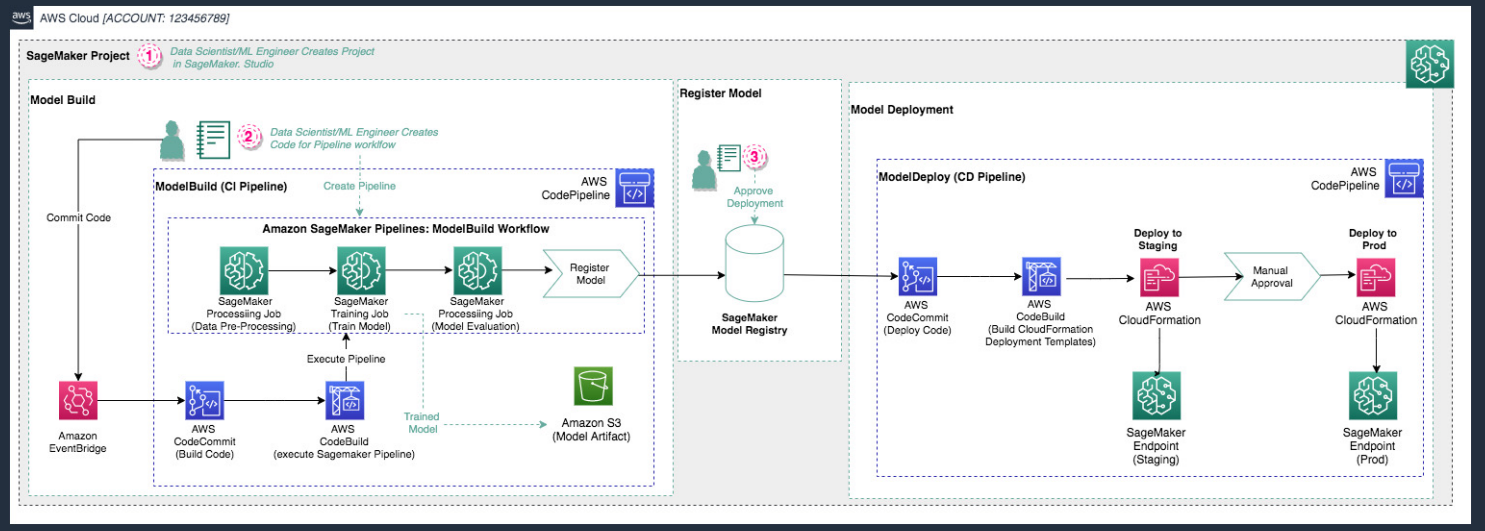
パイプラインで再現性確保と指標値の自動記録

- データ
- 環境
- ロジック

→ 再現性

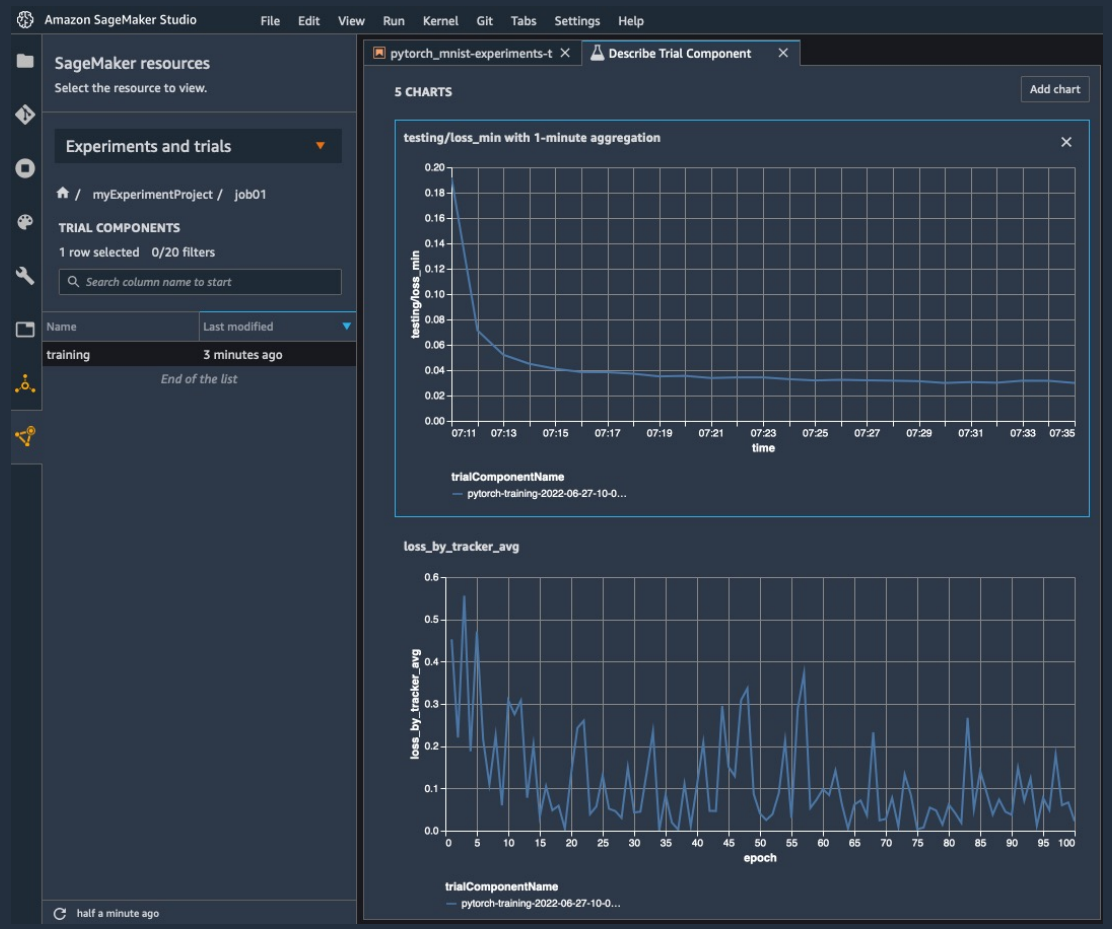
- 指標値の結果

→ 振り返り



SageMaker Experiments で比較する

例：SageMaker Experiments



ジョブ発行前に記載

```
from smexperiments import experiment, tracker

my_experiment = experiment.Experiment.create(experiment_name='myExperimentProject')
my_trial = my_experiment.create_trial(trial_name='job01')

with tracker.Tracker.create(display_name='training') as my_tracker:
    my_tracker.log_input(name="input-dataset-dir", media_type="s3/uri", value=inputs)

estimator.fit({'training': inputs},
    experiment_config={
        "TrialName": my_trial.trial_name,
        "TrialComponentDisplayName": my_tracker.trial_component.display_name,
    })
```

ソースコードに記載

```
from smexperiments import tracker
# load tracker from already existing trial component
my_tracker = tracker.Tracker.load()
 for epoch in range(1, args.epochs + 1):
        model.train()
        for batch_idx, (data, target) in enumerate(train_loader, 1):


        # epochの終わりにlossを記録
        my_tracker.log_metric(metric_name='loss_by_tracker', value=loss.item(), iteration_number=epoch)
```

https://github.com/aws/amazon-sagemaker-examples/blob/main/sagemaker-experiments/mnist-handwritten-digits-classification-experiment/mnist-handwritten-digits-classification-experiment.ipynb

サードパーティツールで比較する

例：Weights & Biases

ジョブ発行前に記載

```
import wandb
wandb.login()
settings = wandb.setup().settings
current_api_key =
wandb.wandb_lib.apikey.api_key(settings=settings)
```

```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point='mnist.py',
                    source_dir="src",
                    role=role,
                    py_version='py3',
                    framework_version='1.8.0',
                    instance_count=1,
                    instance_type='ml.c5.2xlarge',
                    hyperparameters={
                        'epochs': 1,
                        'backend': 'gloo'
                    },
                   environment={"WANDB_API_KEY": current_api_key})
```

ソースコードに記載

```
import wandb

wandb.init(project="sm-pytorch-mnist-studio",
config=vars(args))

wandb.watch(model)

wandb.log({"training/loss": loss.item()})
```

https://github.com/wandb/examples/tree/master/examples/pytorch/pytorch-cifar10-sagemaker

SageMakerは多くのサードパーティツールと連携可能

Weights & Biases
https://docs.wandb.ai/guides/integrations/other/sagemaker

MLflow Tracking
https://aws.amazon.com/jp/blogs/news/machine-learning-managing-your-machine-learning-lifecycle-with-mlflow-and-amazon-sagemaker/

Neptune.ai
https://docs.neptune.ai/integrations-and-supported-tools/ide-and-notebooks/amazon-sagemaker

Comet.ml
https://www.comet.ml/site/building-reliable-machine-learning-pipelines-with-aws-sagemaker-and-comet-ml/

まとめ

・組織で機械学習モデルの開発をするには、実験の再現性を確保することが重要。

・SageMaker ジョブを利用すれば、実験の再現に必要な情報は自動で記録することが可能。パイプラインを構築することで、ソースコードのリポジトリ管理や再現性確保のための情報の自動記録など、さらにガバナンスの効いた仕組みを構築できる。

