AIM301

Use SageMaker to develop ML models faster, featuring Qualtrics

Dr. Ali Arsanjani (he/him)
WW Tech Lead & Chief Principal Architect
AI/ML Specialist Solution Architecture
AWS

Samir Joshi (he/him)
Machine Learning Engineer
iQ R&D
Qualtrics

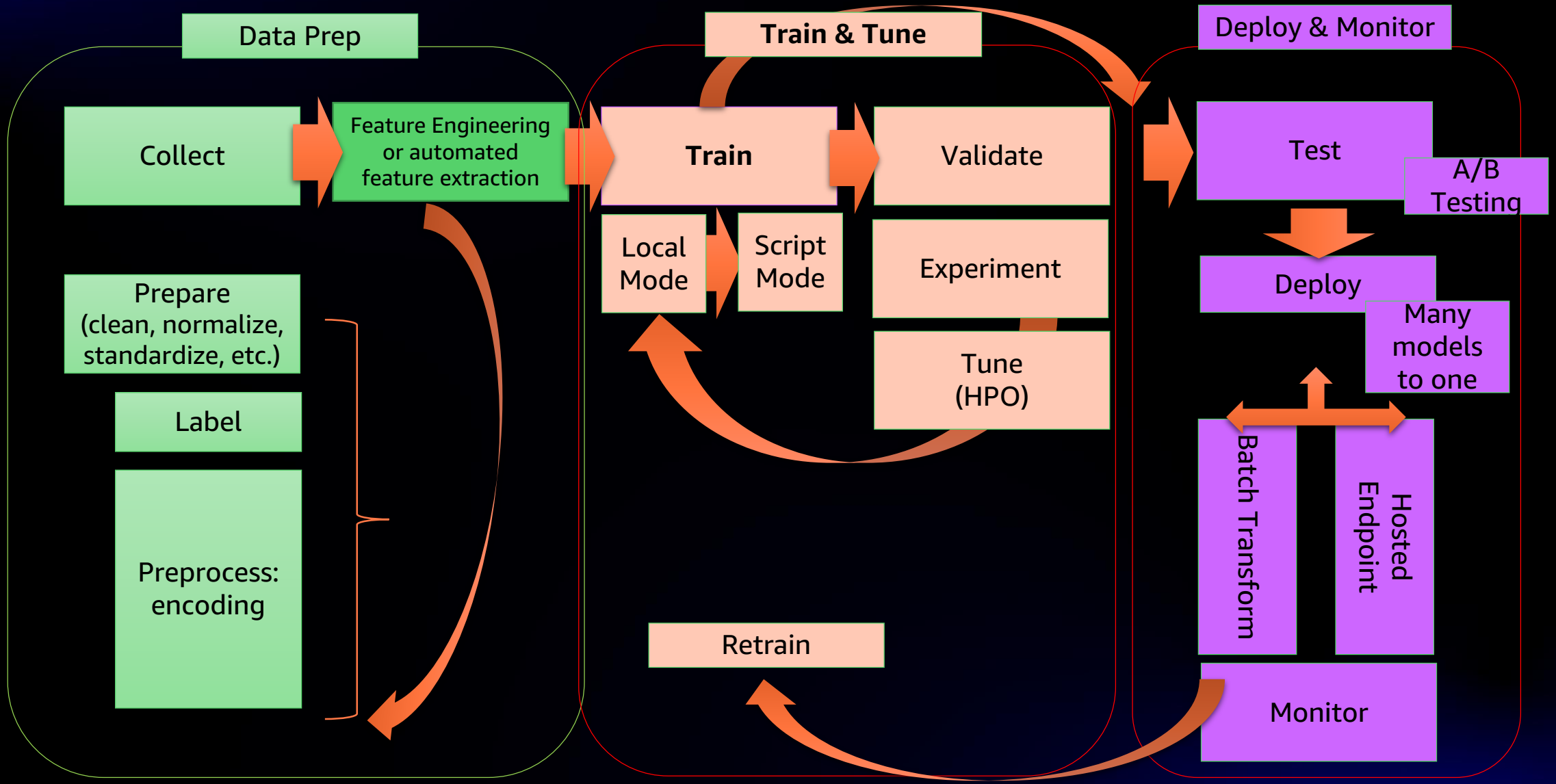
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

1. The ML lifecycle for enterprise ML: Old and new features
2. Experimental and automated ML sections of the lifecycle
3. Music recommendation: An end-to-end example
4. Factors for Accelerating ML maturity and adoption
5. Use-case exploration
6. Q&A

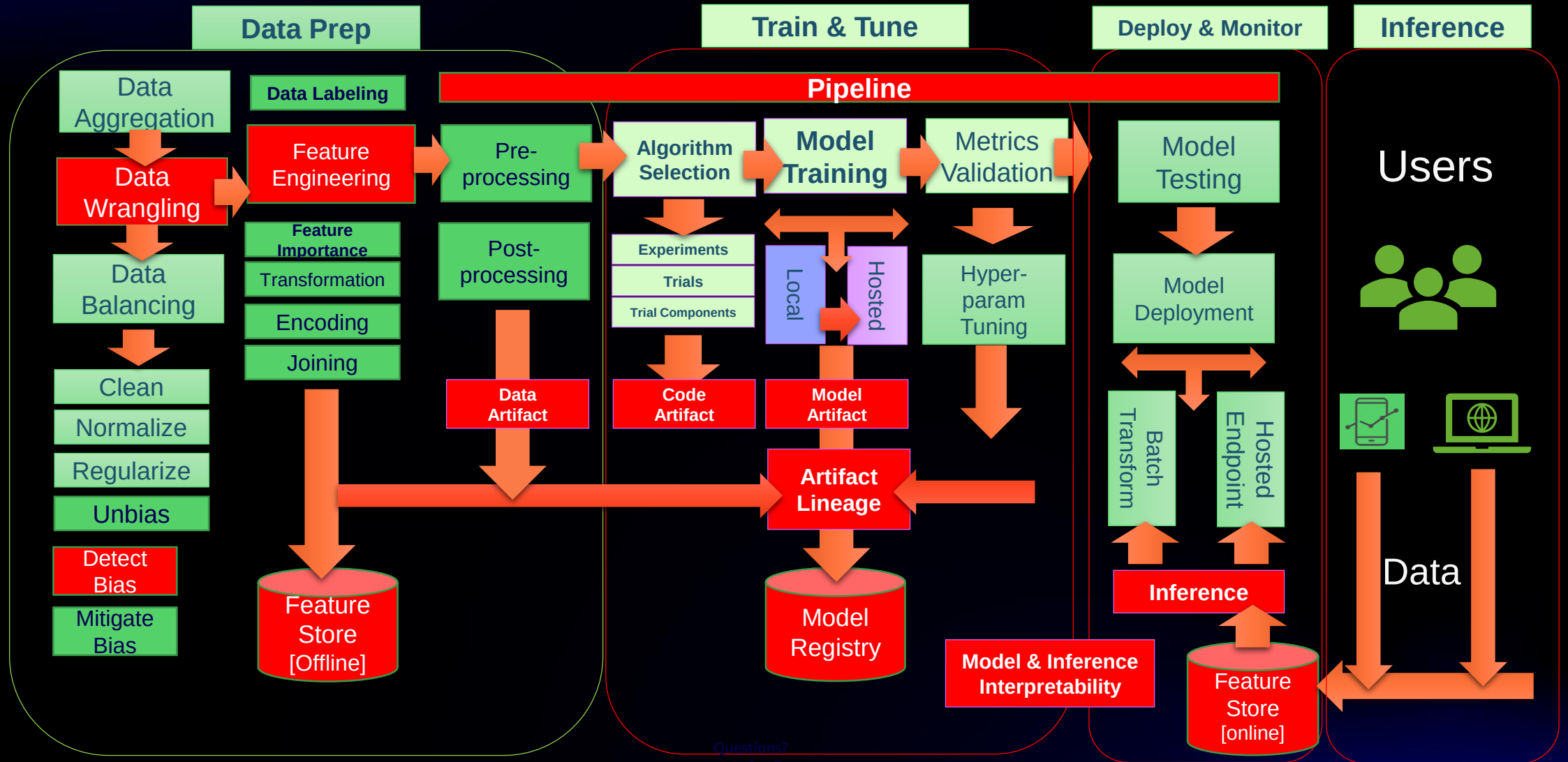
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML lifecycle: High-level view



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The ML Lifecycle: Detailed view

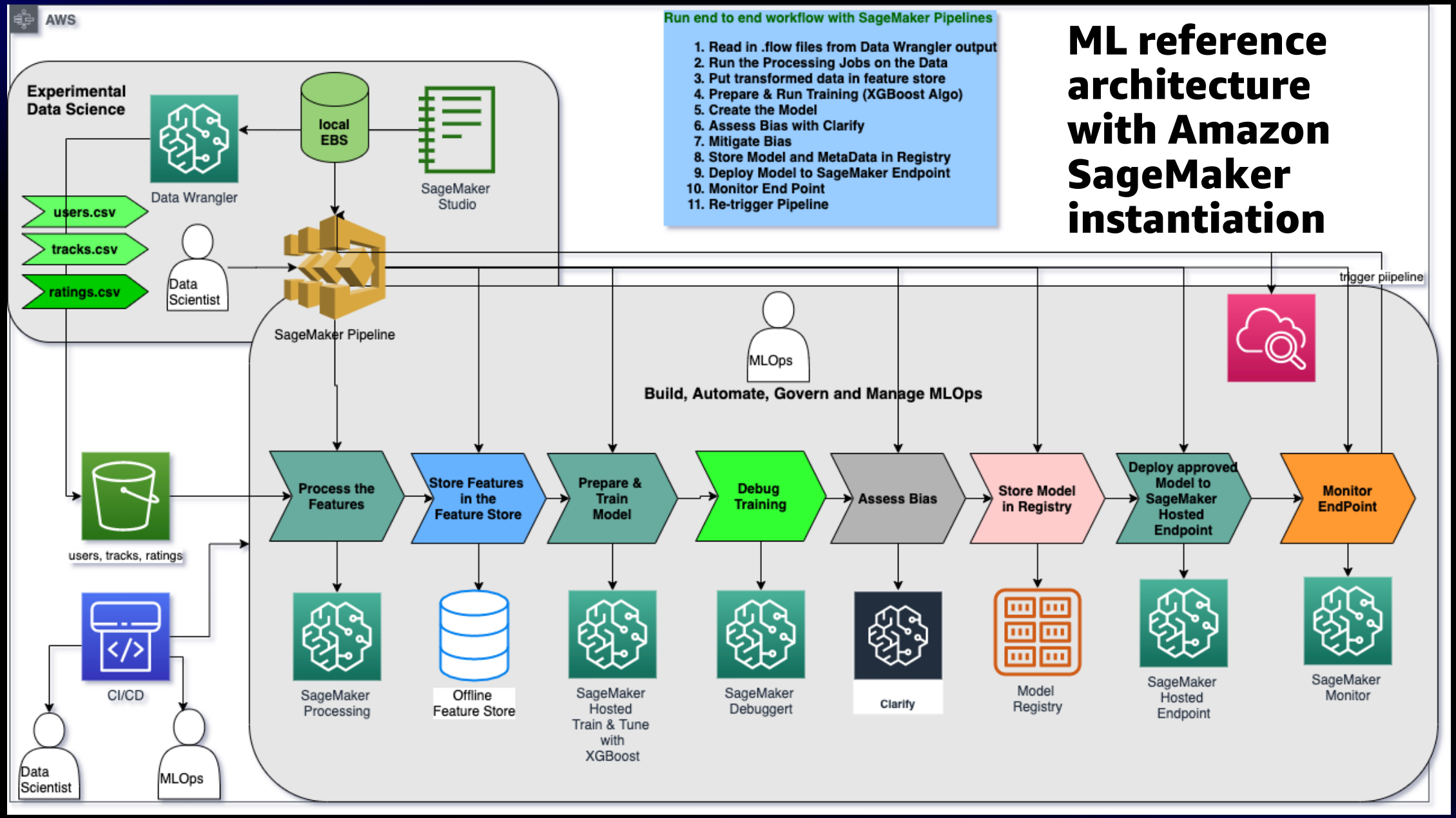


Questions?
arsanjan@amazon.com

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML reference architecture with Amazon SageMaker instantiation



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

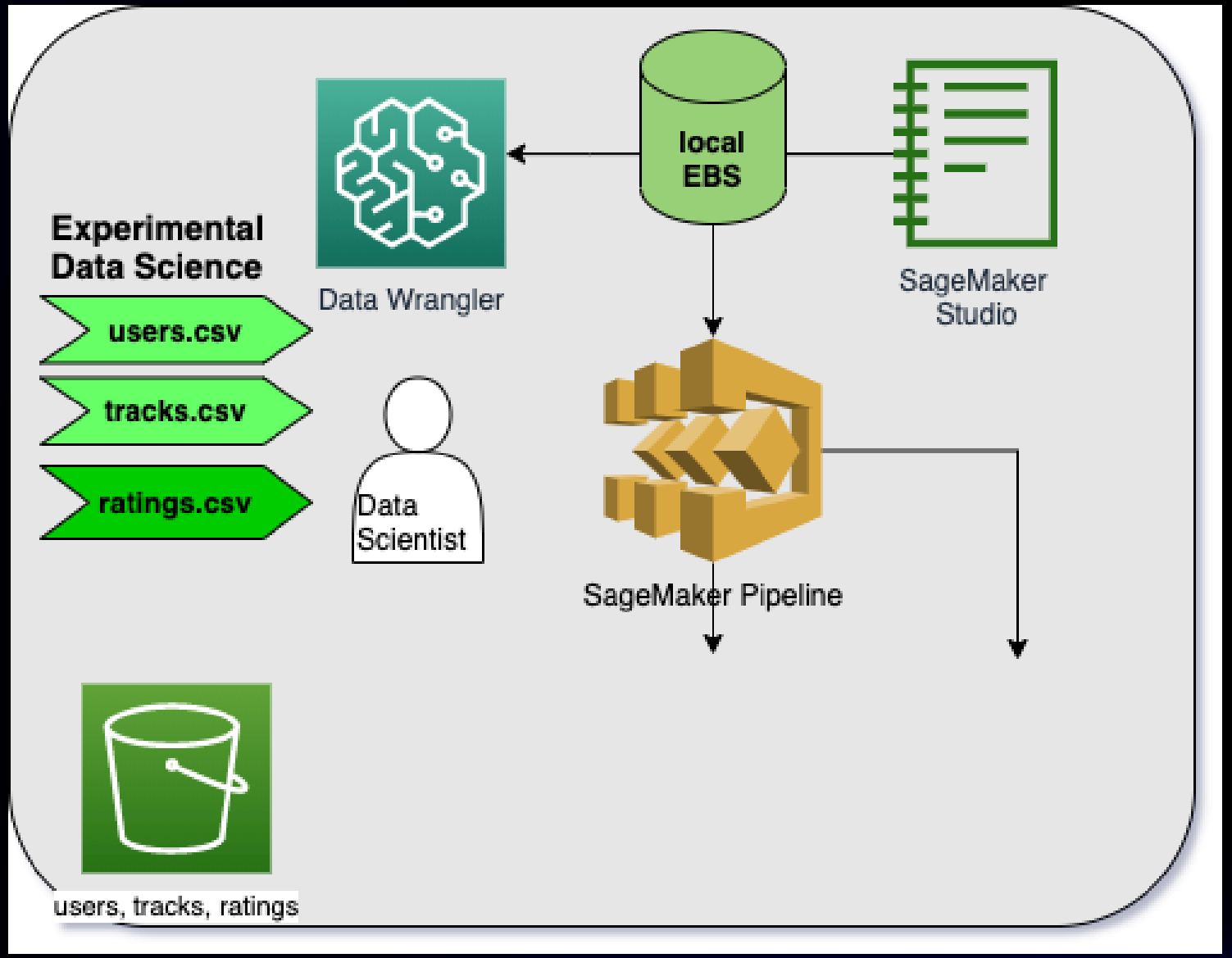
Let’s see an end-to-end implementation:

http://github.com/aws-sagemaker-examples-end-to-end/music_recommendation



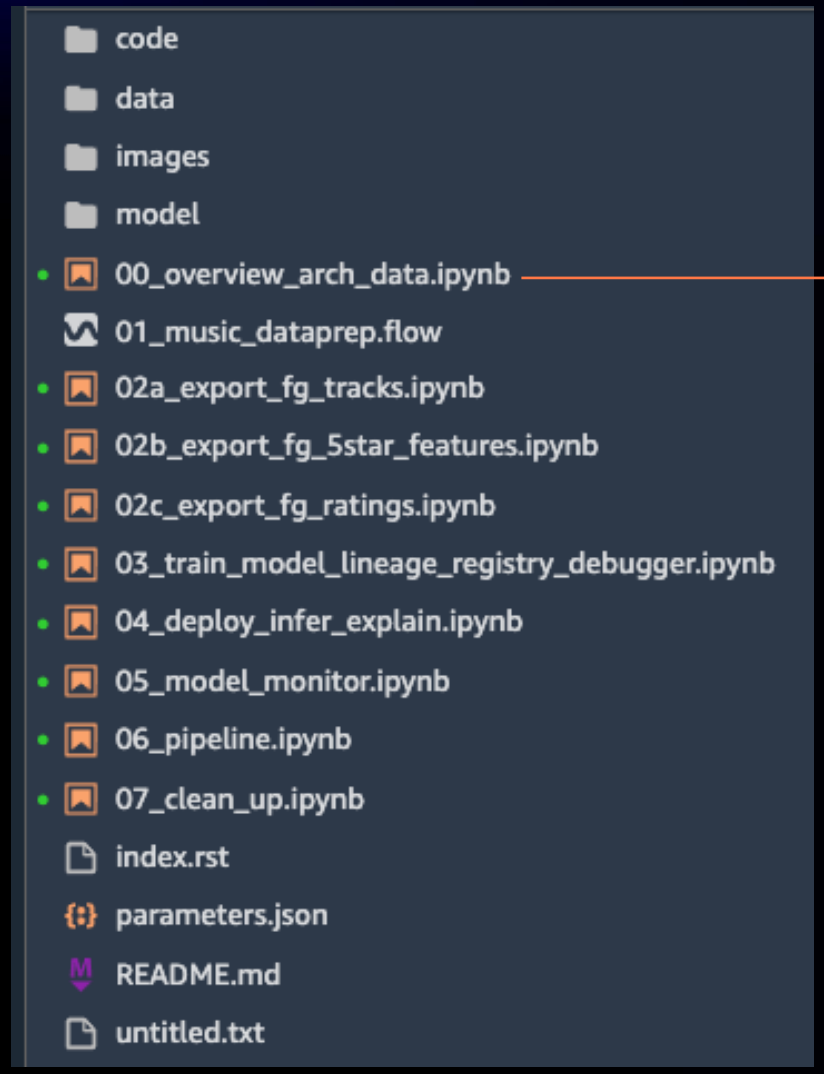
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Exploratory Data Science



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation overview and data



Explore the Data

back to top

[11]:
```
tracks = pd.read_csv('./data/tracks.csv')
ratings = pd.read_csv('./data/ratings.csv')
```

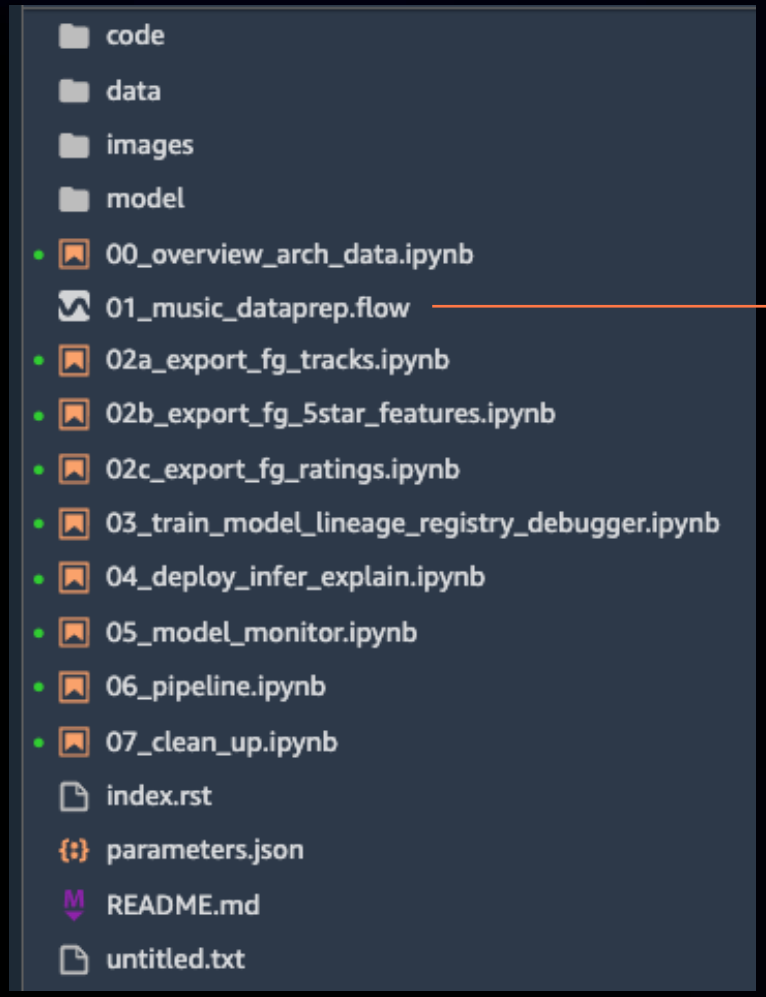
[12]:
```
tracks.head()
```

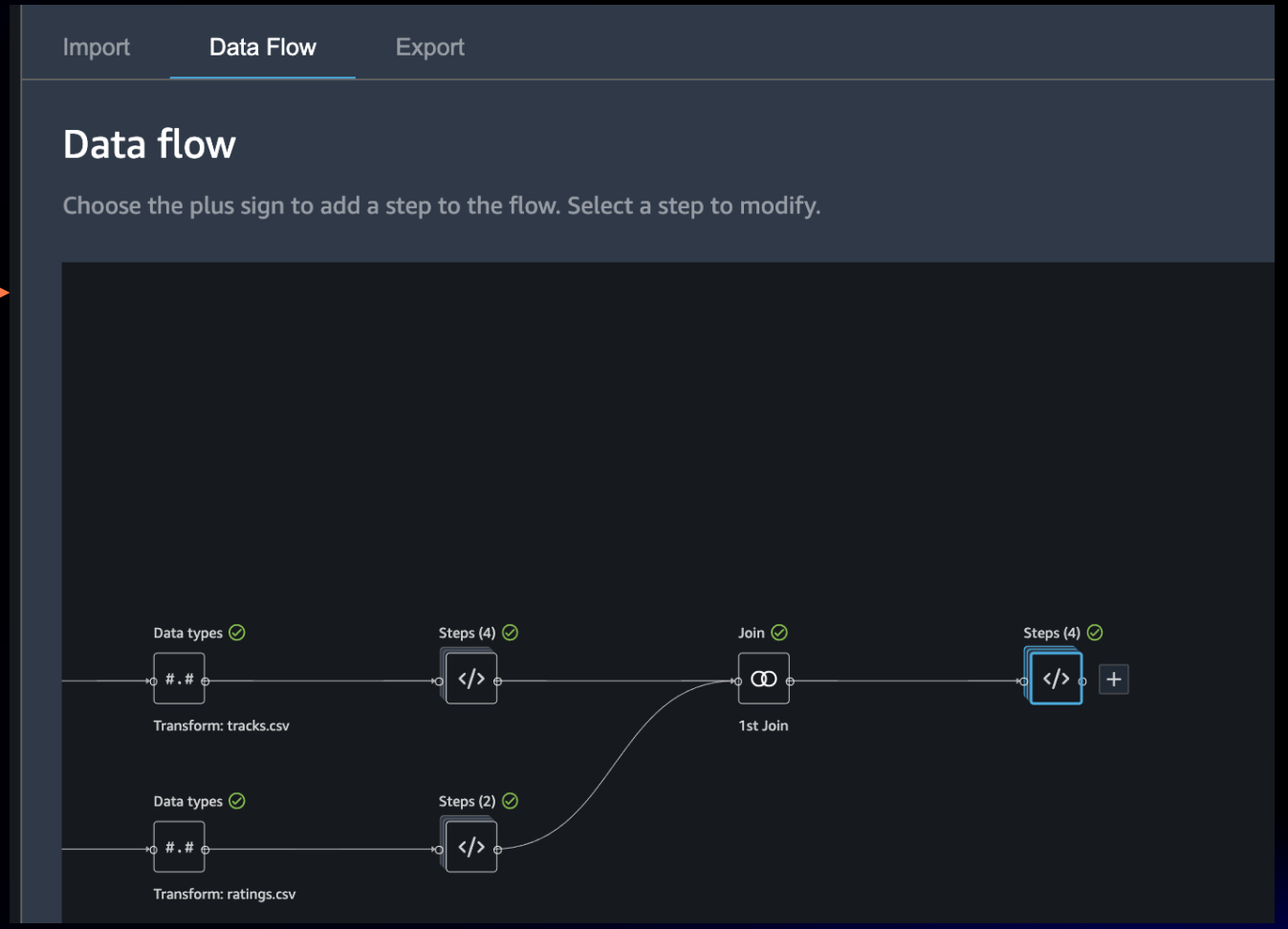
[12]:

	trackId	length	energy	acousticness	valence	speechiness	instrumentalness	liveness	tempo	ger
0	ef529fb1-37d9-47db-98a9-aeccb75a7481	539.0	0.901144	0.000001	0.104279	0.143046	0.000000	0.021130	107.867115	Ja
1	ec0b785a-e9b0-4573-9823-309aefb99f73	236.0	0.795516	0.470181	0.660053	0.122248	0.935359	0.436054	105.471412	Regg
2	ef0784fe-9f11-4996-9582-df2ef888d4c2	156.0	0.673727	0.020541	0.154176	0.078040	0.765548	0.105727	132.955951	Coun
3	1888a3b0-07ce-4982-85c7-c3ba0aa43092	1071.0	0.238988	0.824696	0.136747	0.022800	0.000000	0.371890	194.301703	Blu
4	f2ab455a-8fb7-4233-9e5c-1f9d1c200e24	196.0	0.674222	0.000001	0.552028	0.162047	0.897425	0.379058	130.164660	La

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use Data Wrangler to define transformations for SageMaker Processing and EDA