・SageMaker はサードパーティの実験管理ツールの多くと連携でき、データサイエンティストが使いやすいツールを利用して振り返りを行うこともできる。

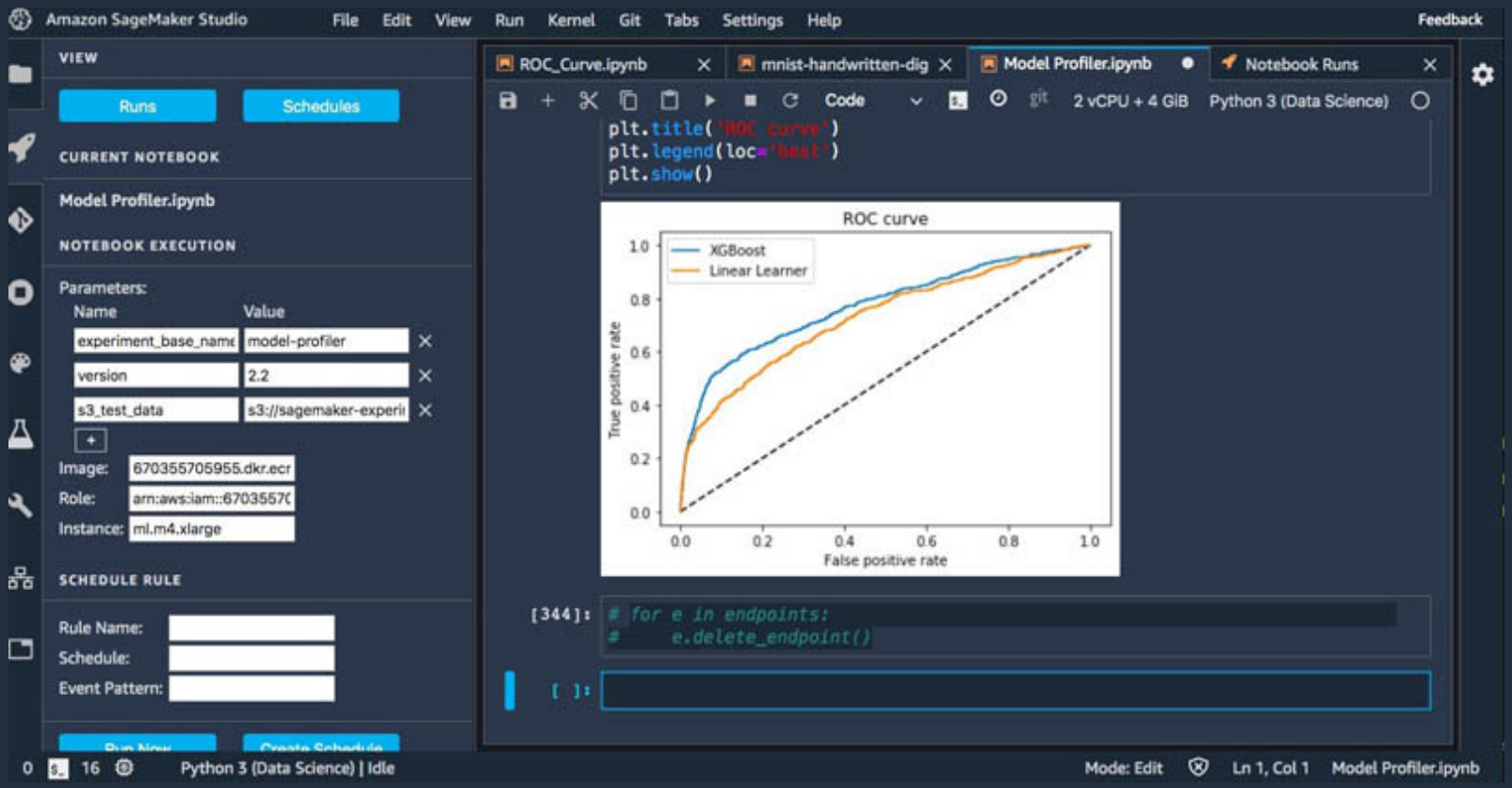
補足：ソリューション構築

Run-notebook
SageMaker Pipelines

SageMaker-run-notebook

使い方は動画後半で

Papermill で Jupyter ノートブックのままバッチ実行が可能



データ

事前にS3に保存する。ノートブックではS3のファイルをロードする。

環境

実行インスタンスタイプとコンテナURLが記録される

ソース・設定

実行前ノートブックと、実行後ノートブックがS3に保存される

振り返り

実行後ノートブックを確認

AWSブログ：https://aws.amazon.com/jp/blogs/news/scheduling-jupyter-notebooks-on-sagemaker-ephemeral-instances/
GitHub：https://github.com/aws-samples/sagemaker-run-notebook

ノートブックのまま実行する環境を構築する

SageMaker Studio システムコンソールなどで以下のコマンドを実行

コマンド1

```
pip install https://github.com/aws-samples/sagemaker-run-notebook/releases/download/v0.20.0/sagemaker_run_notebook-0.20.0.tar.gz
```

run-notebook コマンドのインストール

コマンド2

```
run-notebook create-infrastructure
```

AWS CloudFormationで、AWS Lambda や IAM role が作成される

コマンド3

```
run-notebook create-container
```

AWS CodeBuild が実行され、ノートブックを SageMaker Processing で利用する コンテナ (Papermill 入り) が構築される

コマンド4

```
bash install-run-notebook.sh
```

install-run-noteook.sh の内容（自分で作成する）

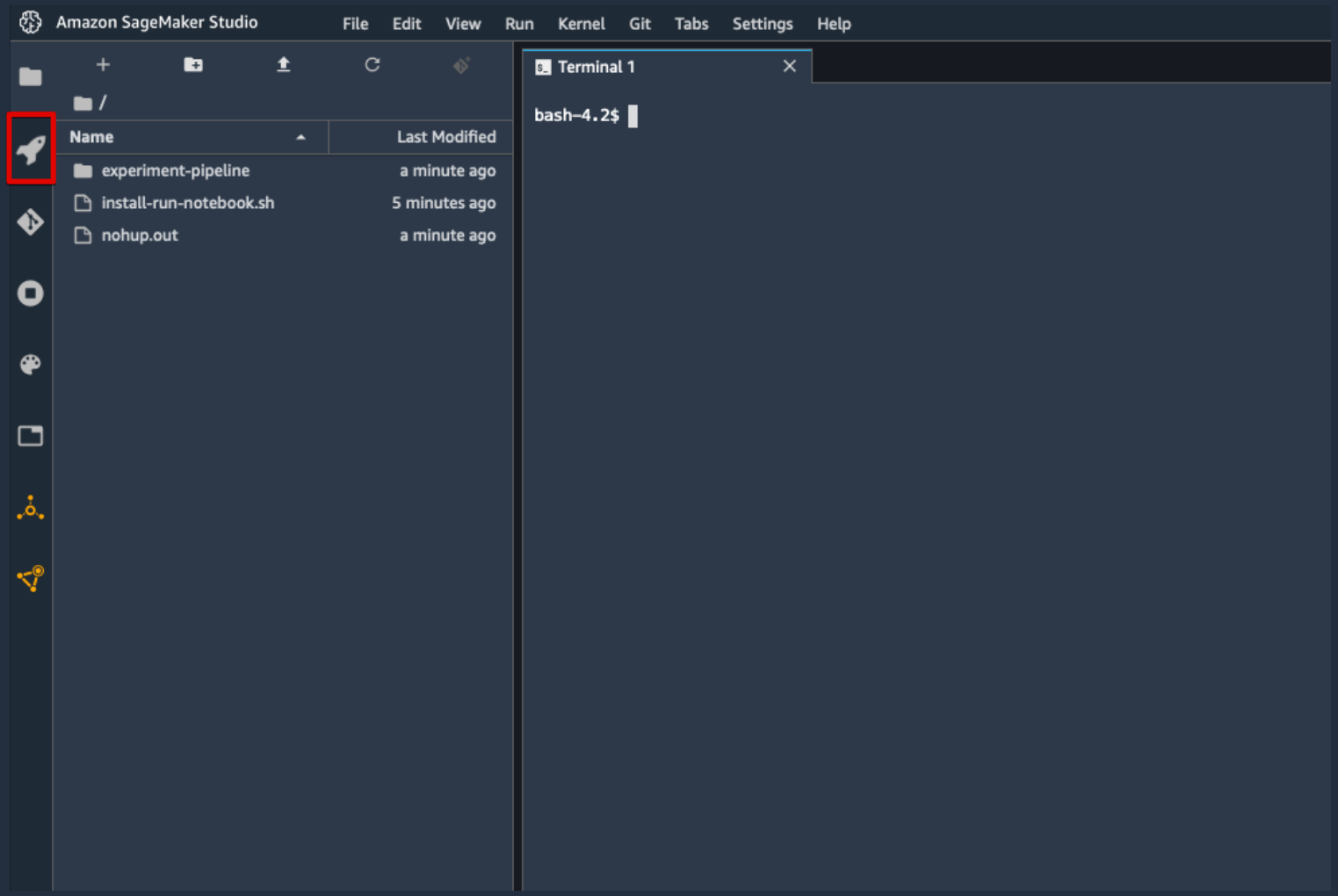
```
version=0.18.0
pip install https://github.com/aws-samples/sagemaker-run-notebook/releases/download/v${version}/sagemaker_run_notebook-${version}.tar.gz
jlpm config set cache-folder /tmp/yarncache
jupyter lab build --debug --minimize=False
nohup supervisorctl -c /etc/supervisor/conf.d/supervisord.conf restart jupyterlabserver
```

https://github.com/aws-samples/sagemaker-run-notebook/blob/master/QuickStart.md

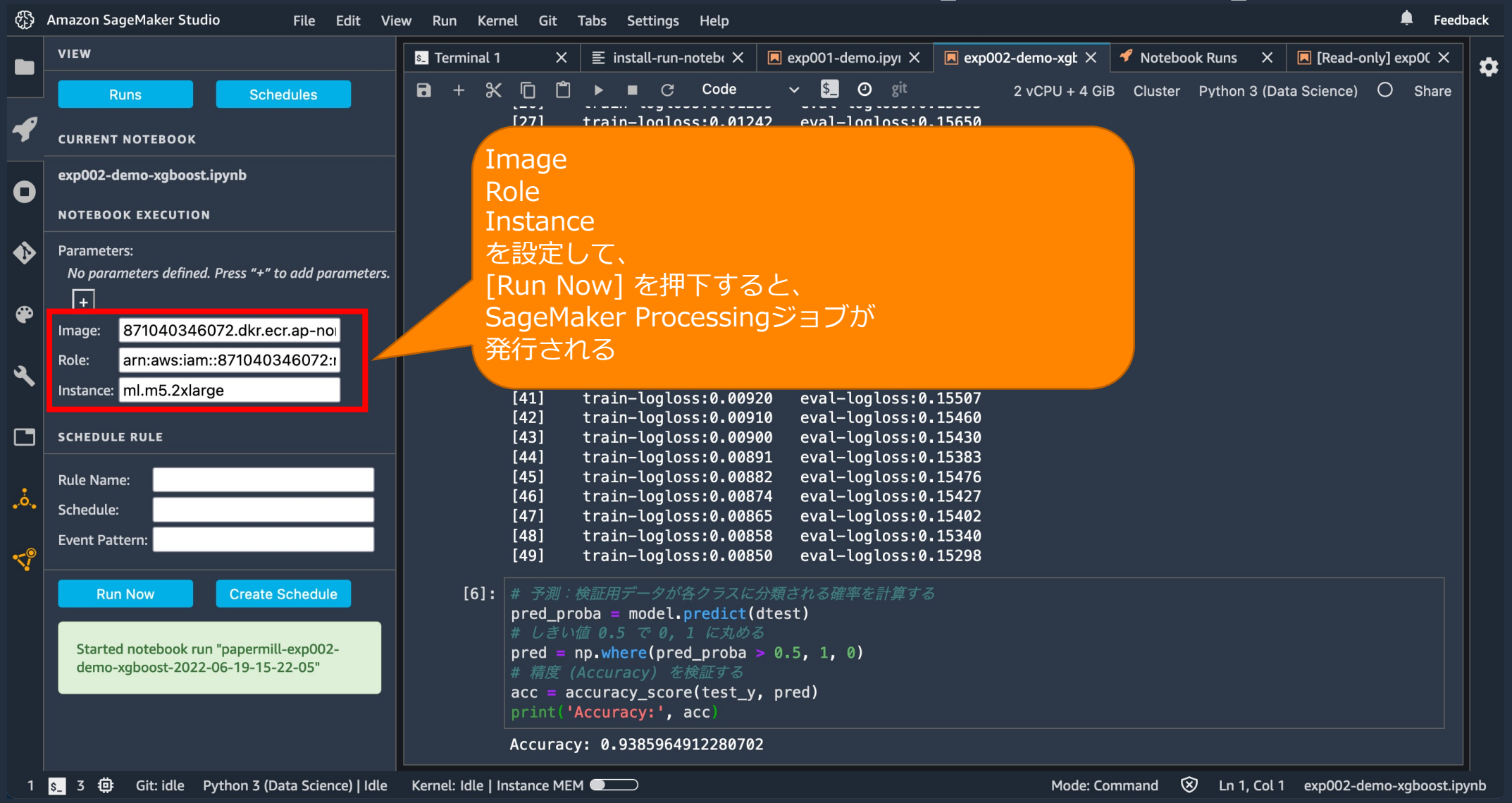
事前に付与しておく権限

Policy name
IAMFullAccess
AmazonS3FullAccess
AWSCodeBuildAdminAccess
AmazonSageMakerFullAccess
AWSCloudFormationFullAccess
AWSLambda_FullAccess