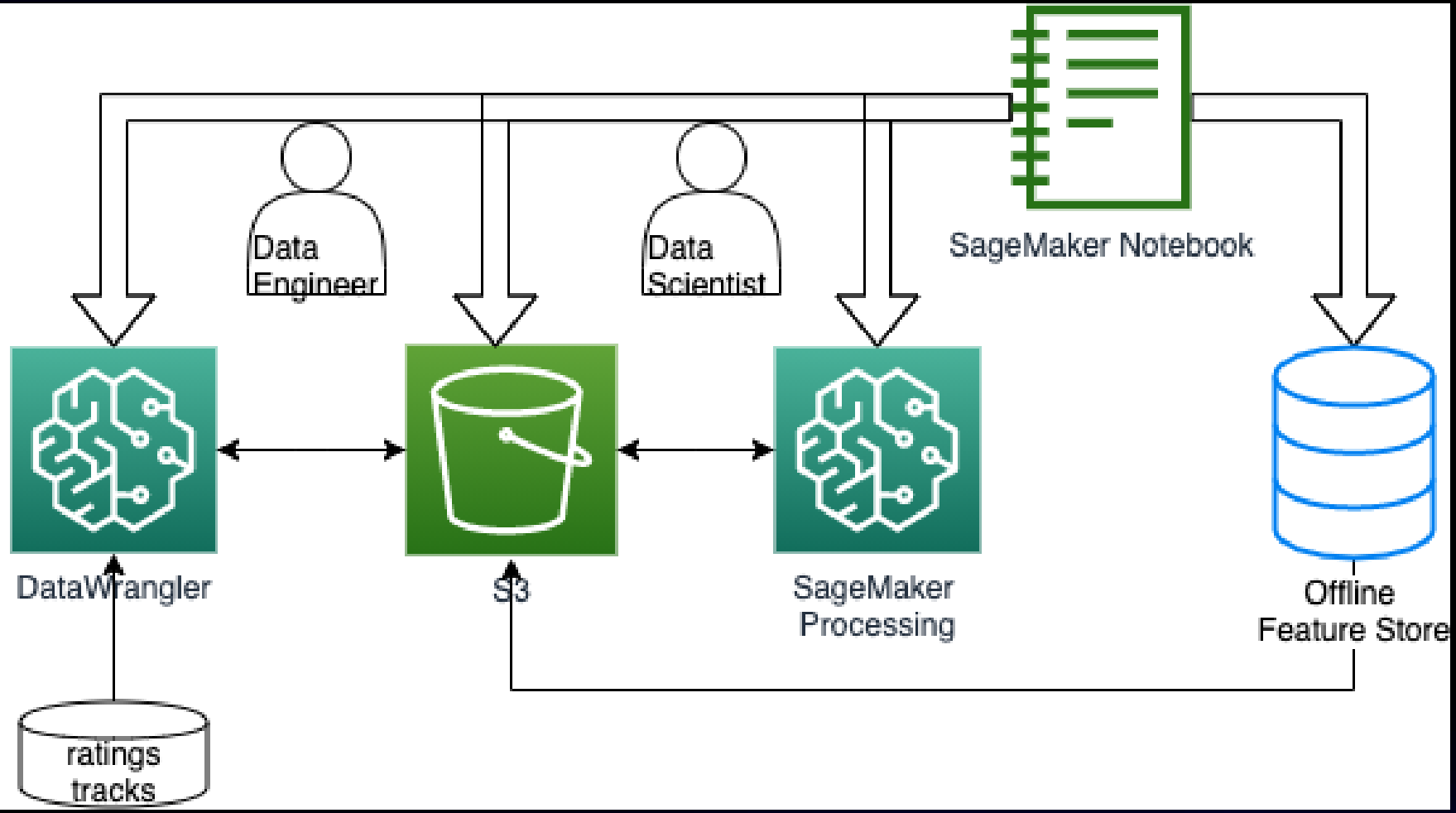


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data prep

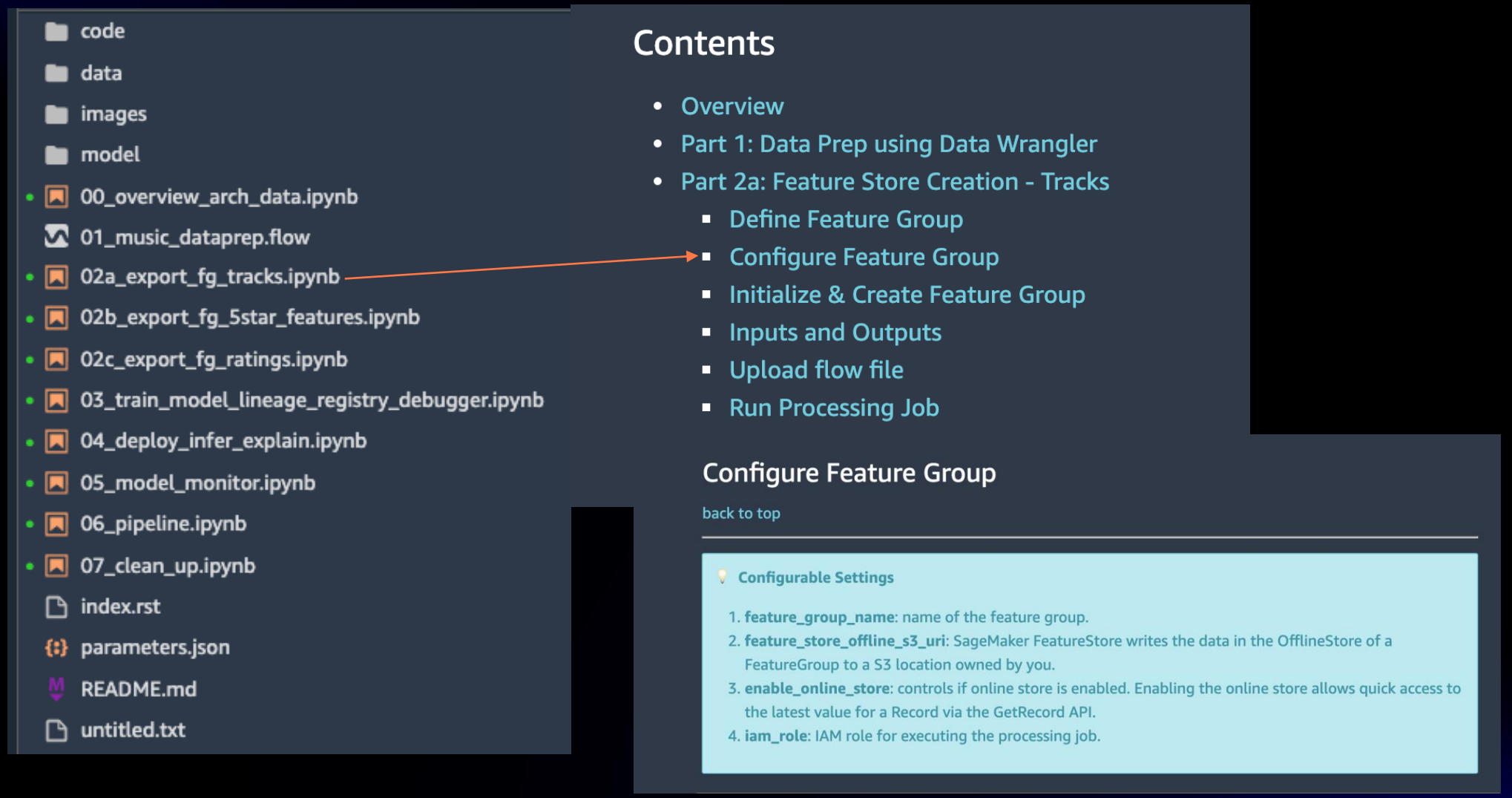
Data Prep:

1. Collect
2. Wrangle
3. Featurize
4. Transform
5. Pre-process
6. Store in Feature Store



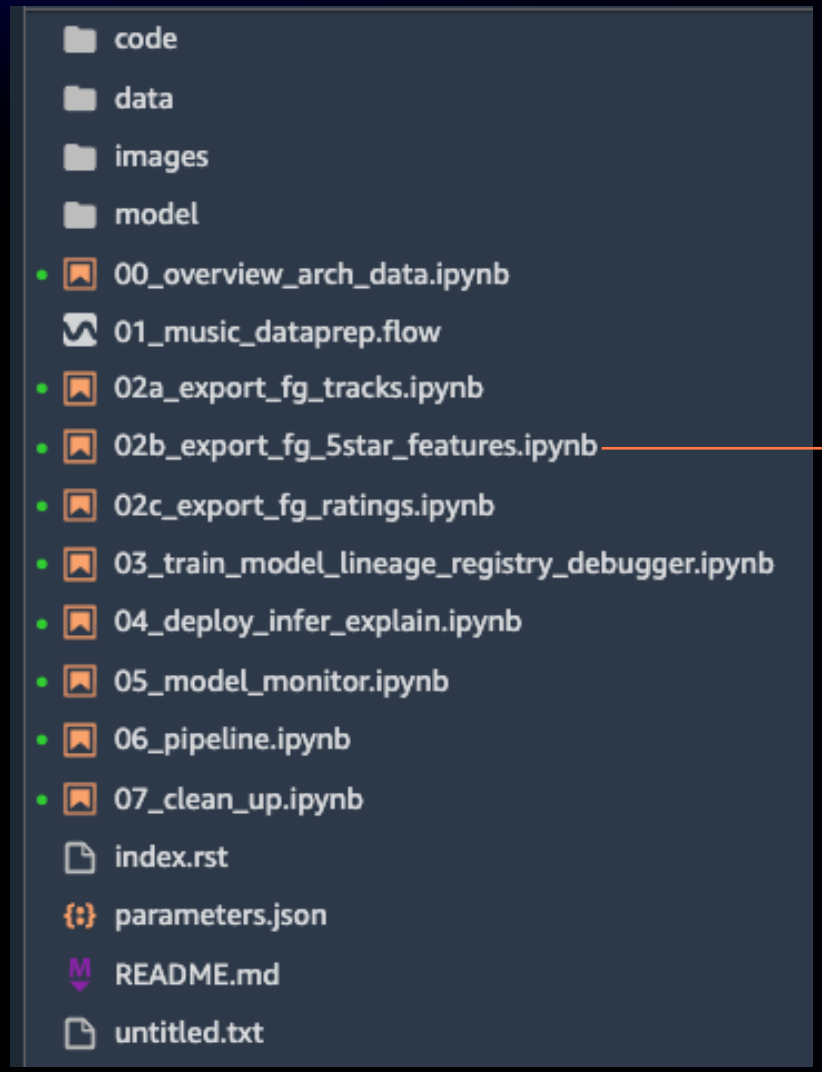
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Prep: Configuring the Feature Store



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Prep: Process features and deposit in Feature Store




```
from sagemaker.feature_store.feature_group import FeatureGroup

feature_group = FeatureGroup(
    name=feature_group_name, sagemaker_session=feature_store_session, feature_definitions=feat

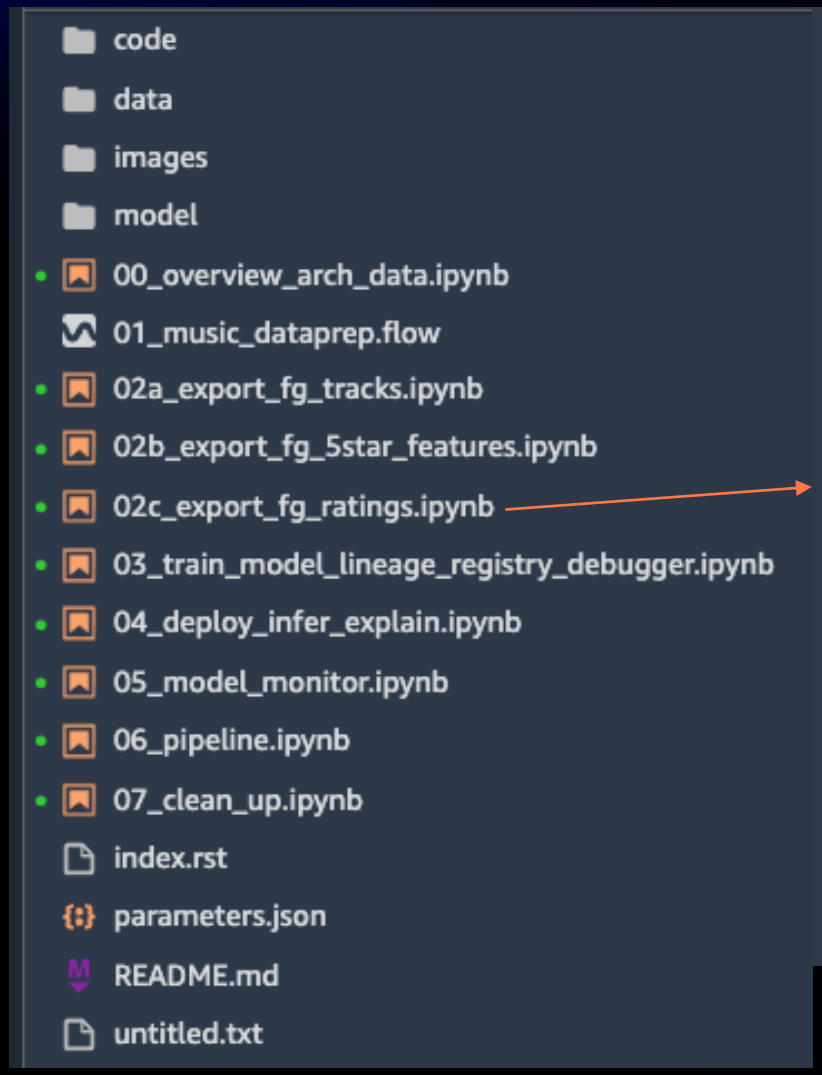
# only create feature group if it doesn't already exist
try:
    sagemaker_client.describe_feature_group(FeatureGroupName=feature_group_name, NextToken='st
    feature_group_exists=True
    print("Feature Group {0} already exists. Using {0}".format(feature_group_name))
except Exception as e:
    feature_group.create(
        s3_uri=feature_store_offline_s3_uri,
        record_identifier_name=record_identifier_feature_name,
        event_time_feature_name=event_time_feature_name,
        role_arn=iam_role,
        enable_online_store=enable_online_store
    )
    if error == 'ResourceInUse':
        feature_group_exists=True
        print("Feature Group {0} already exists. Using {0}".format(feature_group_name))
```

```
Feature Group user-5star-track-features-music-rec already exists. Using user-5star-track-feat
ures-music-rec
```

Invoke the Feature Store API to create the feature group and wait until it is ready

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Prep: Set up input dataset sources of the processing job



Input - S3 Source: tracks.csv

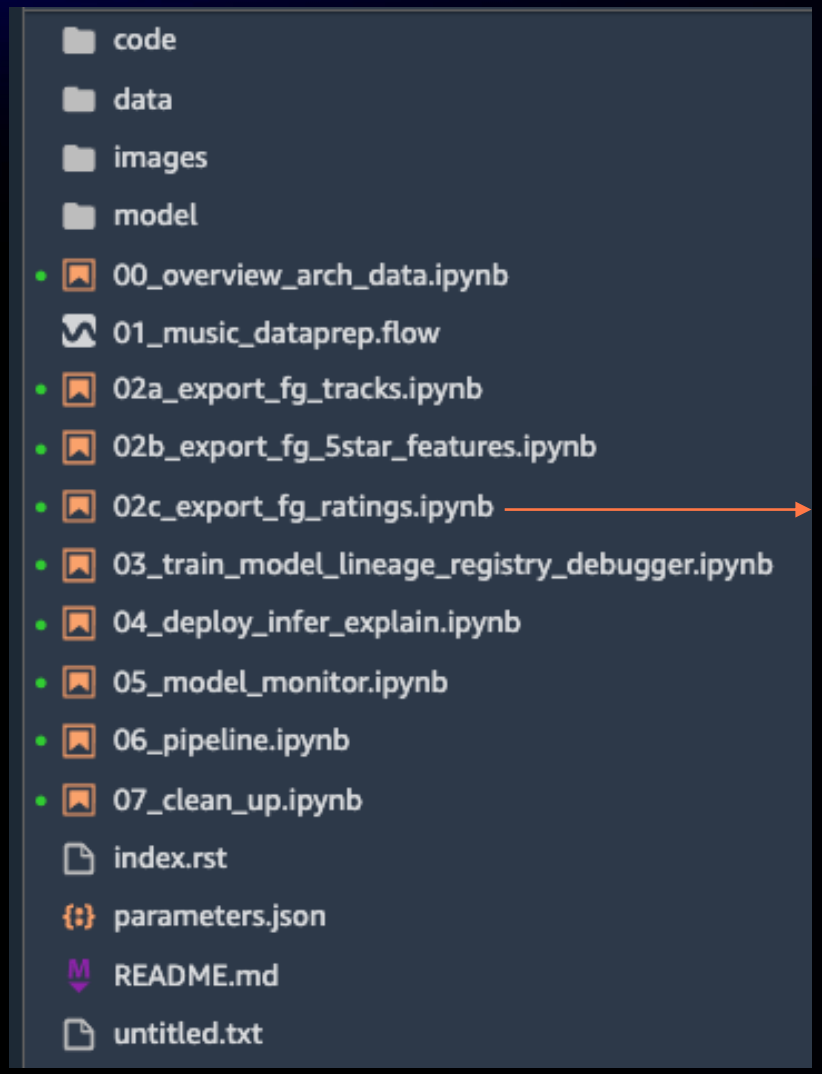
```
data_sources.append(ProcessingInput(
    source=f"{tracks_data_source}", # You could override this to point to another dataset on S
    destination="/opt/ml/processing/tracks.csv",
    input_name="tracks.csv",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
))
```

Input - S3 Source: ratings.csv

```
data_sources.append(ProcessingInput(
    source=f"{ratings_data_source}", # You could override this to point to another dataset on
    destination="/opt/ml/processing/ratings.csv",
    input_name="ratings.csv",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
))
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Prep: Set up output destination of the processing job



Output: Feature Store

Below are the inputs required by the SageMaker Python SDK to launch a processing job with feature store as an output.

```
]: from sagemaker.processing import FeatureStoreOutput

# Output name is auto-generated from the select node's ID + output name from the flow file.
output_name = "9a283380-91ca-478e-be99-6ba3bf57c680.default" # ratings node