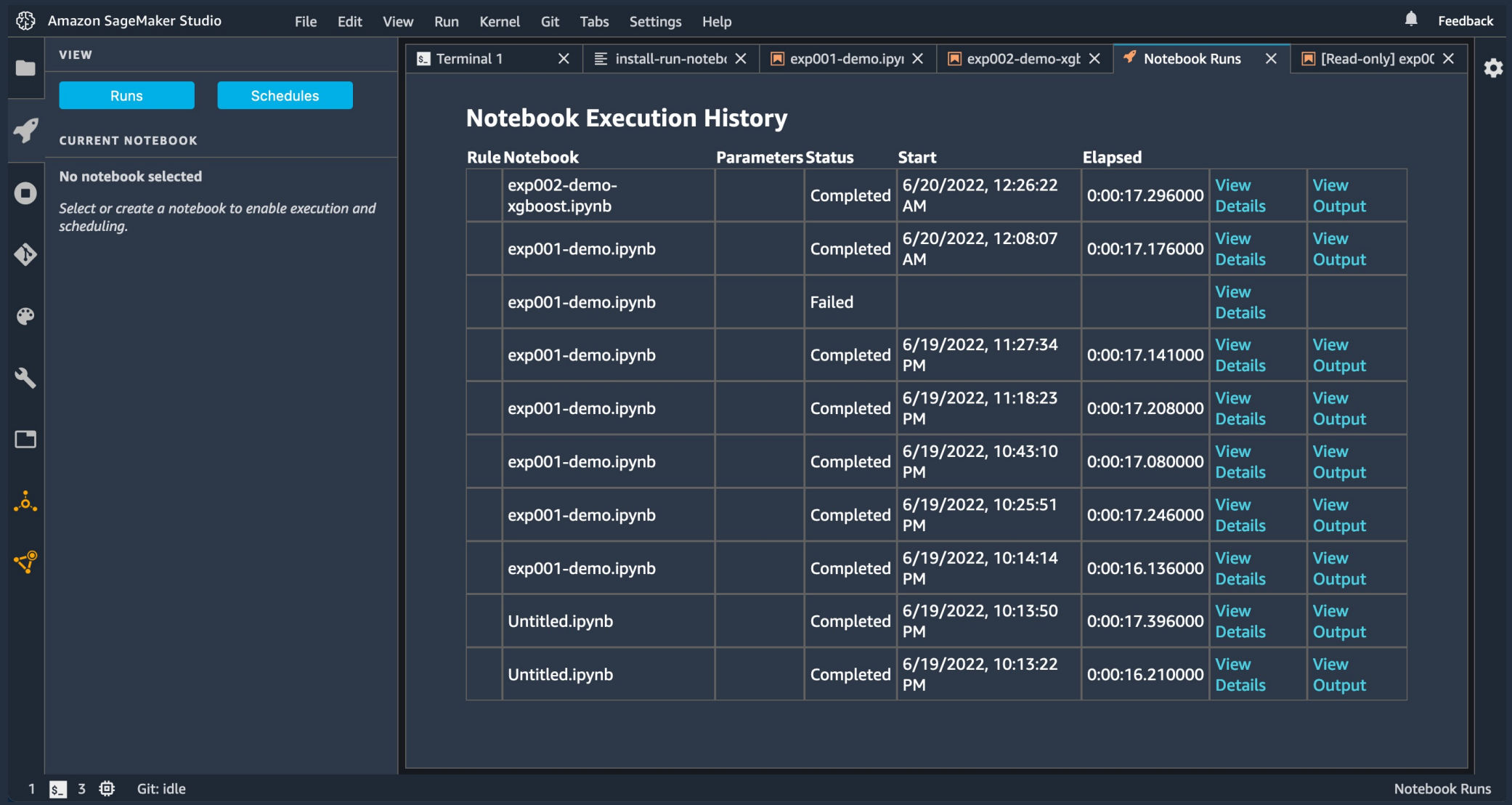
ブラウザ更新で、拡張機能が表示される



ノートブックを選択した状態で、[Run Now]で実行

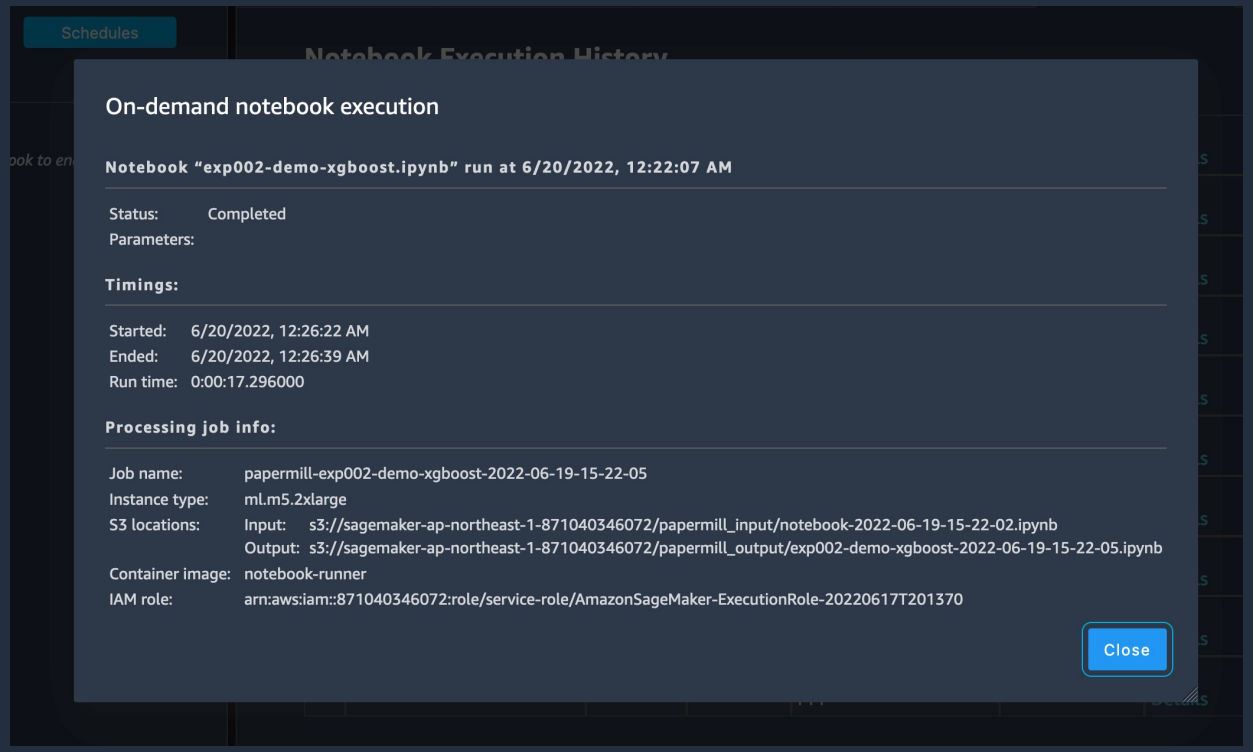


実行ジョブは一覧で確認することができる

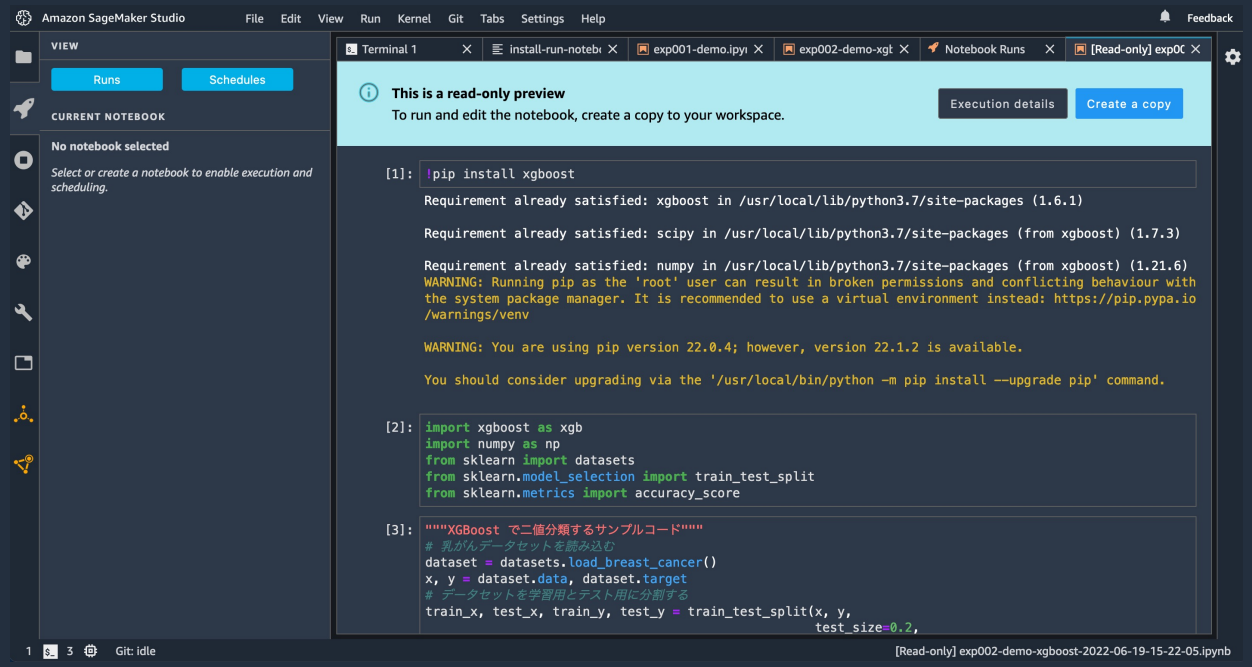


実行条件と、実行結果の確認

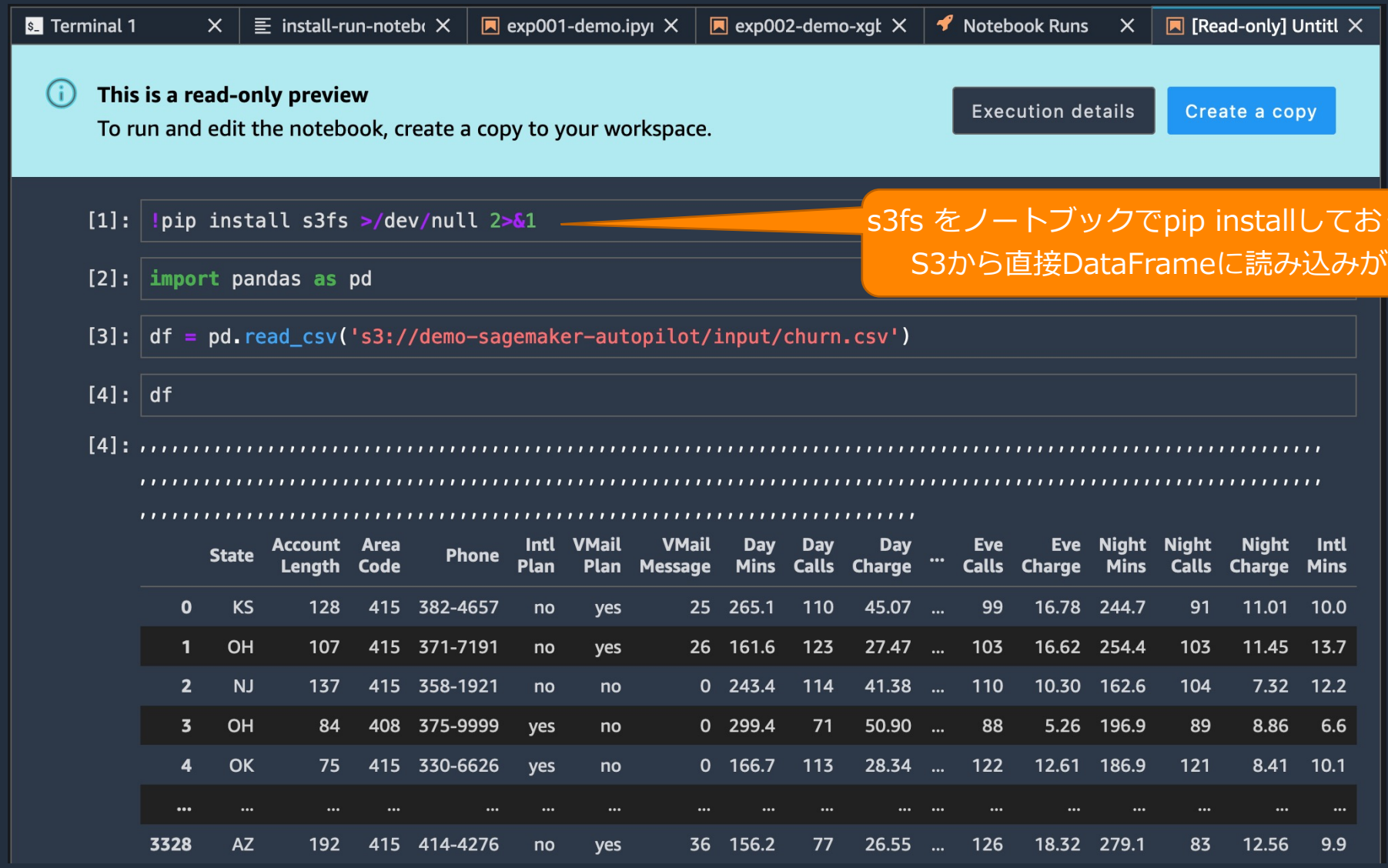
実行条件



実行結果（ノートブックのセル出力）



Tips: S3から直接DataFrameに読み込み



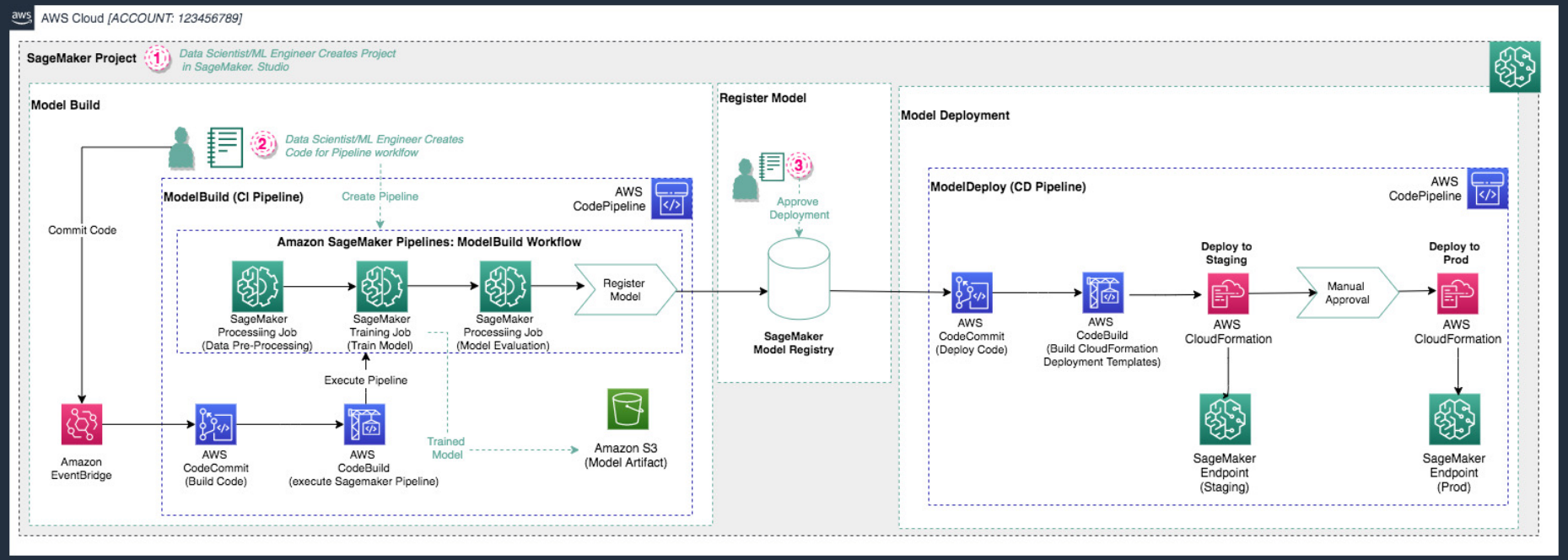
s3fs をノートブックでpip installしておくと、
S3から直接DataFrameに読み込みが可能

	State	Account Length	Area Code	Phone	Intl Plan	VMail Plan	VMail Message	Day Mins	Day Calls	Day Charge	...	Eve Calls	Eve Charge	Night Mins	Night Calls	Night Charge	Intl Mins
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3328	AZ	192	415	414-4276	no	yes	36	156.2	77	26.55	...	126	18.32	279.1	83	12.56	9.9

SageMaker Pipelines

SageMaker Studio から数クリックで構築可能

使い方は動画後半で



データ

SageMakerジョブと同等

環境

SageMakerジョブと同等

ソース・設定

リポジトリに保存される

振り返り

SageMakerジョブと同等

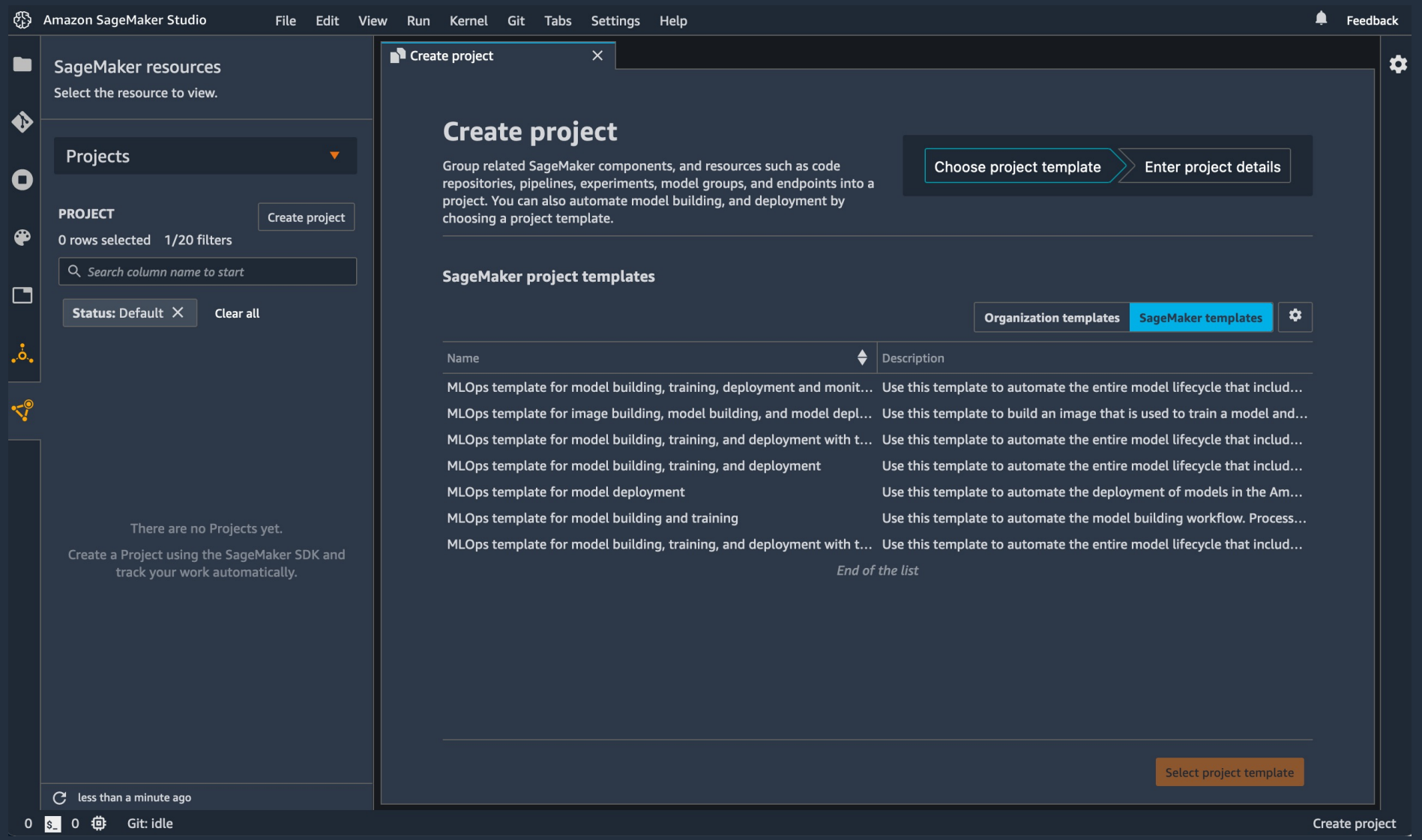
・MLOpsエンジニアはパイプラインを構築・管理する

・成果物をリポジトリに提出するルールとすることで、ガバナンスを確立

・データサイエンティストはコーディング、リポジトリへのpushを担当し、モデル構築に注力する

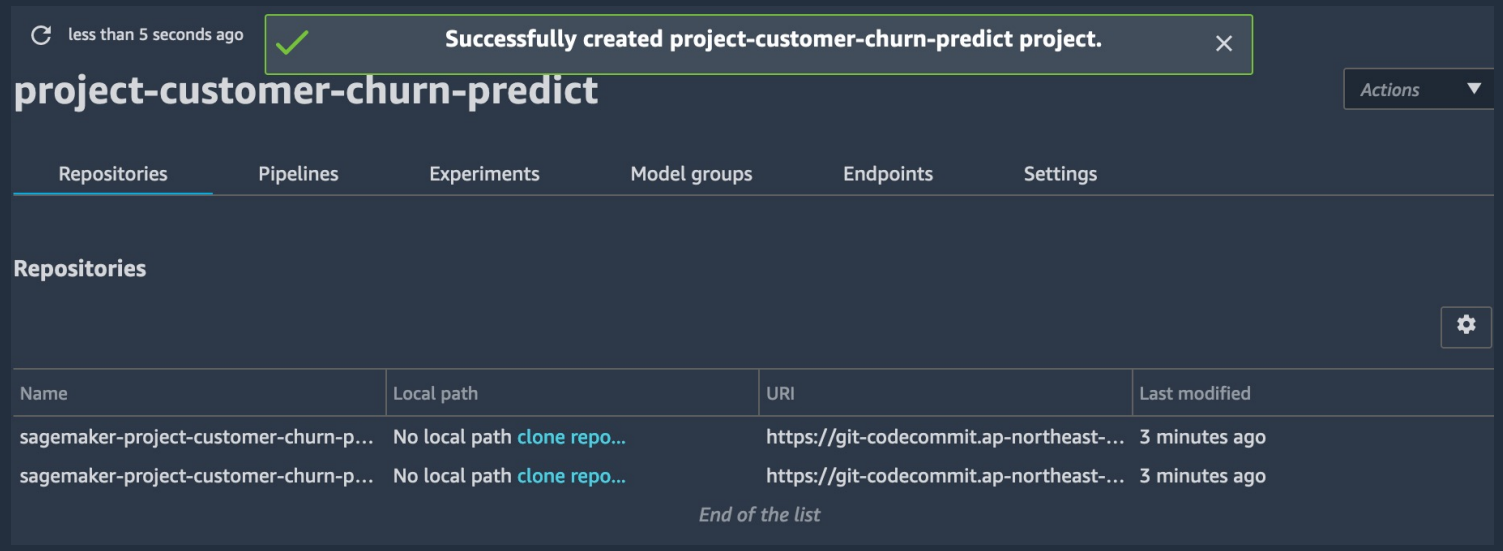
https://aws.amazon.com/blogs/machine-learning/building-automating-managing-and-scaling-ml-workflows-using-amazon-sagemaker-pipelines/