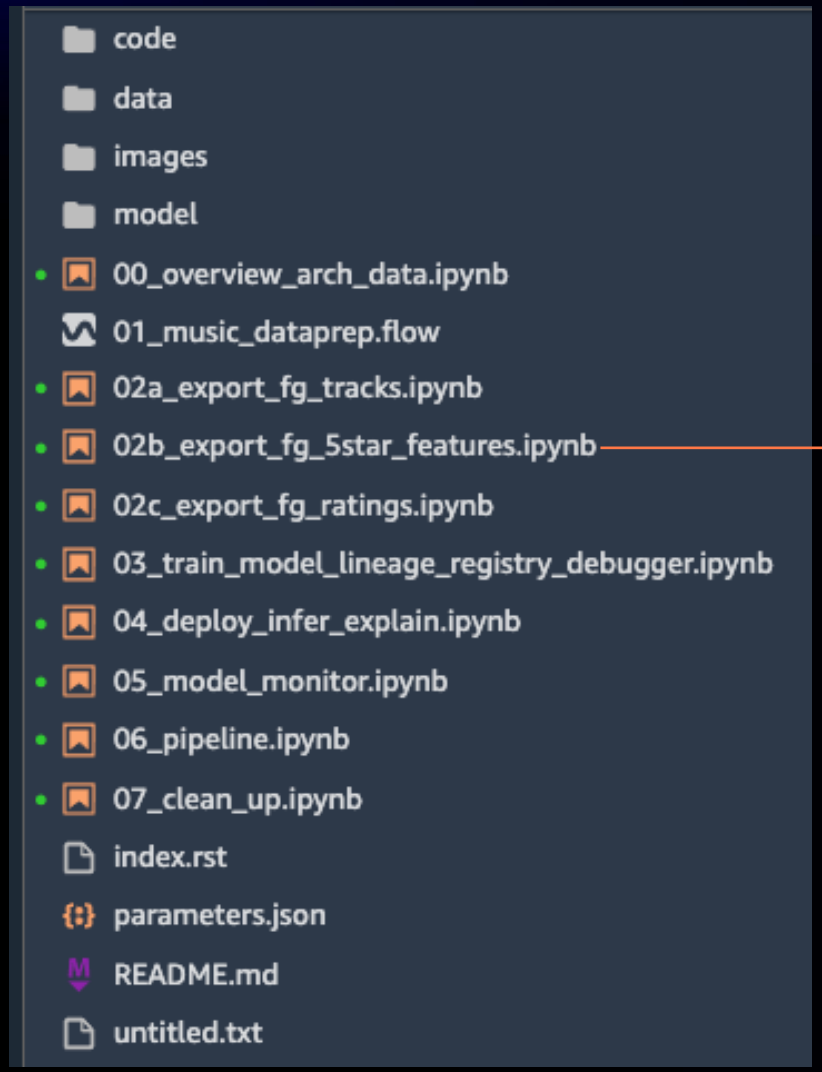
processing_job_output = ProcessingOutput(
    output_name=output_name,
    app_managed=True,
    feature_store_output=FeatureStoreOutput(feature_group_name=feature_group_name),
)
```

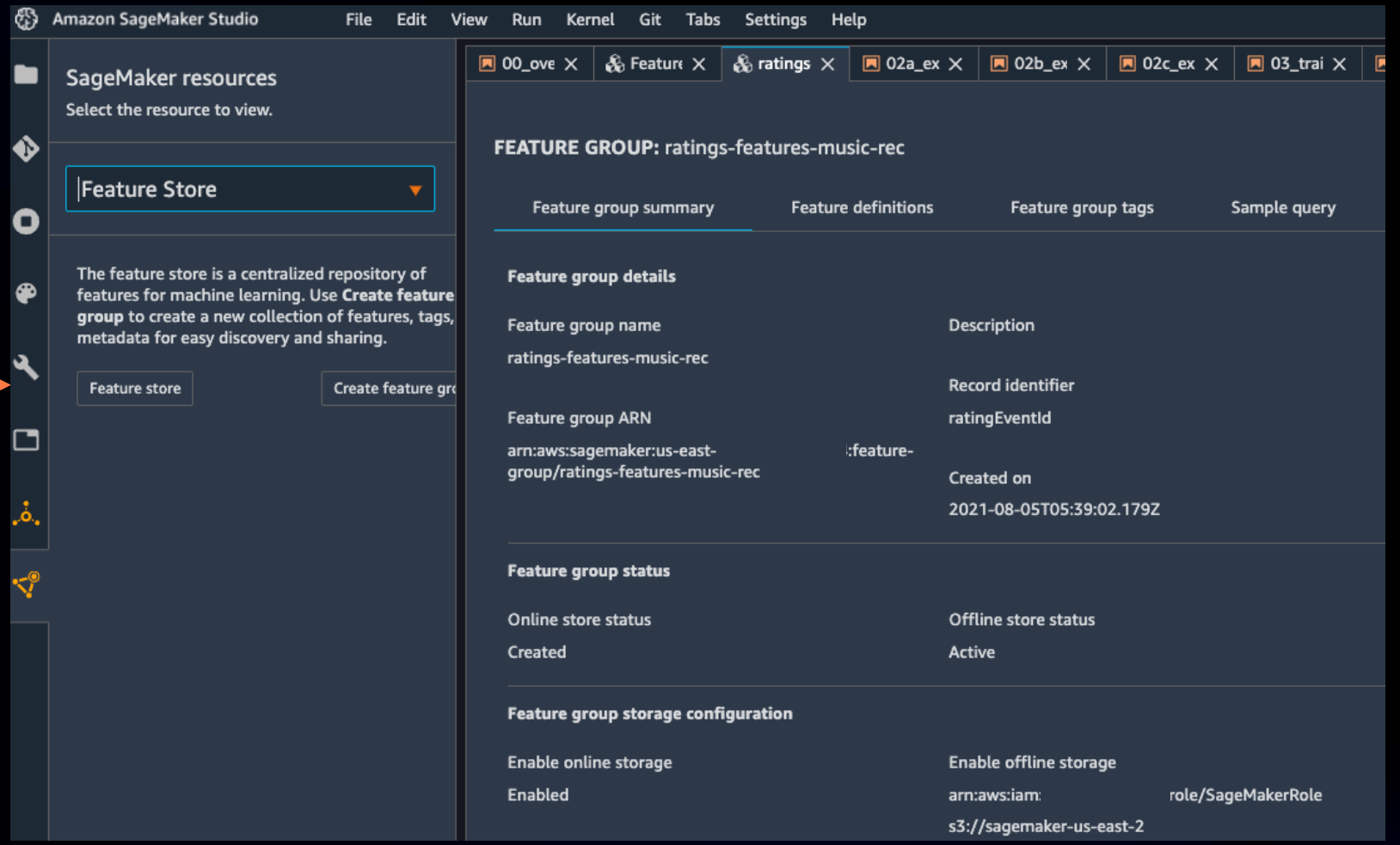
We already uploaded our flow file in the `02a` notebook. Here the Data Wrangler Flow is also provided to the Processing Job as an input source which we configure below.

```
]: ## Input - Flow: 01_music_dataprep.flow
flow_input = ProcessingInput(
    source=flow_s3_uri,
    destination="/opt/ml/processing/flow",
    input_name="flow",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Feature Store – View from SageMaker Studio

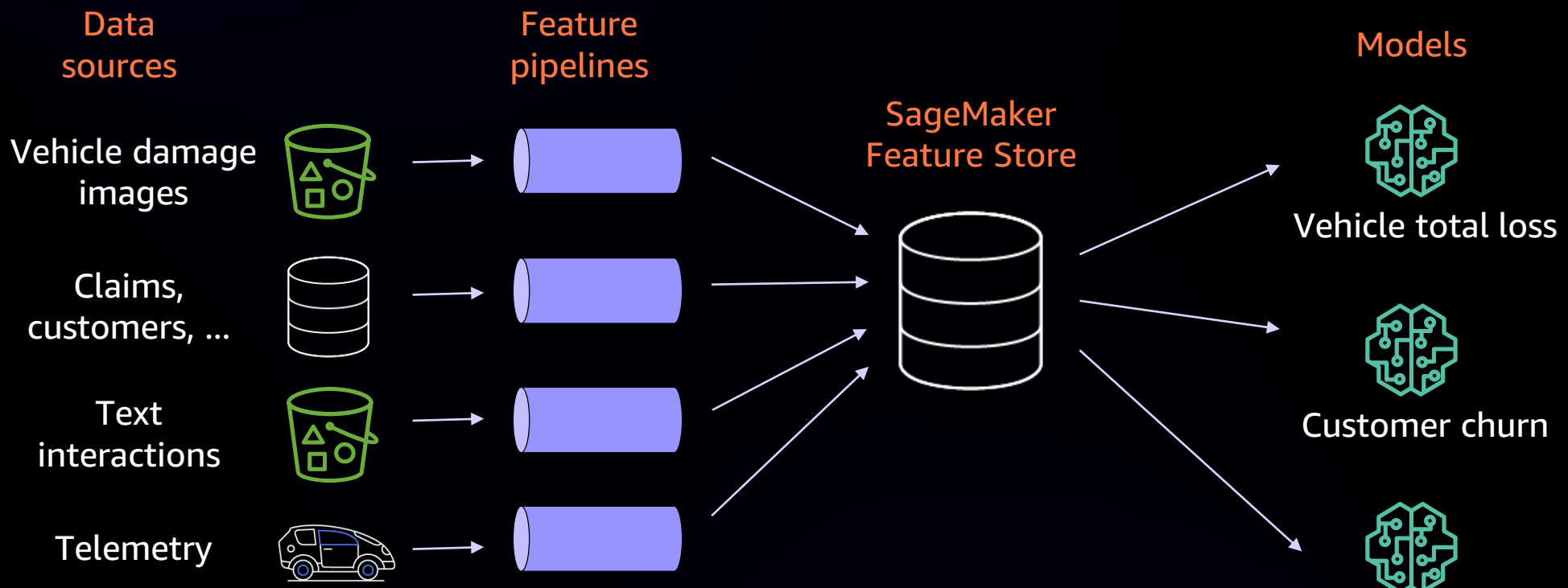




© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

With SageMaker Feature Store...

Build features once, reuse them across teams and models



- Feature groups discoverable via search
- Reproducible feature transformations
- Extract accurate training datasets
- Low latency lookups for inference
- Consistent features for training and inference

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon SageMaker Feature Store

Securely store, discover, and share features for both real-time inference and training



Batch and streaming ingestion

High throughput writes for ingesting features



Online and offline features

Online features for real-time prediction, and offline features for historical data for model training and batch prediction



Feature metadata and data cataloging

Store metadata for features and leverage automatic data cataloging to easily query and extract feature data



Feature discovery and reuse

Search for features to reuse, before starting new development



Security and access control

Access control for feature data and feature metadata, and support for encryption at rest, Amazon VPC, and AWS PrivateLink



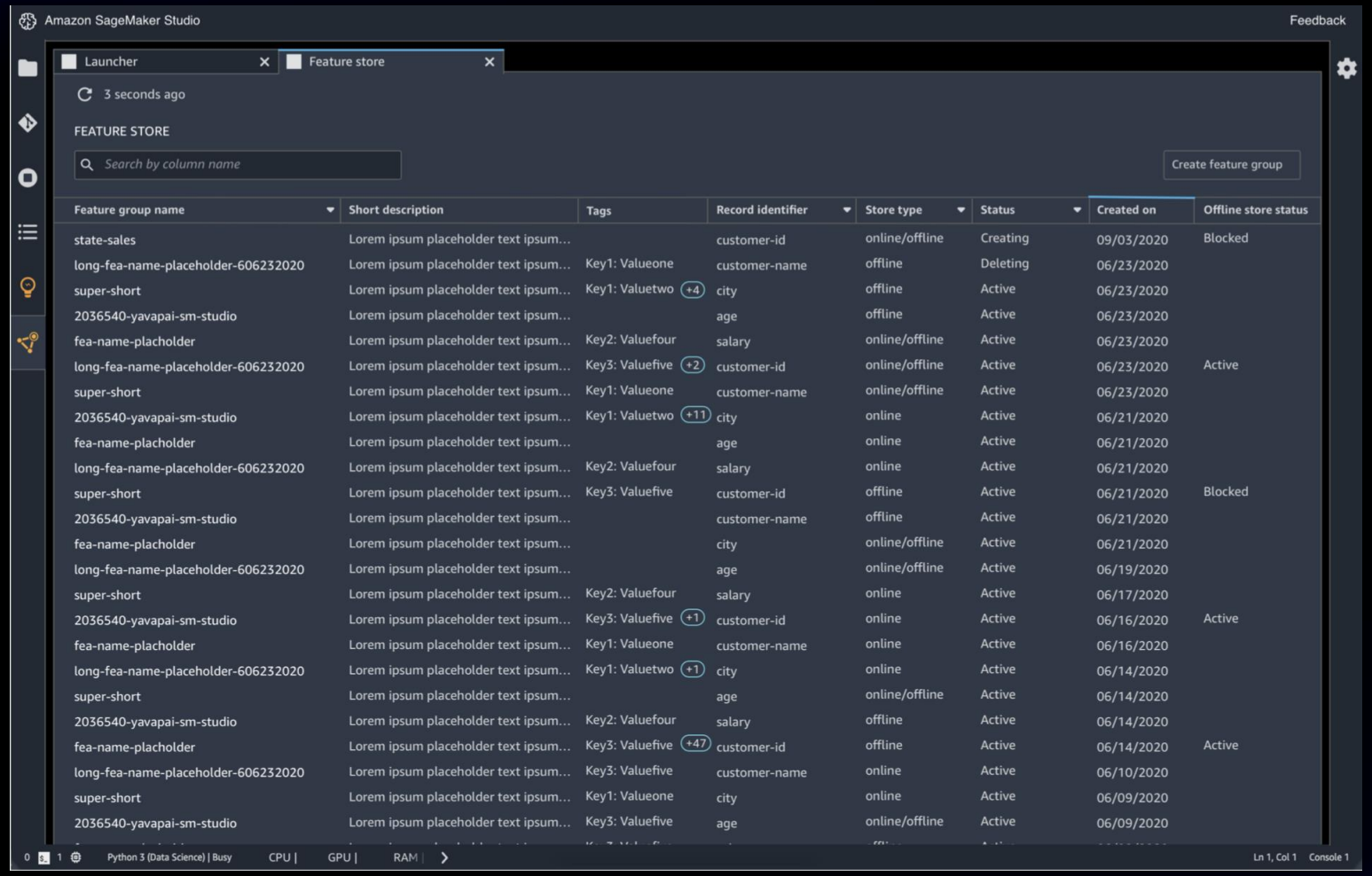
Fully managed

Online features cached in low-latency store; maintain consistency between online and offline store to avoid train-infer skew