SageMaker Studioから、Projects を選択

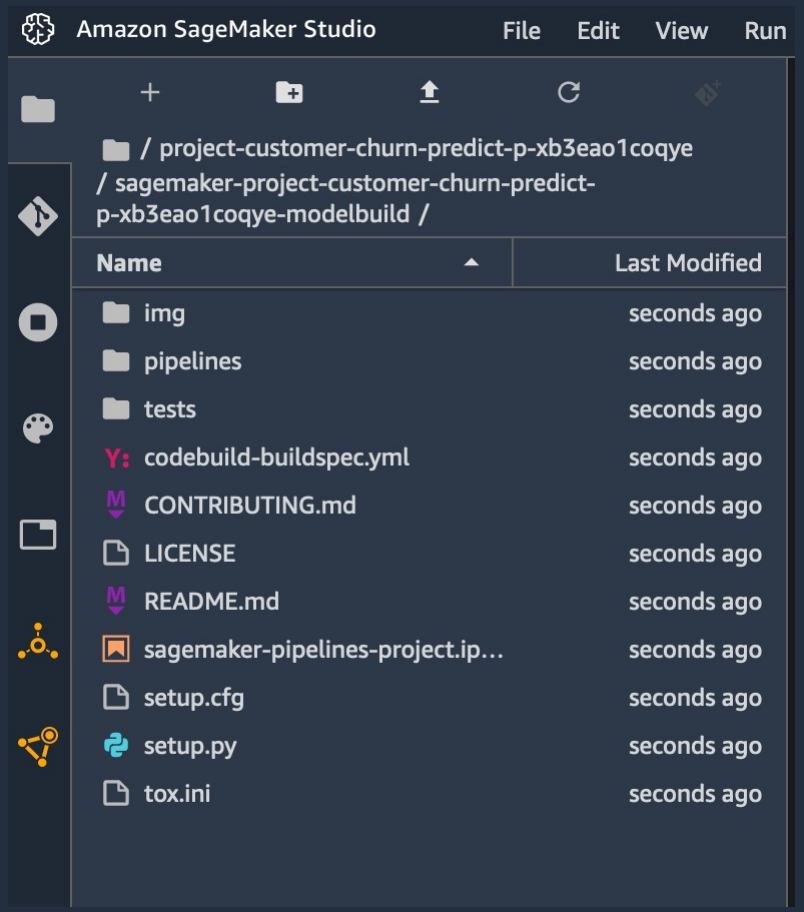


テンプレートを展開し、構築完了

パイプライン完成(5分程度)

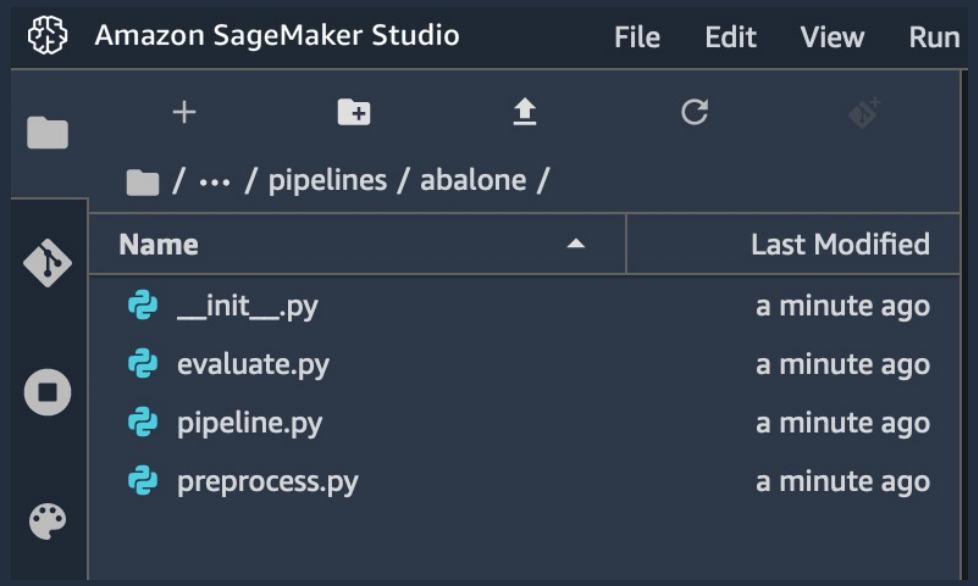


Cloneされたファイル群

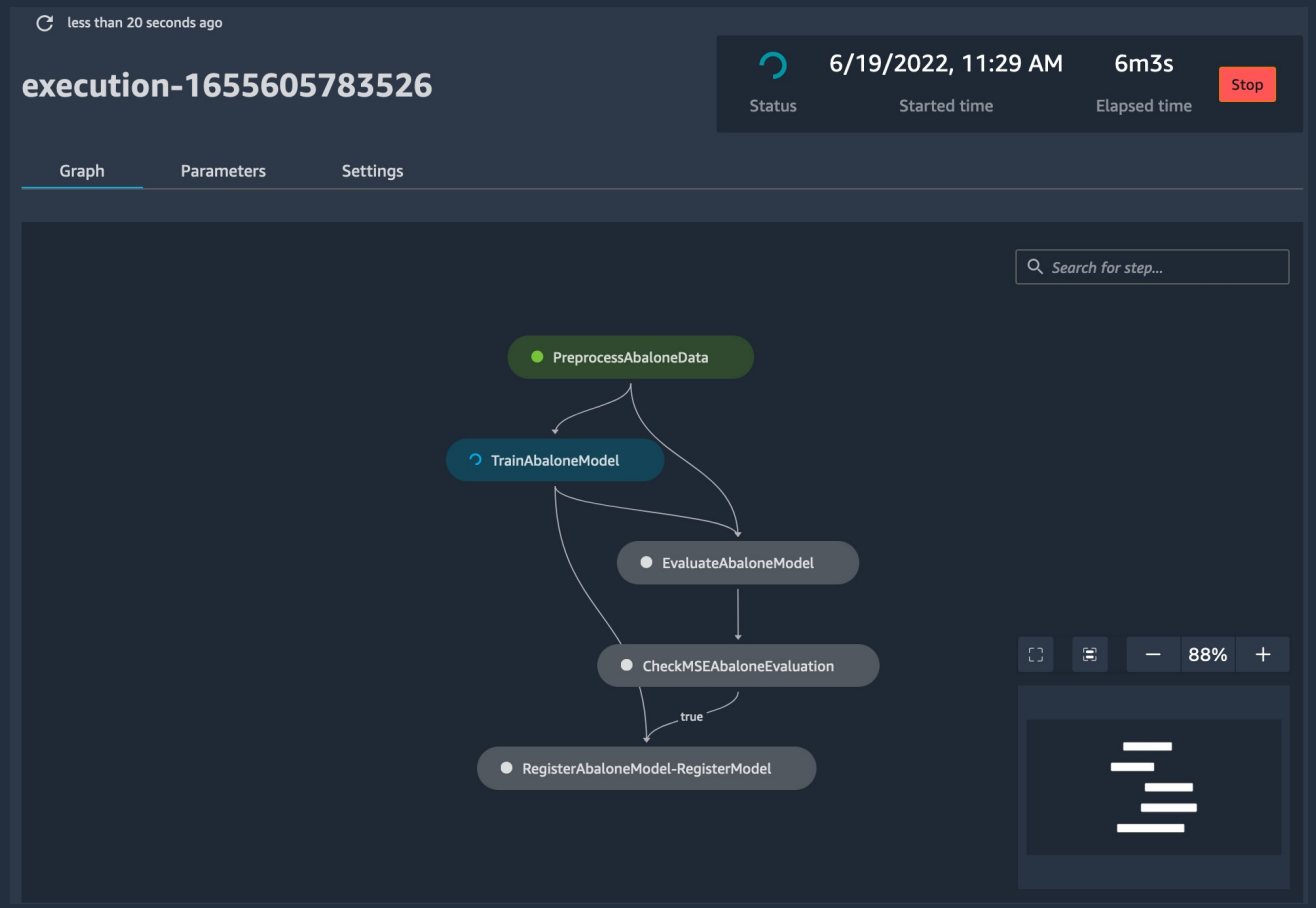


リポジトリpushをトリガに、パイプラインを実行

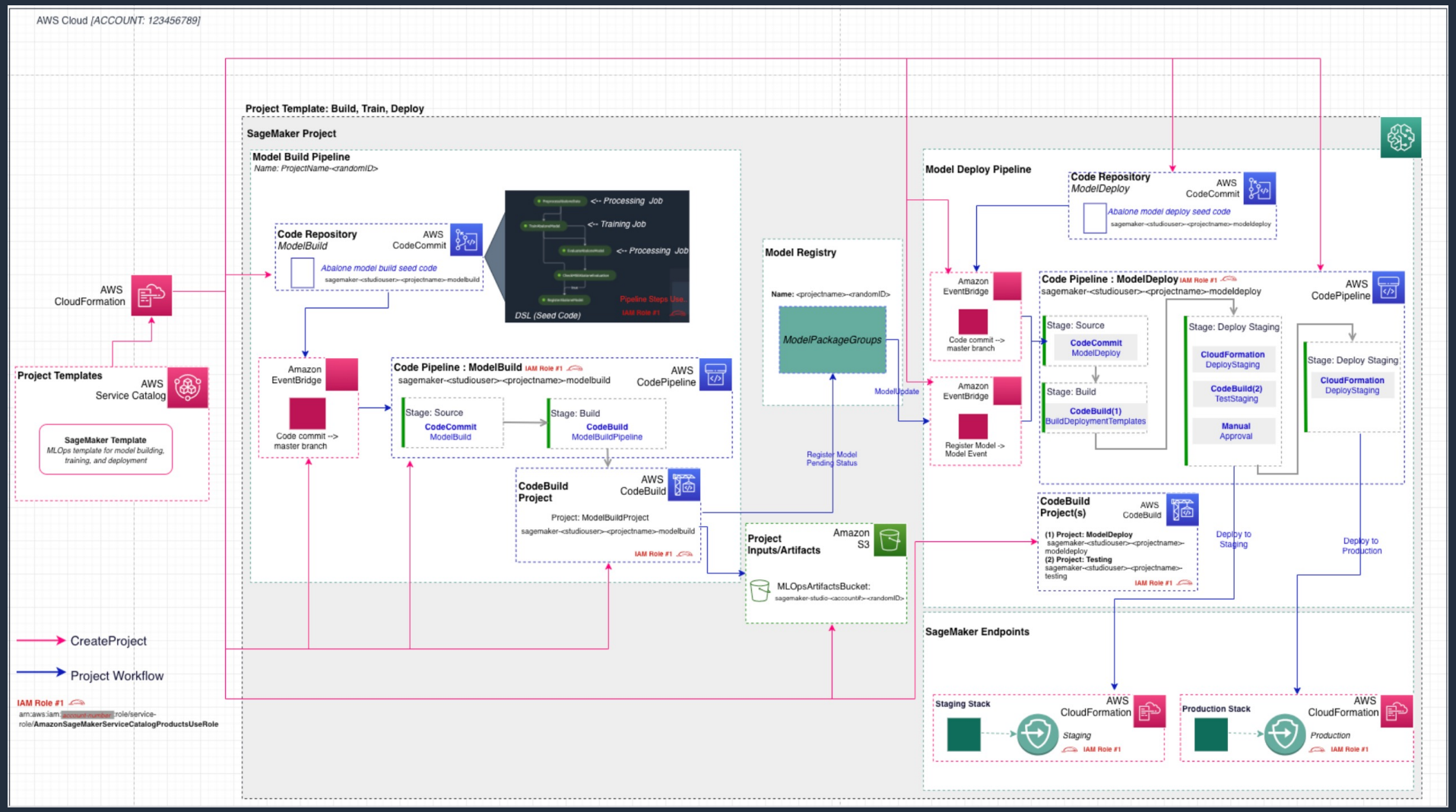
コードを変更してpush



パイプラインが実行される

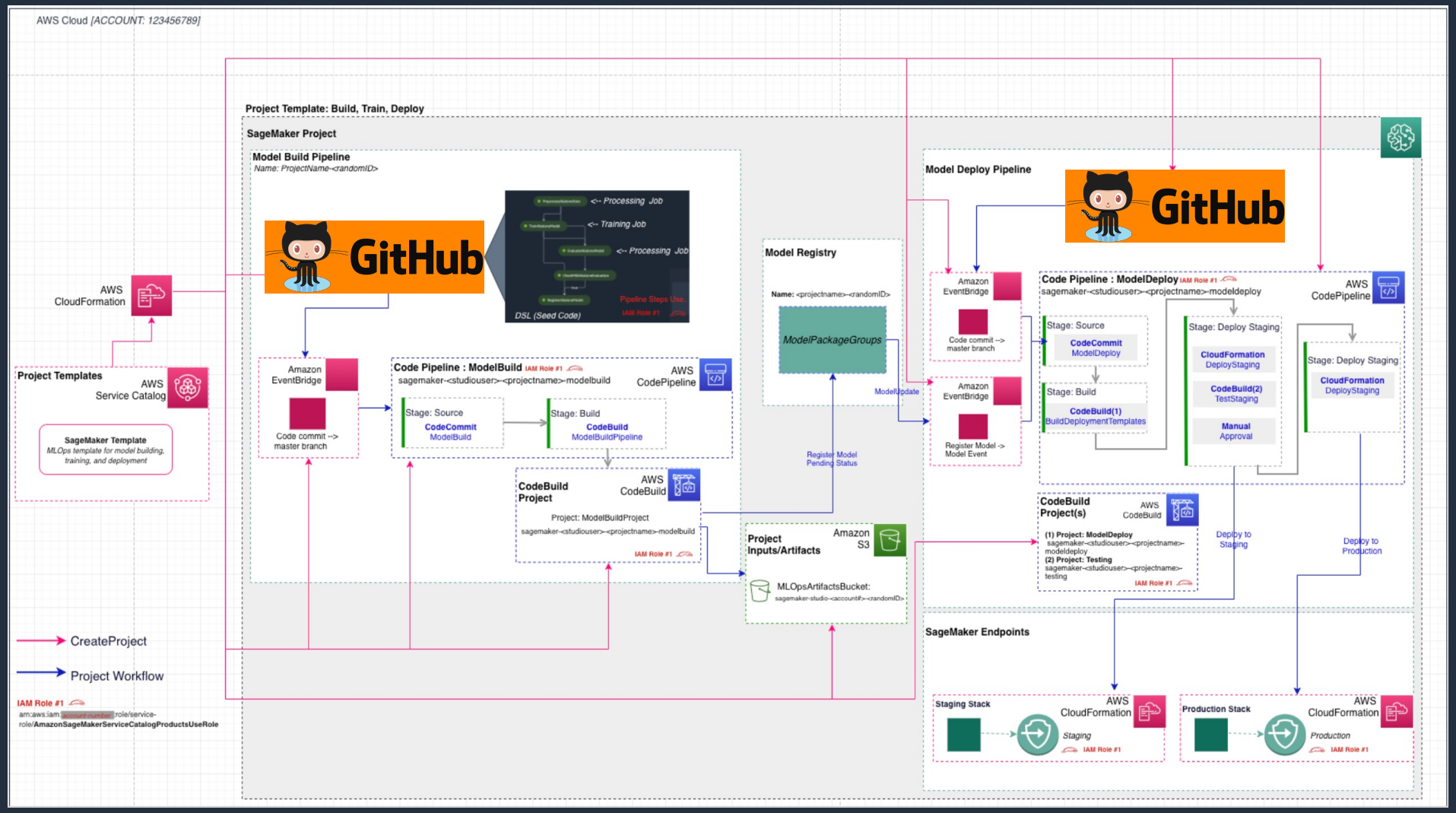


参考：アーキテクチャ – AWS CodeCommit 版



https://catalog.us-east-1.prod.workshops.aws/workshops/63069e26-921c-4ce1-9cc7-dd882ff62575/ja-JP/lab6

参考：アーキテクチャ – GitHub 版



AWS CodeStar Connectionの作成

コネクション作成

```
aws codestar-connections create-connection \
--provider-type GitHub\
--connection-name MyConnection \
--tags Key=sagemaker,Value=true
```

コネクション確認

```
aws codestar-connections list-connections
```

タグ確認

```
aws codestar-connections list-tags-for-resource \
  --resource-arn <codestar-connections ARN>
```

この時点では[Pending]ステータス

AWS CloudShell

ap-northeast-1

If the arrow keys aren't working correctly in PowerShell, see Troubleshooting AWS CloudShell