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

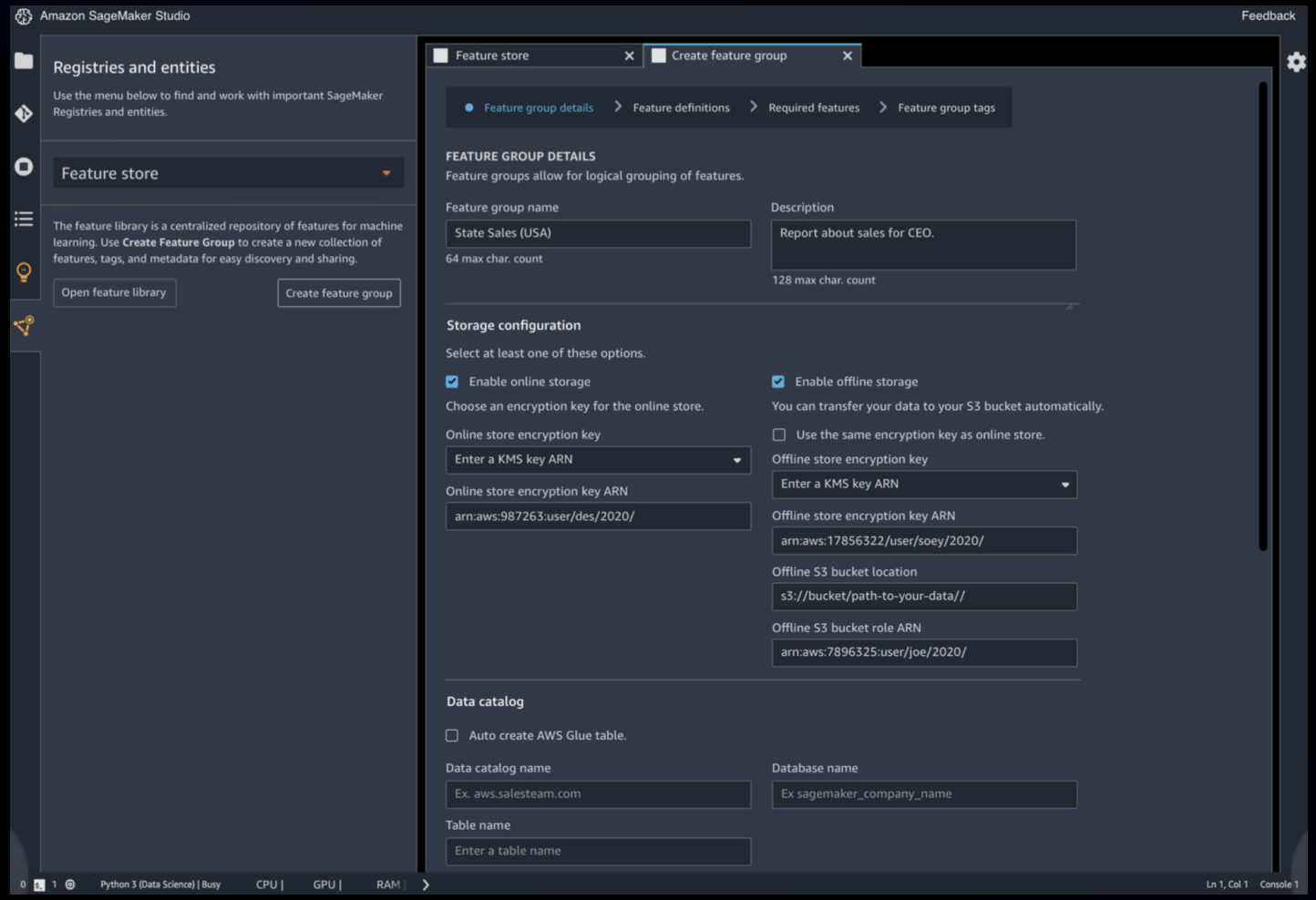
Search and discover feature groups



- Discover feature groups by name, description, tags, and other metadata

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

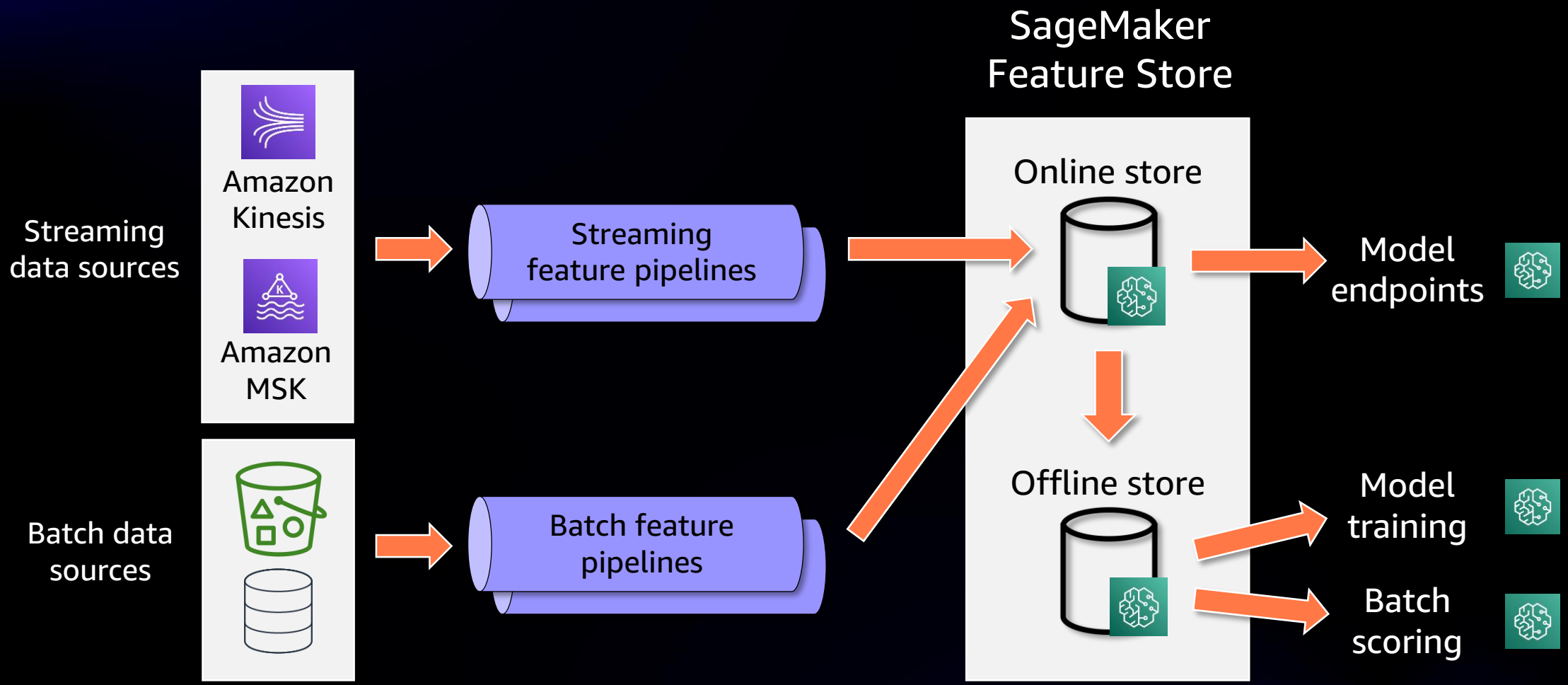
Manage features using Feature Groups



- Store features in collections called Feature Groups
- Configure Feature Groups for online and/or offline storage
- Create data catalog for Feature Groups
- Manage comprehensive metadata using Feature Group tags

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Feature store in context

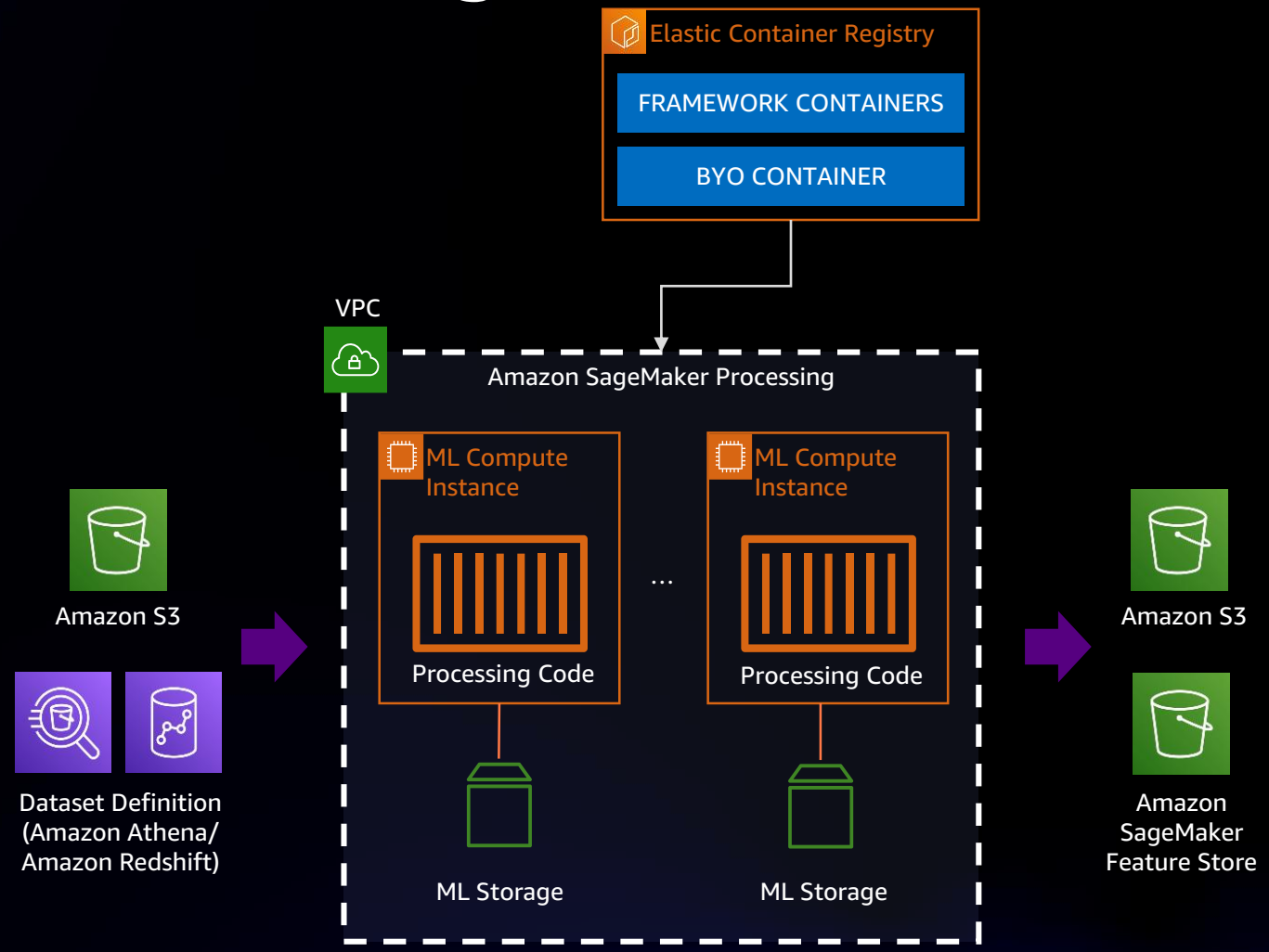


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon SageMaker Processing

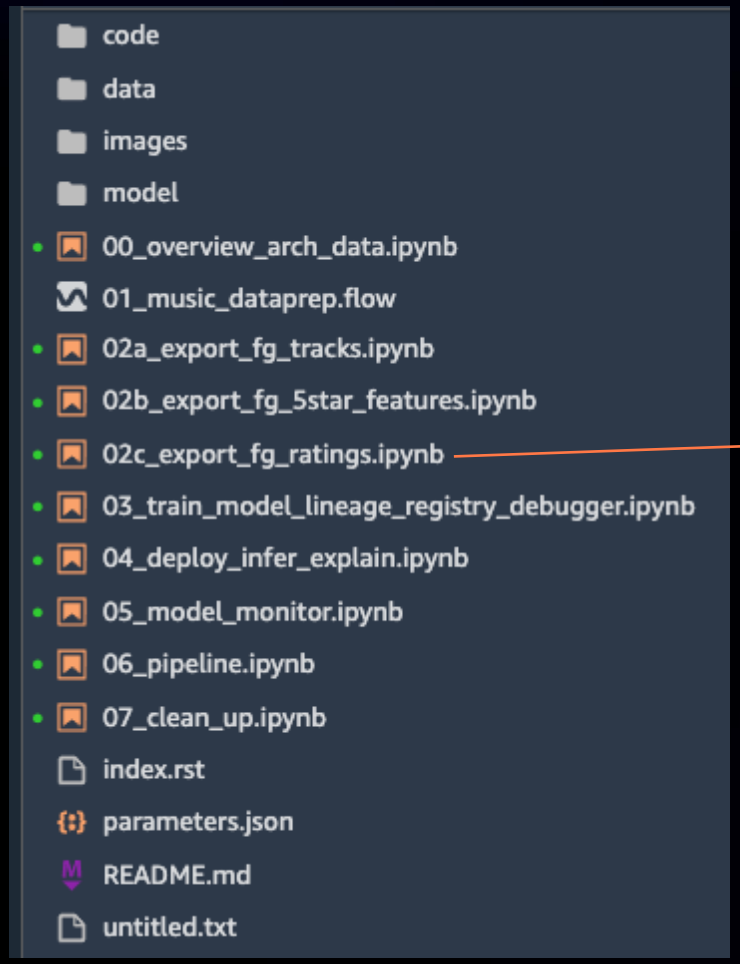
```
CreateProcessingJob()

{
   "AppSpecification": {
      "ContainerArguments": [ "string" ],
      "ContainerEntrypoint": [ "string" ],
      "ImageUri": "string"
   },
   "Environment": {
      "string" : "string"
   },
   "ExperimentConfig": ... ,
   "NetworkConfig": ... ,
   "ProcessingInputs": ... ,
   "ProcessingJobName": ... ,
   "ProcessingOutputConfig": ... ,
   },
   "ProcessingResources": {
      "ClusterConfig": {
         ...
      }
   },
   "RoleArn": ... ,
   "StoppingCondition": ... ,
   "Tags": ... ,
   ]
}
```



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation – Data prep: Set up the processing job



Create Processing Job

To launch a Processing Job, you will use the SageMaker Python SDK to create a Processor function.

```
from sagemaker.processing import Processor
from sagemaker.network import NetworkConfig
import json

processor = Processor(
    role=iam_role,
    image_uri=container_uri,
    instance_count=instance_count,
    instance_type=instance_type,
    volume_size_in_gb=volume_size_in_gb,
    network_config=NetworkConfig(enable_network_isolation=enable_network_isolation),
    sagemaker_session=sess
)
```

Job Status & S3 Output Location

Below you wait for processing job to finish. If it finishes successfully, your feature group should be populated with transformed feature values. In addition the raw parameters used by the Processing Job will be printed.

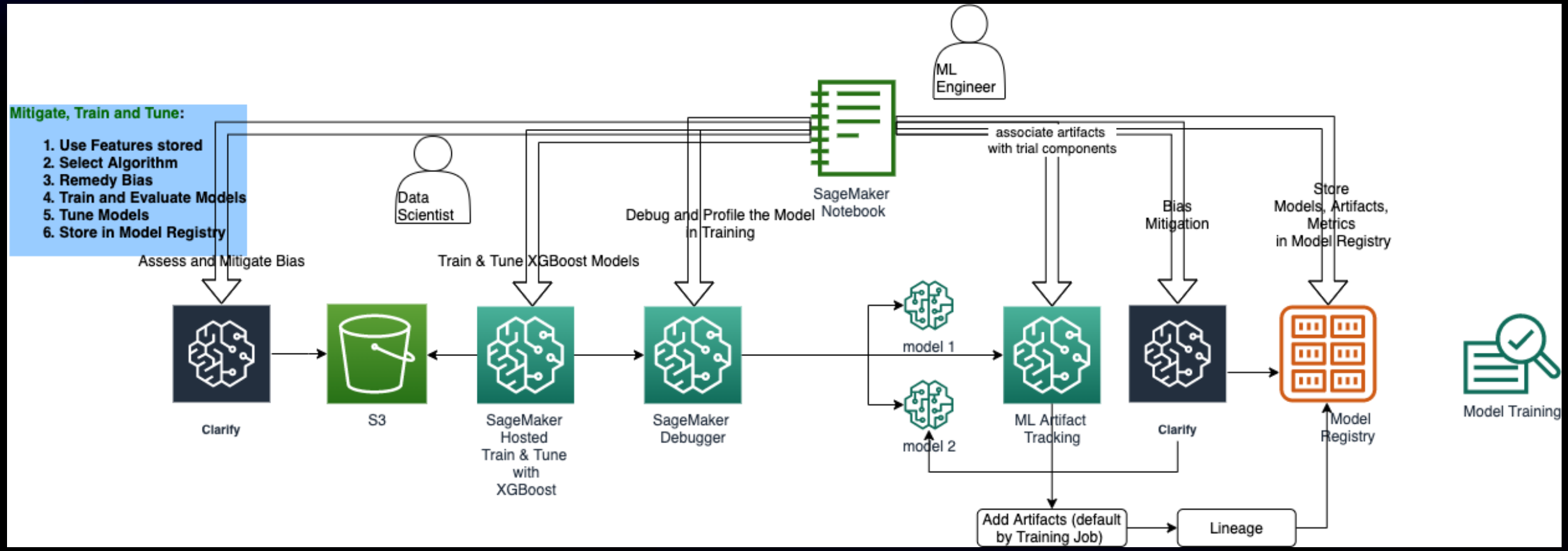
```
%%time

# Run Processing Job if job not already previously ran
if feature_group_exists:
    print("Feature Group {0} already exists therefore we will not run a processing job to create it ag
else:
    print("Creating Processing Job: {}".format(feature_group_name))
    processor.run(
        inputs=[flow_input] + data_sources,
        outputs=[processing_job_output],
        arguments=[f"--output-config '{json.dumps(output_config)}'"],
        wait=False,
        logs=False,
        job_name=processing_job_name
    )

    job_result = sess.wait_for_processing_job(processing_job_name)
```