Don't show this message again

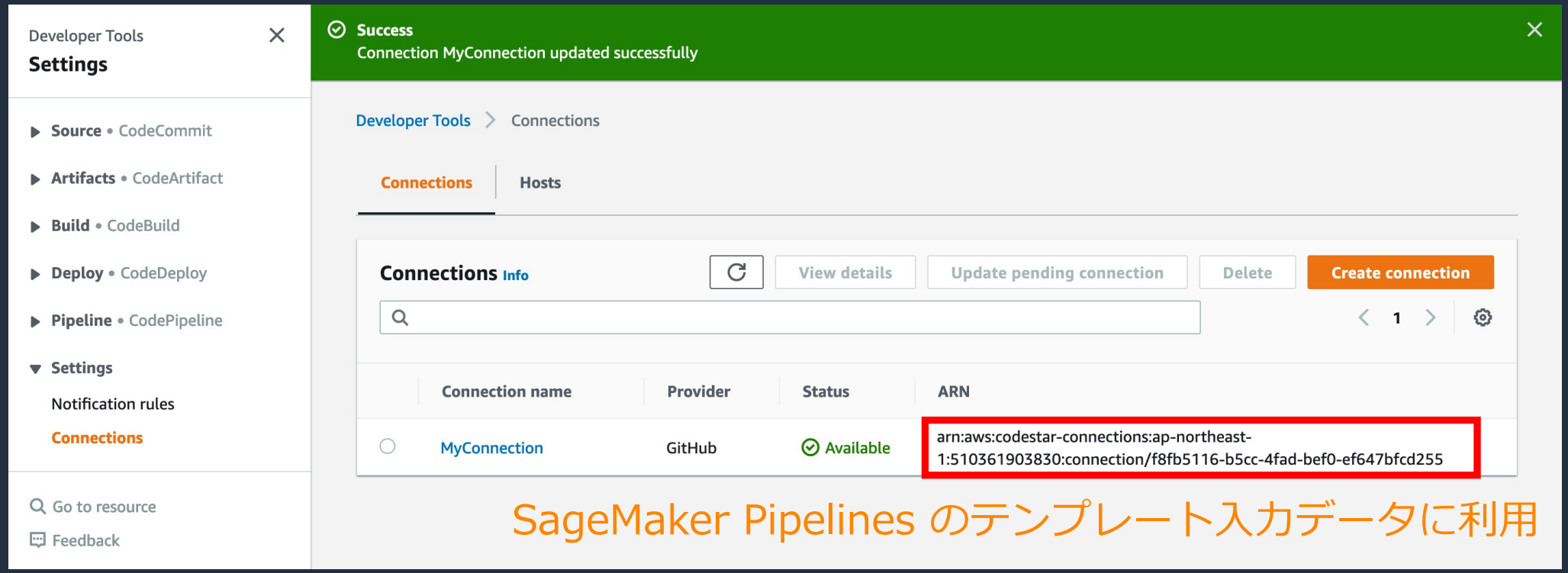
```
Preparing your terminal...
[cloudshell-user@ip-10-0-174-255 ~]$ Try these commands to get started:
aws help  or  aws <command> help  or  aws <command> --cli-auto-prompt
[cloudshell-user@ip-10-0-174-255 ~]$ aws codestar-connections create-connection \
>     --provider-type GitHub\
>     --connection-name MyConnection \
>     --tags Key=sagemaker,Value=true
{
    "ConnectionArn": "arn:aws:codestar-connections:ap-northeast-1:510361903830:connection/f8fb5116-b5cc-4fad-bef0-ef647bfcd255",
    "Tags": [
        {
            "Key": "sagemaker",
            "Value": "true"
        }
    ]
}
```

```
[cloudshell-user@ip-10-0-174-255 ~]$ aws codestar-connections list-connections
{
    "Connections": [
        {
            "ConnectionName": "MyConnection",
            "ConnectionArn": "arn:aws:codestar-connections:ap-northeast-1:510361903830:connection/f8fb5116-b5cc-4fad-bef0-ef647bfcd255",
            "ProviderType": "GitHub",
            "OwnerAccountId": "510361903830",
            "ConnectionStatus": "PENDING"
        }
    ]
}
```

```
[cloudshell-user@ip-10-0-174-255 ~]$ aws codestar-connections list-tags-for-resource     --resource-arn arn:aws:codesta
r-connections:ap-northeast-1:510361903830:connection/f8fb5116-b5cc-4fad-bef0-ef647bfcd255
{
    "Tags": [
        {
            "Key": "sagemaker",
            "Value": "true"
        }
    ]
}
```

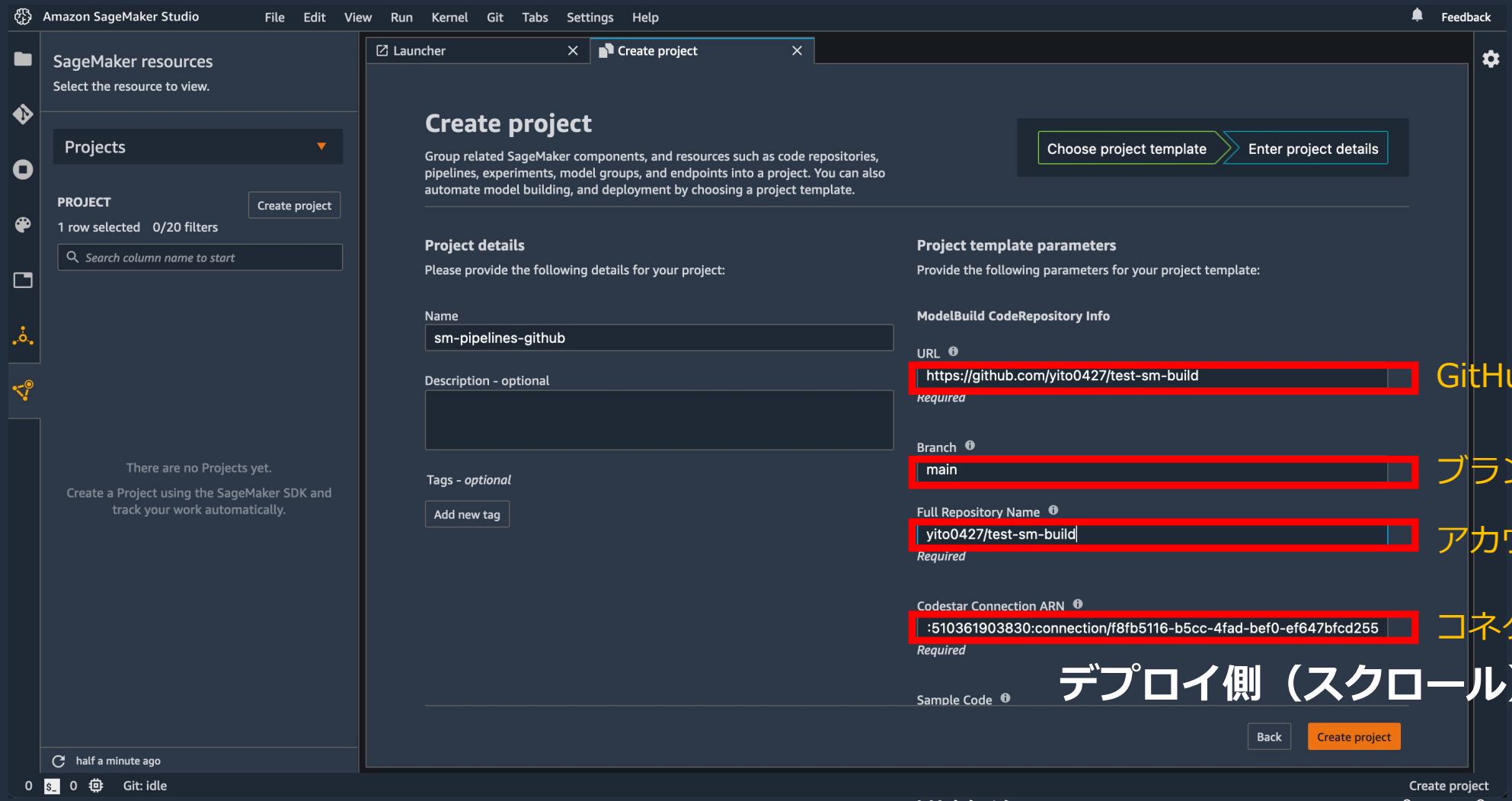
Connection の Available 化

マネジメントコンソール上から、作成したGitHubへのコネクションをactivateする



https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-projects-walkthrough-3rdgit.html

テンプレート入力（ビルト側設定）



GitHubリポジトリURL

ブランチ名

アカウント名/リポジトリ名

コネクションARN

デプロイ側（スクロール）も同様に設定

構築後、GitHubへのpushでパイプラインが起動

リファレンス

Amazon SageMaker Experiments Python SDK doc
https://sagemaker-experiments.readthedocs.io/en/latest/#

SageMaker Experiments Python SDK
https://github.com/aws/sagemaker-experiments

GitHub samples
https://github.com/aws/amazon-sagemaker-examples/tree/main/sagemaker-experiments

Developer Guide
https://docs.aws.amazon.com/sagemaker/latest/dg/experiments.html

次回予告（Dark Part）

Amazon SageMaker Training（デモ編）

前回の動画「Amazon SageMaker Training（座学編）」で紹介したコードのデモを実施します

ML Enablement Seriesの動画

機械学習モデルをビジネス価値につなげる方法を全力で解説！

Light Part

製品やサービスに機械学習を導入するプロジェクトの進め方

https://bit.ly/3M1F9as

Step Up!!

Dark Part

機械学習モデルの開発や運用をマネージドサービスで効率的に行う方法

https://bit.ly/3927PCN

資料集・お問合せ・Special Thanks

AWSの日本語資料の場所: 「AWS 資料」で検索

AWSのハンズオン資料の場所: 「AWS ハンズオン」で検索

お問合せ

技術的なお問合せ

料金のお問合せ

個別相談会のお申込み

Special Thanks

- 音楽素材: PANICPUMPKIN様

Thank you!