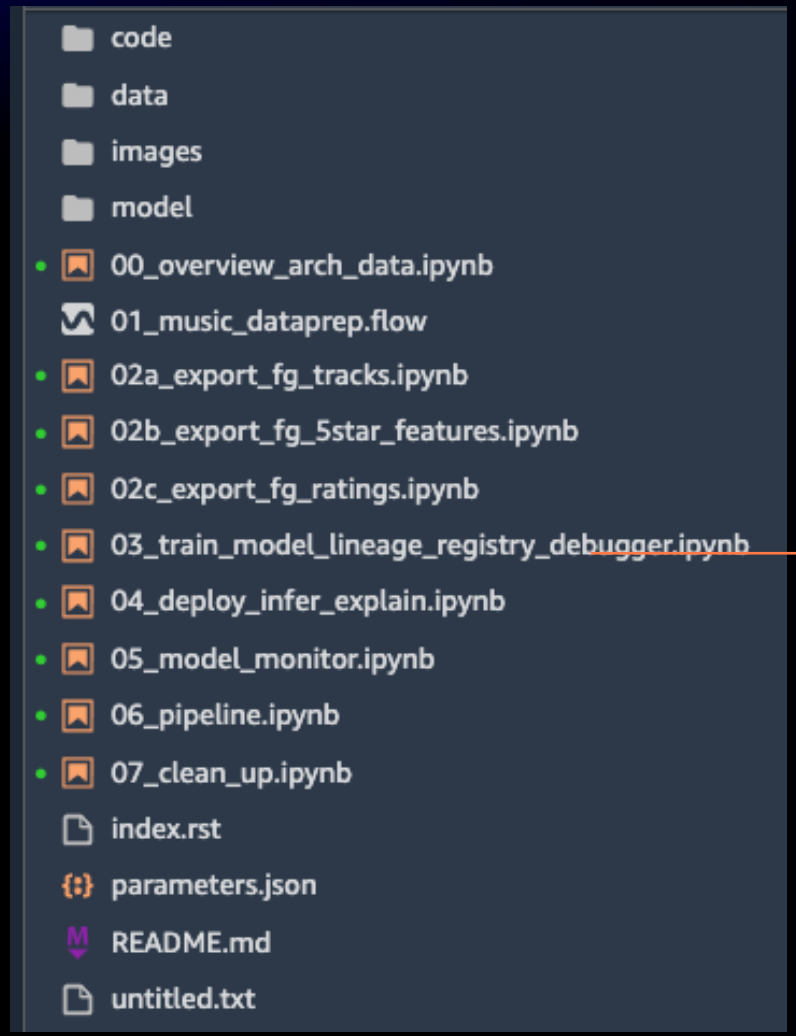
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Train, Lineage, and Tune



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Lineage for Data, Algo, and Models




```
data_artifact = artifact.Artifact.create(
    artifact_name=channel,
    source_uri=data_s3_uri,
    artifact_type='Dataset',
    sagemaker_session=sagemaker_session)
print(f'Create artifact {data_artifact.artifact_arn}: SUCCESSFUL')
```

```
model_artifact = artifact.Artifact.create(
    artifact_name='TrainedModel',
    source_uri=trained_model_s3_uri,
    artifact_type='Model',
    sagemaker_session=sagemaker_session)
```

```
artifact_list = data_artifact_list + [model_artifact]

for artif in artifact_list:
    if artif.artifact_type == 'Dataset':
        assoc = 'ContributedTo'
    else:
        assoc = 'Produced'
    try:
        association.Association.create(
            source_arn=artif.artifact_arn,
            destination_arn=trial_component_arn,
            association_type=assoc,
            sagemaker_session=sagemaker_session)
        print(f"Association with {artif.artifact_type}: SUCCESSFUL")
    except:
        print(f"Association already exists with {artif.artifact_type}")
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

Examine Lineage

back to top

Though you already know the training job details from
Lineage to produce the training job details which prod

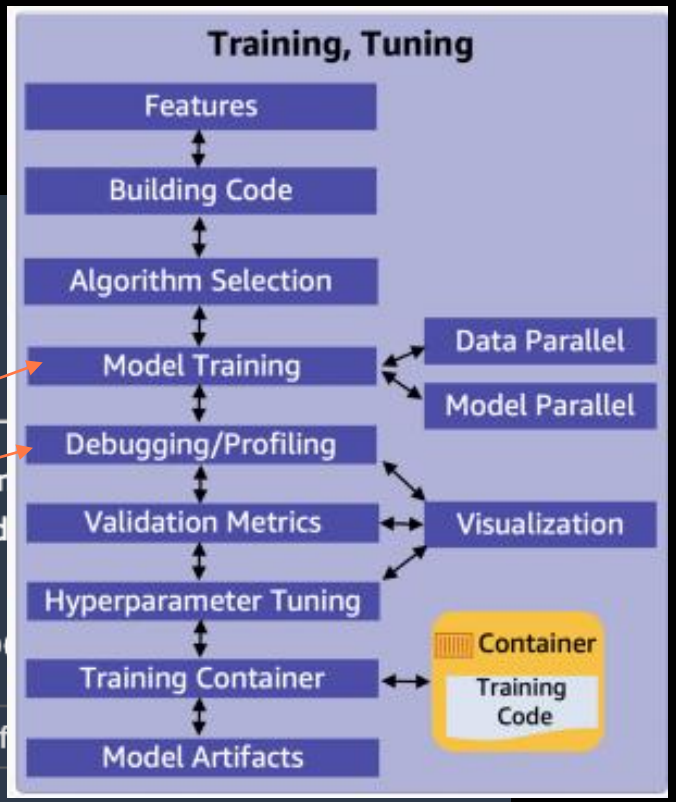
Data Lineage and Metrics for Best Mo

[33]:
```
from sagemaker.lineage import context, artif
```

Training data artifact

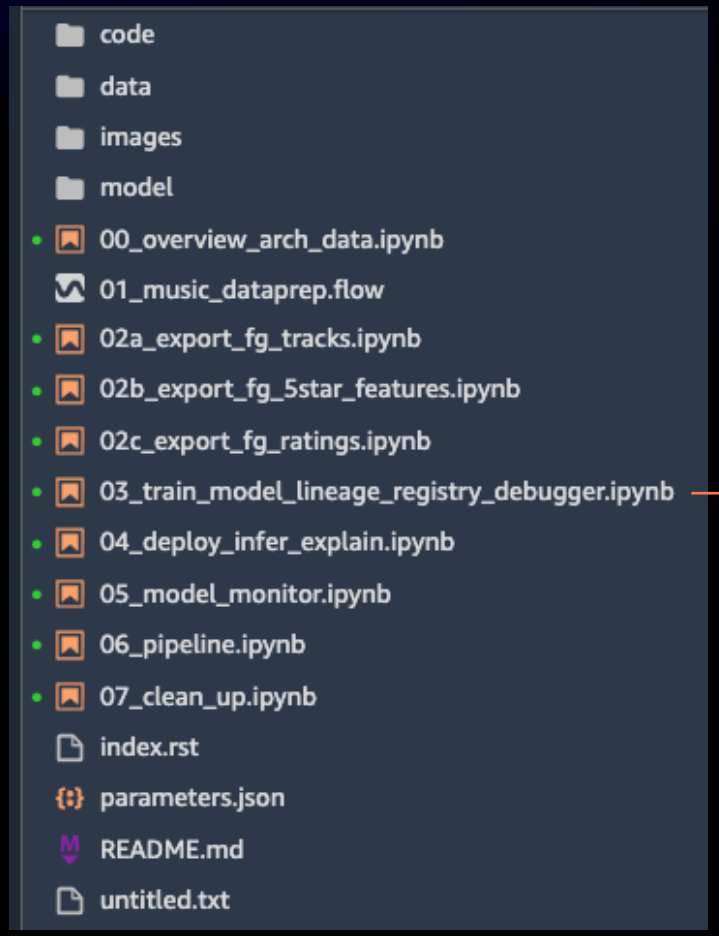
[34]:
```
data_artifact_list = []
for data_input in training_job_info['InputDataConfig']:
    channel = data_input['ChannelName']
    data_s3_uri = data_input['DataSource']['S3DataSource']['S3Uri']

    matching_artifacts = list(artifact.Artifact.list(
        source_uri=data_s3_uri,
        sagemaker_session=sagemaker_session)
    )
```



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training, get data from Feature Store



- Part 3: Train Model with Debugger Hooks. Set Artifacts and Register Model.
  - Fetch Data from Feature Store
  - Split Data and Save to S3
  - Train Model
  - SageMaker Debugger Reports

```
def get_train_val():
    for k, v in tables.items():
        query = v['feature_group'].athena_query()
        joined_cols = ", ".join(v['cols'])
        # limit number of datapoints for training time
        query_string = "SELECT {} FROM \"{}\" LIMIT 500000".format(joined_cols, query.table_name)
        print(query_string,'\n')

        output_location = f's3://{bucket}/{prefix}/query_results/'
        query.run(query_string=query_string, output_location=output_location)
        query.wait()

        tables[k]['df'] = query.as_dataframe()

    ratings = tables['ratings']['df']
    tracks = tables['tracks']['df']
    user_prefs = tables['user_5star_features']['df']

    print('Merging datasets...')
    print(f'Ratings: {ratings.shape}\nTracks: {tracks.shape}\nUser Prefs: {user_prefs.shape}\n')

    dataset = pd.merge(ratings, tracks, on='trackid', how='inner')
    dataset = pd.merge(dataset, user_prefs, on='userid', how='inner')
    dataset.drop_duplicates(inplace=True)
    dataset.drop(['userid', 'trackid'], axis=1, inplace=True)

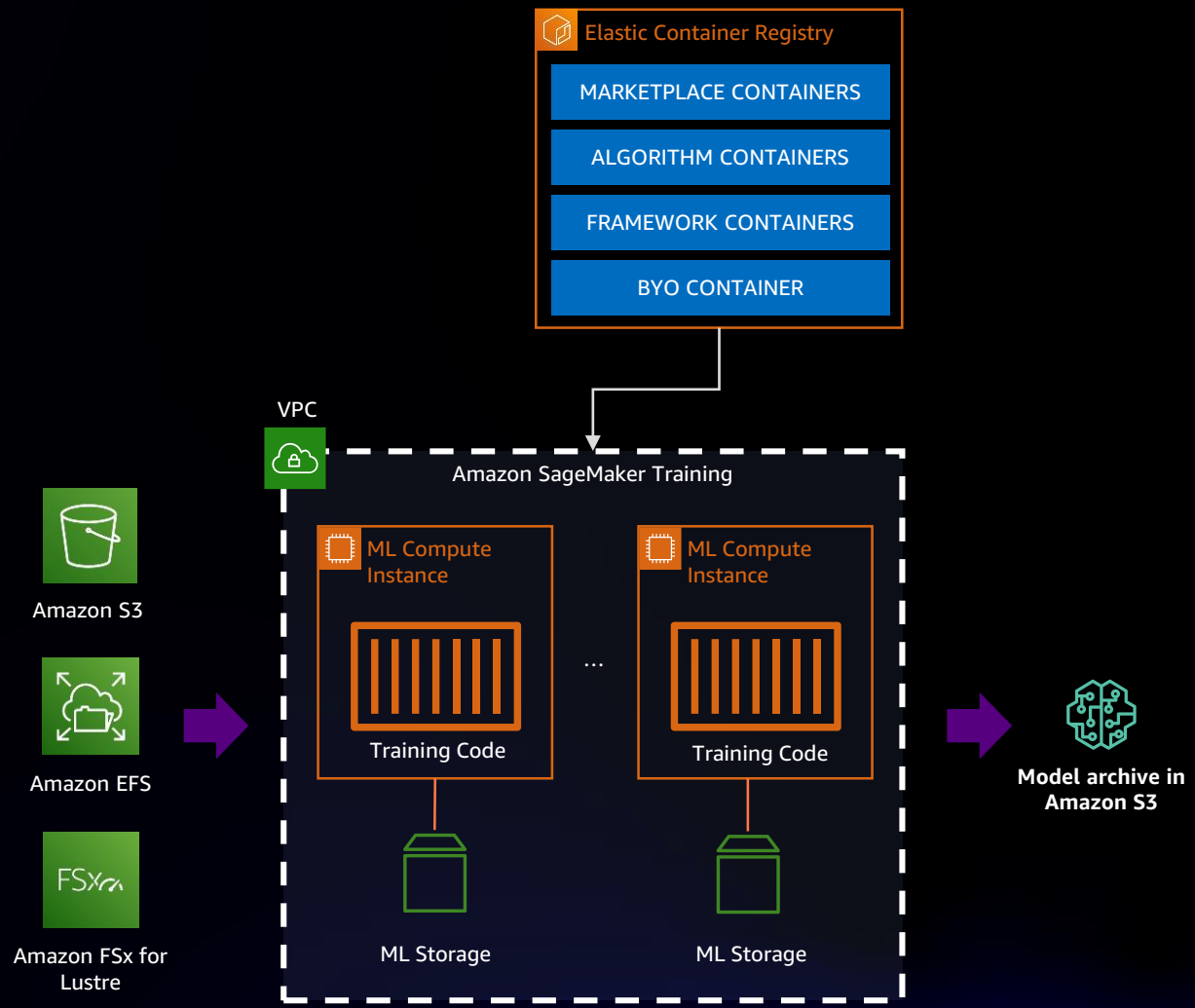
    # split data
    from sklearn.model_selection import train_test_split
    train, val = train_test_split(dataset, test_size=0.2, random_state=42)
    print("Training dataset shape: {}\nValidation dataset shape: {}\n".format(train.shape, val.shape))

    return train, val
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

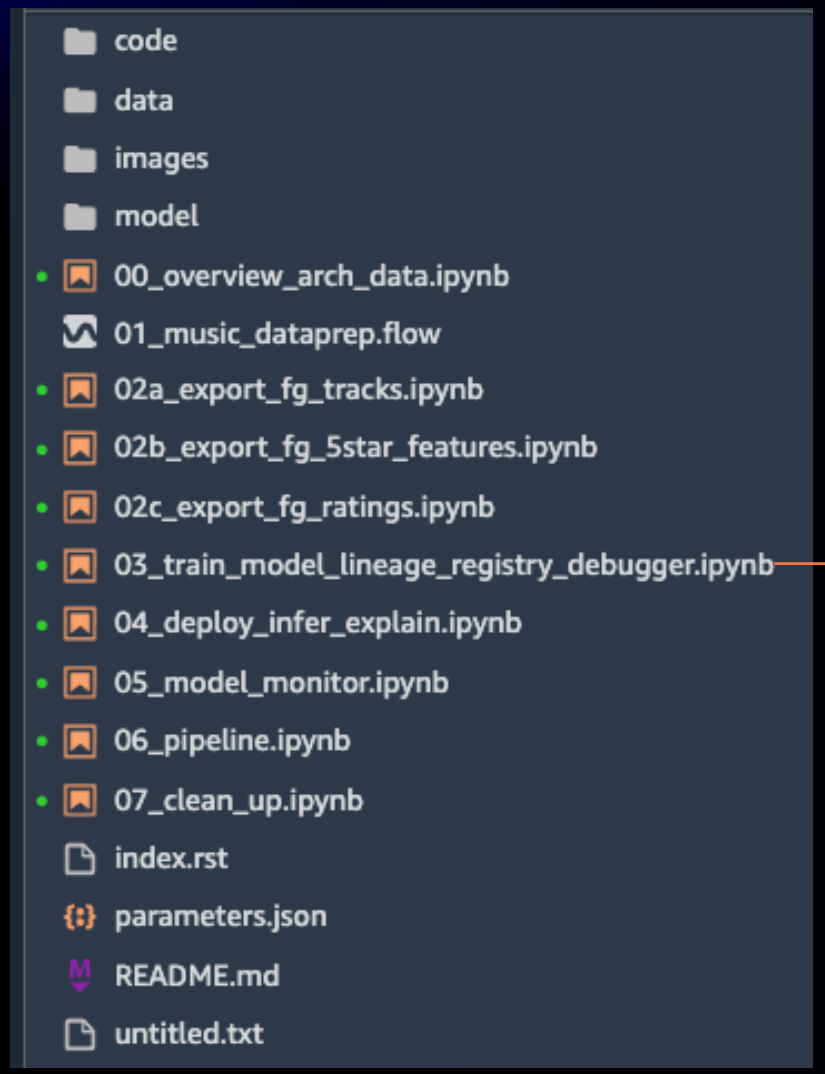
Amazon SageMaker Training

```
CreateTrainingJob()
{
   "AlgorithmSpecification": {
      "AlgorithmName": "string",
      "EnableSageMakerMetricsTimeSeries": ...,
      "MetricDefinitions": ... ,
      "TrainingImage": "string",
      "TrainingInputMode": ...
   },
   "CheckpointConfig": ...,
   "DebugHookConfig": ...,
   "DebugRuleConfigurations": ...,
   "EnableInterContainerTrafficEncryption": ...,
   "EnableManagedSpotTraining": ...,
   "EnableNetworkIsolation": ...,
   "Environment": ...,
   "ExperimentConfig": ...,
   "HyperParameters": {
      "string" : "string"
   },
   "InputDataConfig": ...,
   "OutputDataConfig": ...,
   "ProfilerConfig": ...,
   "ProfilerRuleConfigurations": ...,
   "ResourceConfig": ...,
   "RoleArn": ...,
   "StoppingCondition": ...,
   "Tags": ...,
   "TensorBoardOutputConfig": ...,
   "TrainingJobName": ...,
   "VpcConfig": ...
}
```



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training




```
xgb_estimator = Estimator(
    role=sagemaker_role,
    instance_count=train_instance_count,
    instance_type=train_instance_type,
    image_uri=image,
    hyperparameters=hyperparameters,
    base_job_name=model_name,
    output_path=estimator_output_path,

    debugger_hook_config=DebuggerHookConfig(
        s3_output_path=estimator_output_path+'/debugger',
        collection_configs=[
            CollectionConfig(
                name="metrics",
                parameters={
                    "save_interval": str(save_interval)
                }
            ),
            CollectionConfig(
                name="feature_importance",
                parameters={
                    "save_interval": str(save_interval)
                }
            ),
            CollectionConfig(
                name="full_shap",
                parameters={
                    "save_interval": str(save_interval)
                }
            ),
            CollectionConfig(
                name="average_shap",
                parameters={
                    "save_interval": str(save_interval)
                }
            ),
        ],
    ),

    rules=[
        Rule.sagemaker(
            rule_configs.loss_not_decreasing(),
```

An Estimator with Debugger Hooks and Rules

aws

ervices, Inc. or its affiliates. All rights reserved.

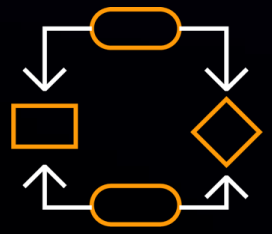
Amazon SageMaker Debugger

Analysis & debugging, explainability, and alert generation



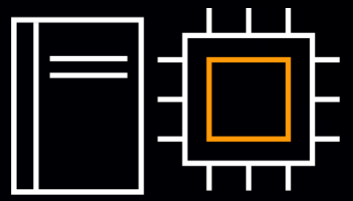
Relevant data capture

Data is automatically captured for analysis



Data analysis & debugging

Analyze & debug data with no code changes



Automatic error detection

Errors are automatically detected based on rules



Improved productivity with alerts

Take corrective action based on alerts



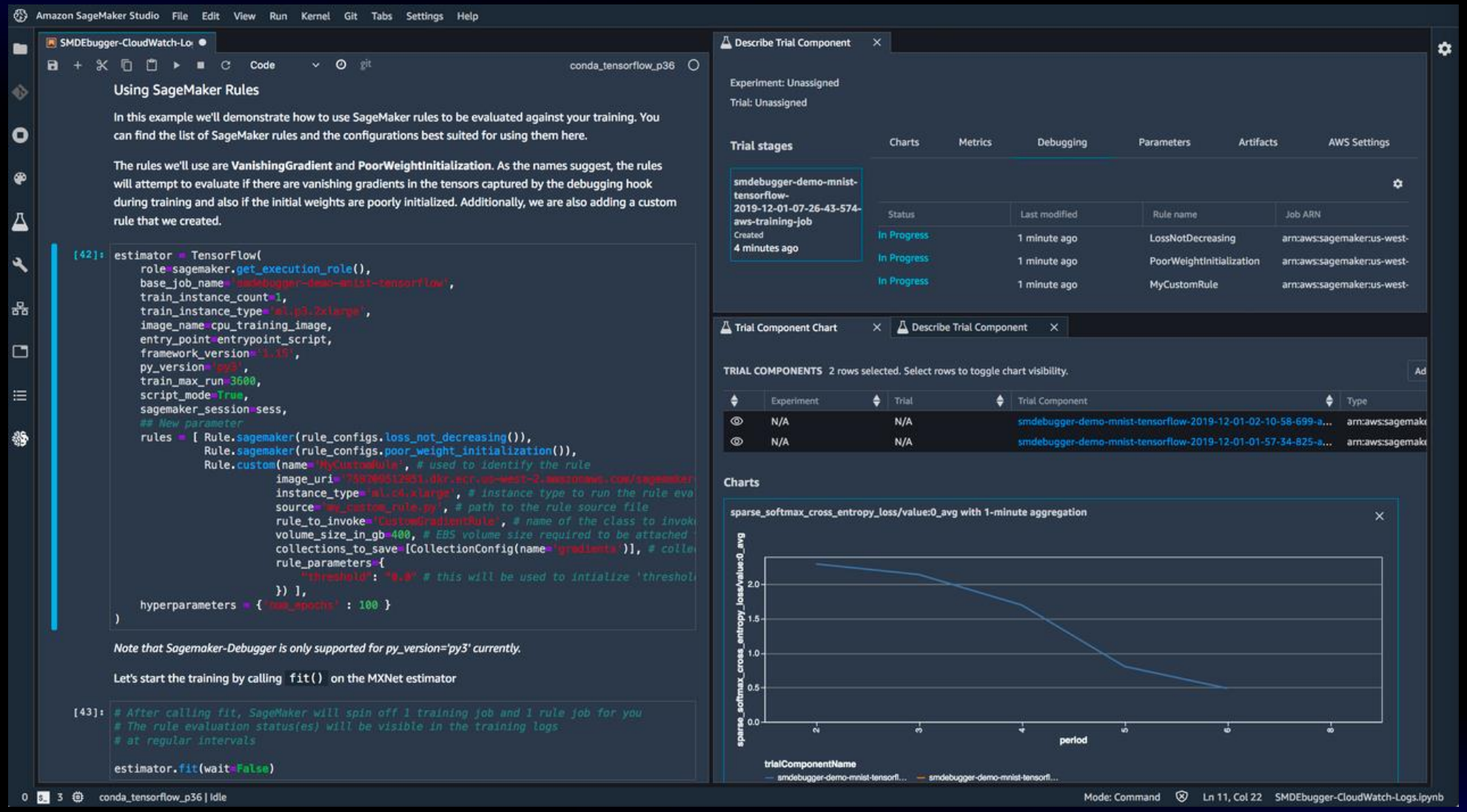
Visual analysis and debugging

Visually analyze & debug from SageMaker Studio

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

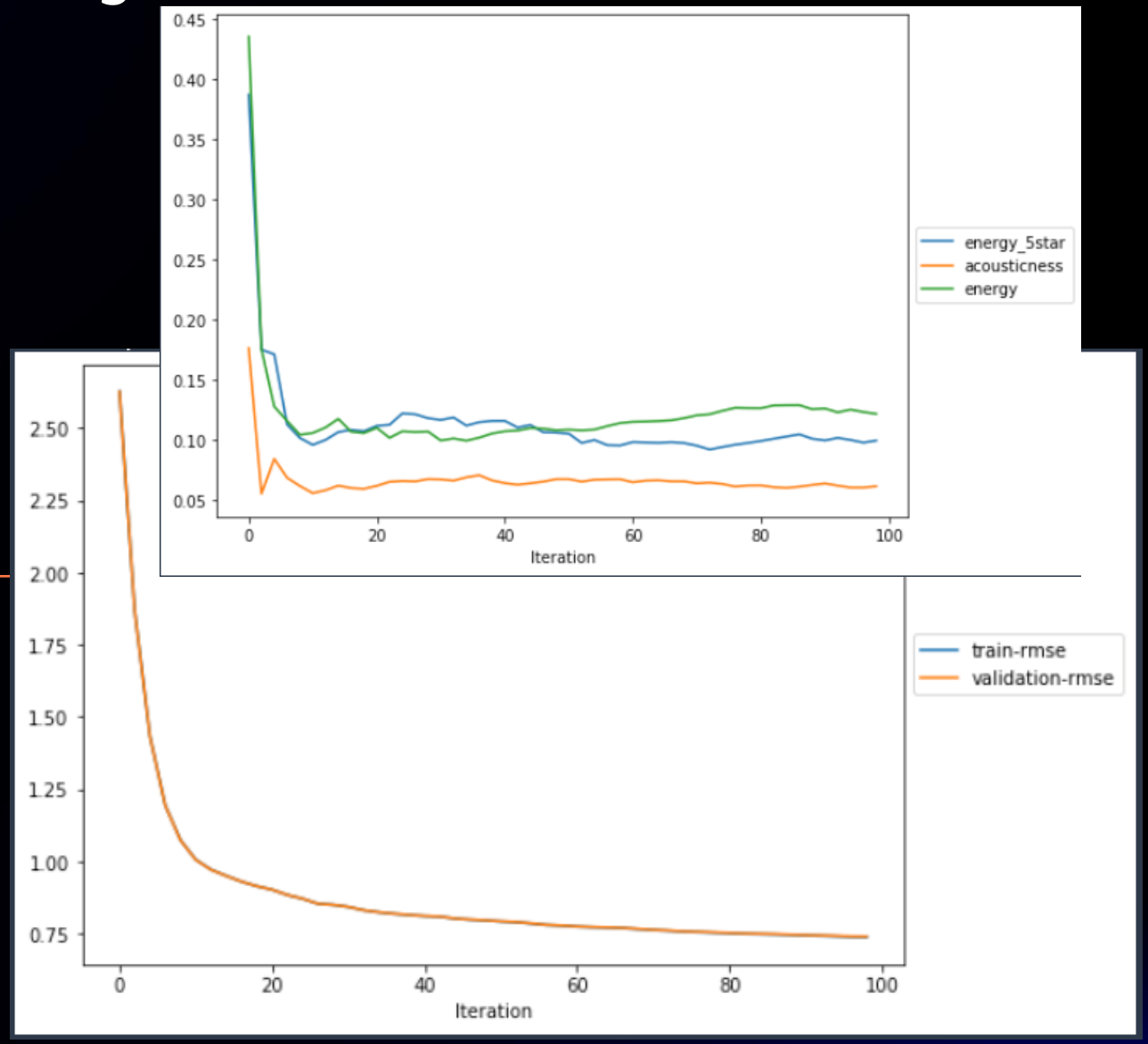
Amazon SageMaker Debugger



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training

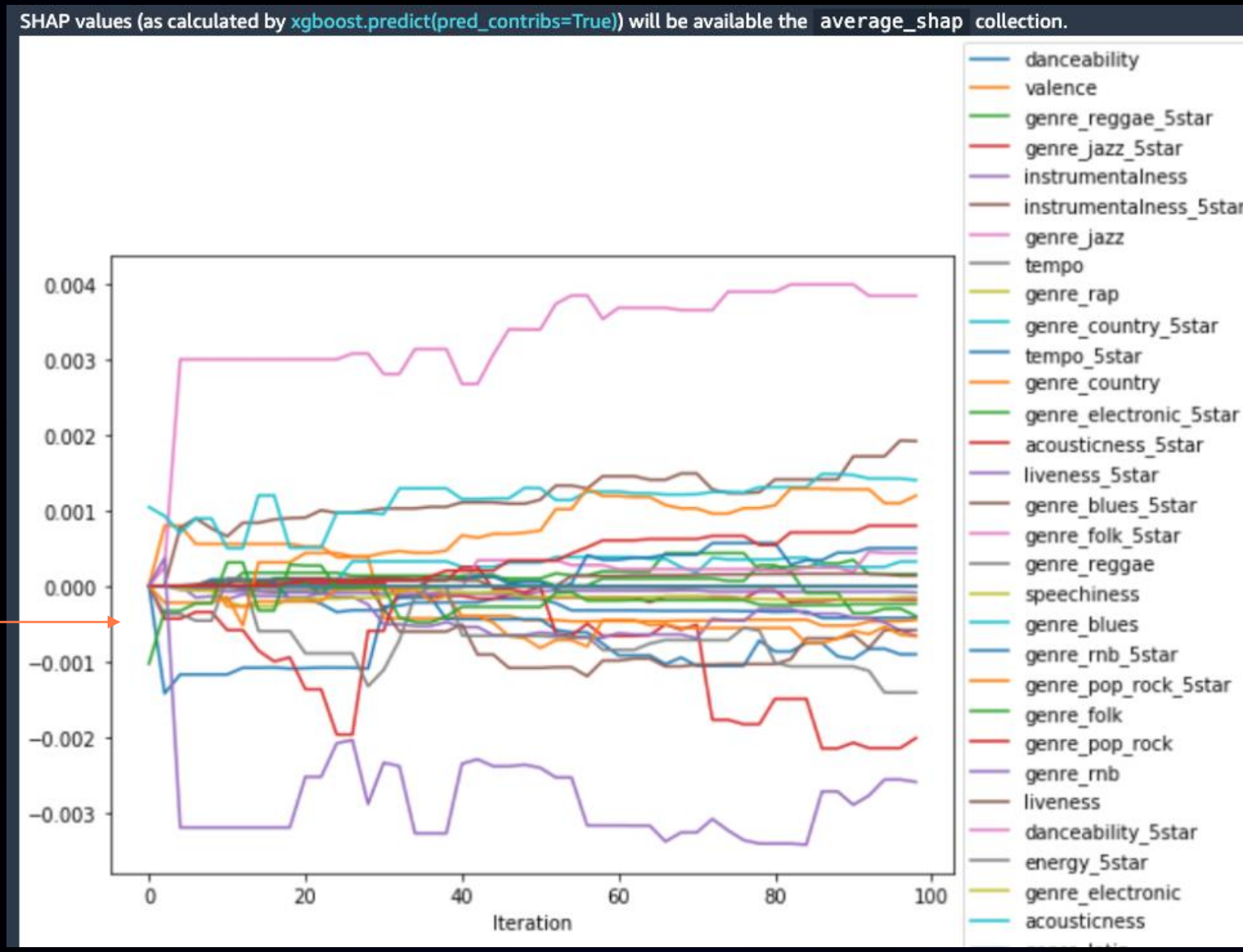
- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

```
import shap

shap_values = trial.tensor("full_shap/f0").value(trial.last_complete_step)
shap_no_base = shap_values[:, :-1]
shap_base_value = shap_values[0, -1]
shap.summary_plot(shap_no_base, plot_type='bar', feature_names=feature_names)
```

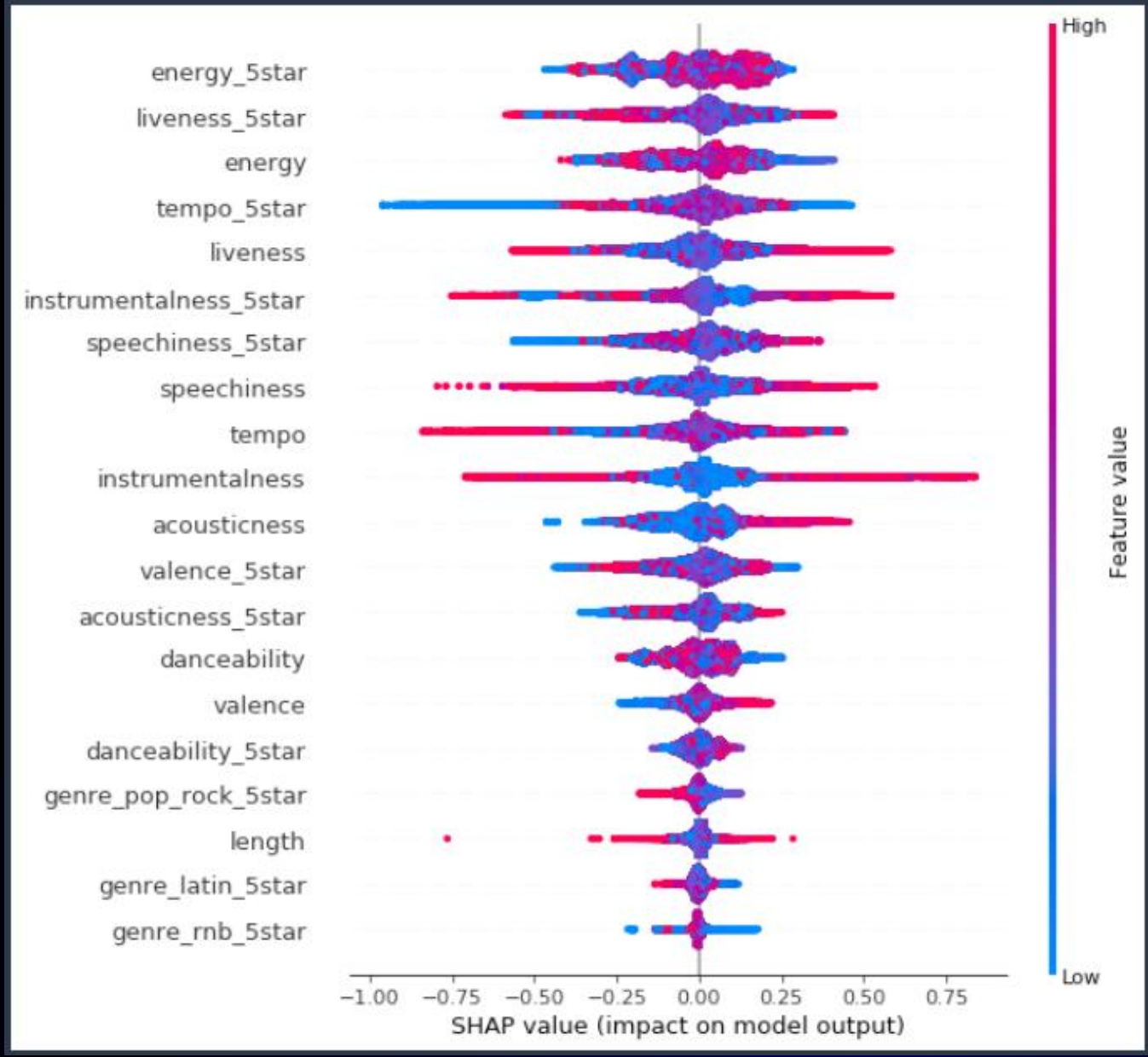


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation: Training

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

```
X_train = train.iloc[:, 1:]
shap.summary_plot(shap_no_base, X_train)
```

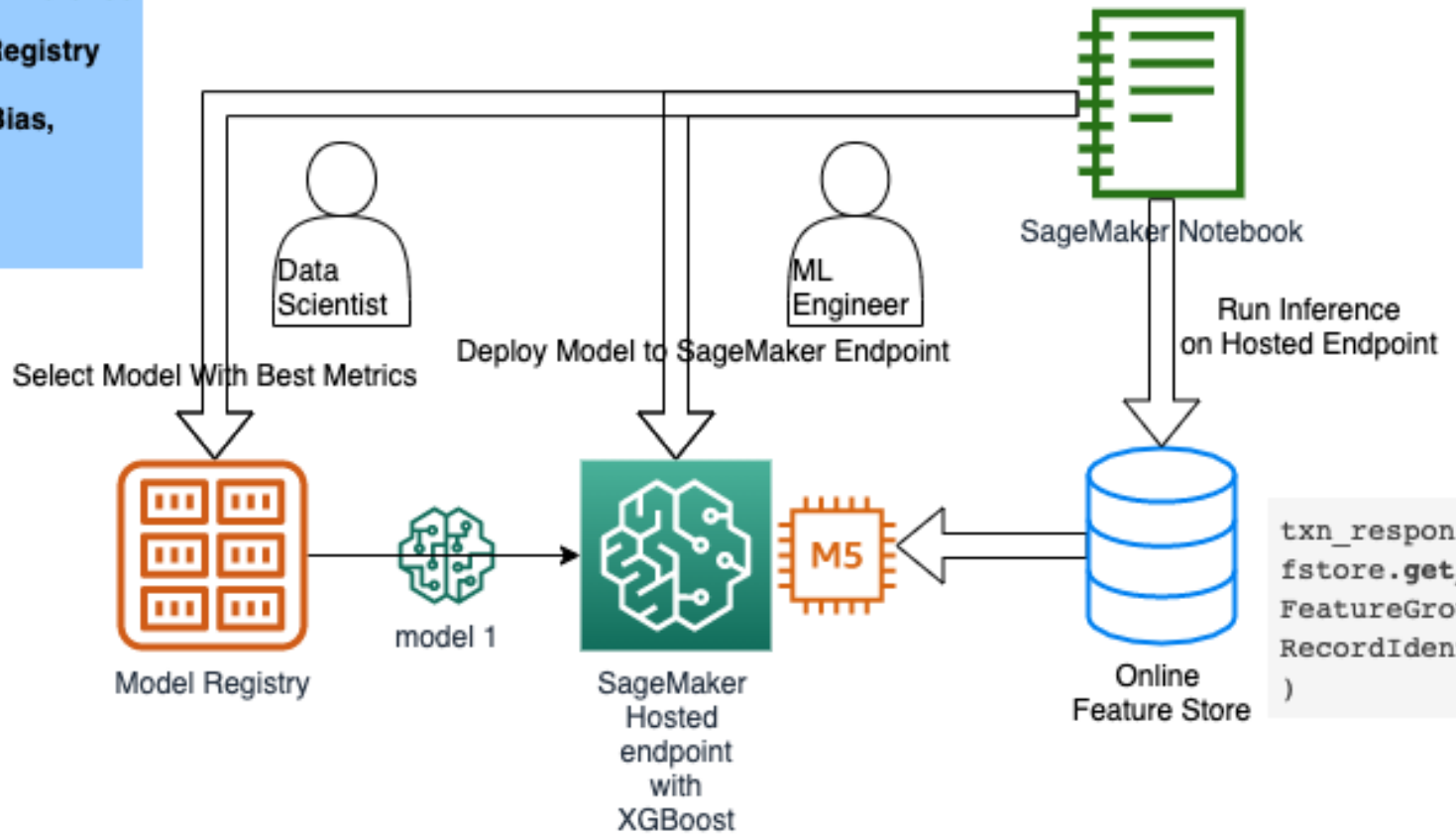


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deploy and serve

Deploy and Serve:Real-time Inference:

1. Select Models from Registry
2. Deploy Models
3. Test Inference, Test Bias,
4. Verify Explainability
5. Serve the Model,
   Real-time Inference




```
txn_response =
fstore.get_record(
FeatureGroupName=txn_fgroup_name,
RecordIdentifierValueAsString=txn_id
)
```

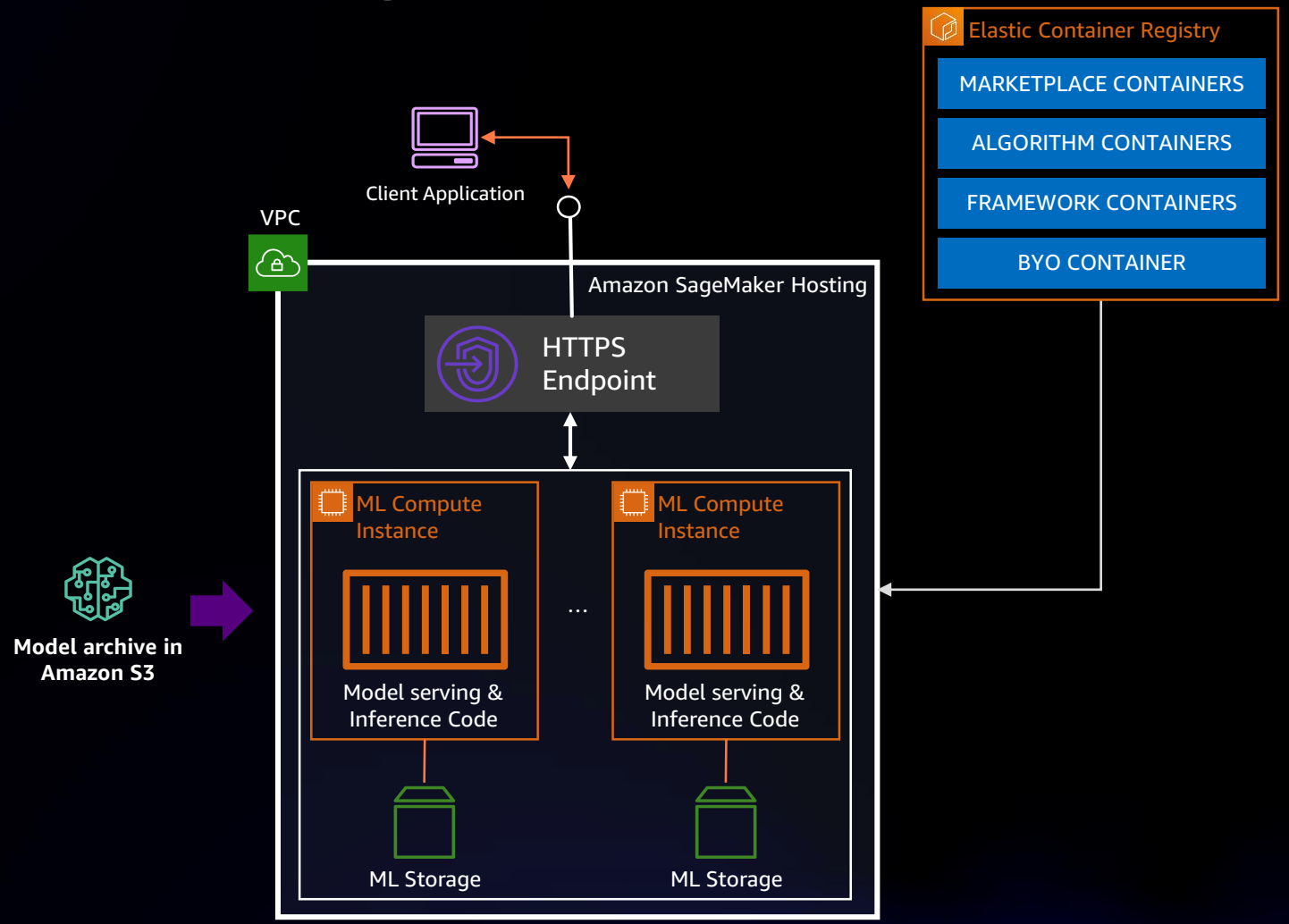
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon SageMaker Hosting

```
CreateModel()
{
   "Containers": ...,
   "EnableNetworkIsolation": ...,
   "ExecutionRoleArn": ...,
   "ModelName": ...,
   "PrimaryContainer": {
      "ContainerHostname": ...,
      "Environment": {
         "string" : "string"
      },
      "Image": "string",
      "ImageConfig": ...,
      "Mode": ...,
      "ModelDataUrl": "string",
      "ModelPackageName": "string"
   },
   "Tags": ...,
   "VpcConfig": ...
}
```

```
CreateEndpointConfig()
{
   "DataCaptureConfig": ...,
   "EndpointConfigName": ...,
   "KmsKeyId": ...,
   "ProductionVariants": ...,
   "Tags": ...
}
```

```
CreateEndpoint()
{
   "EndpointConfigName":
...,
   "EndpointName": ...,
   "Tags": ...
}
```



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Model deployment

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

- Part 4: Deploy Model & Inference using Online Feature Store
  - Deploy model
  - Create predictor
  - Infer new songs
  - Explain model predictions

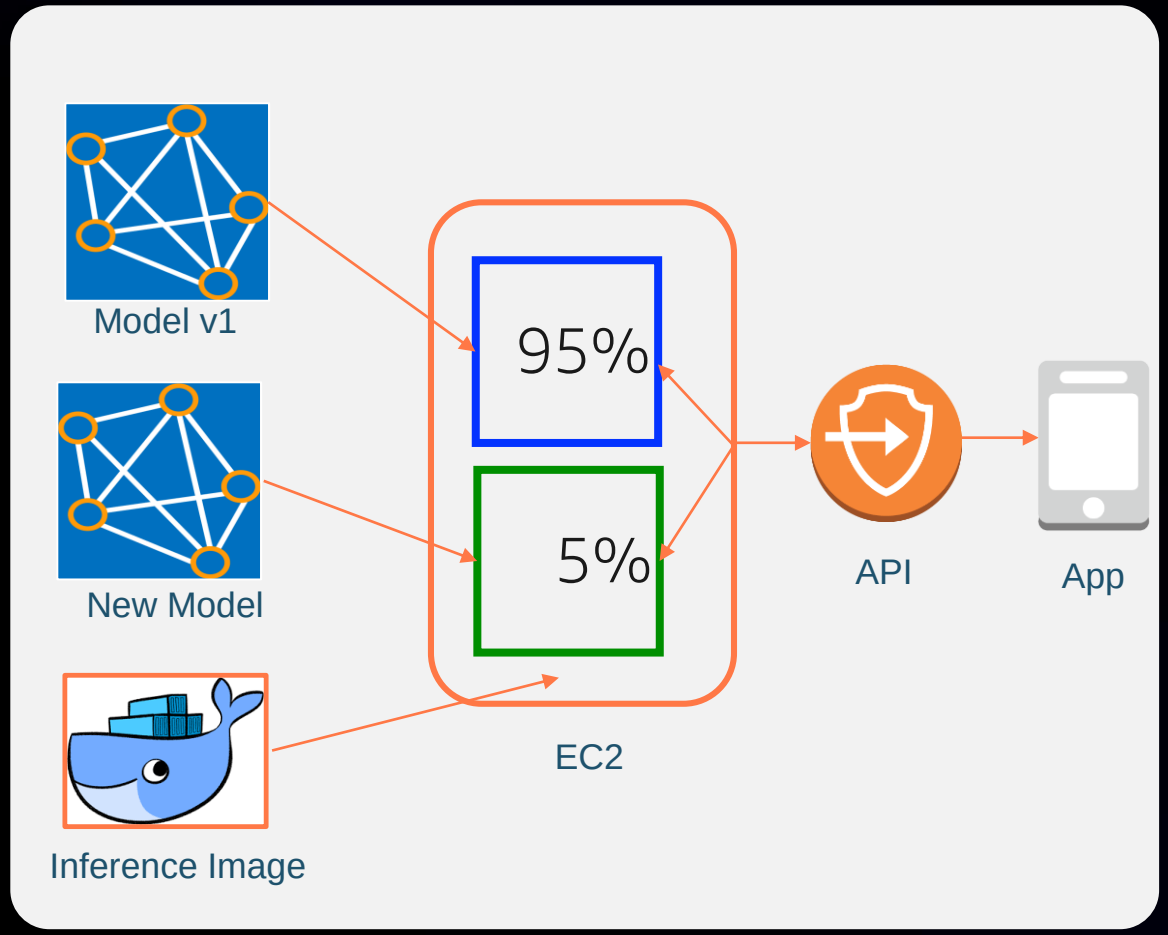
Create endpoint

```
%%time
if len(endpoint_list['Endpoints']) > 0:
    print(f"Using existing endpoint: {endpoint_list['Endpoints'][0]['EndpointName']}")
else:
    # deploy endpoint for model if it doesn't already exist
    xgb_estimator.deploy(initial_instance_count=1,
                         instance_type='ml.m4.xlarge',
                         model_name=model_name,
                         endpoint_name=endpoint_name
                         )
```

```
Using existing endpoint: music-recommendation-model-endpoint-notebooks
CPU times: user 239 µs, sys: 0 ns, total: 239 µs
Wall time: 171 µs
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

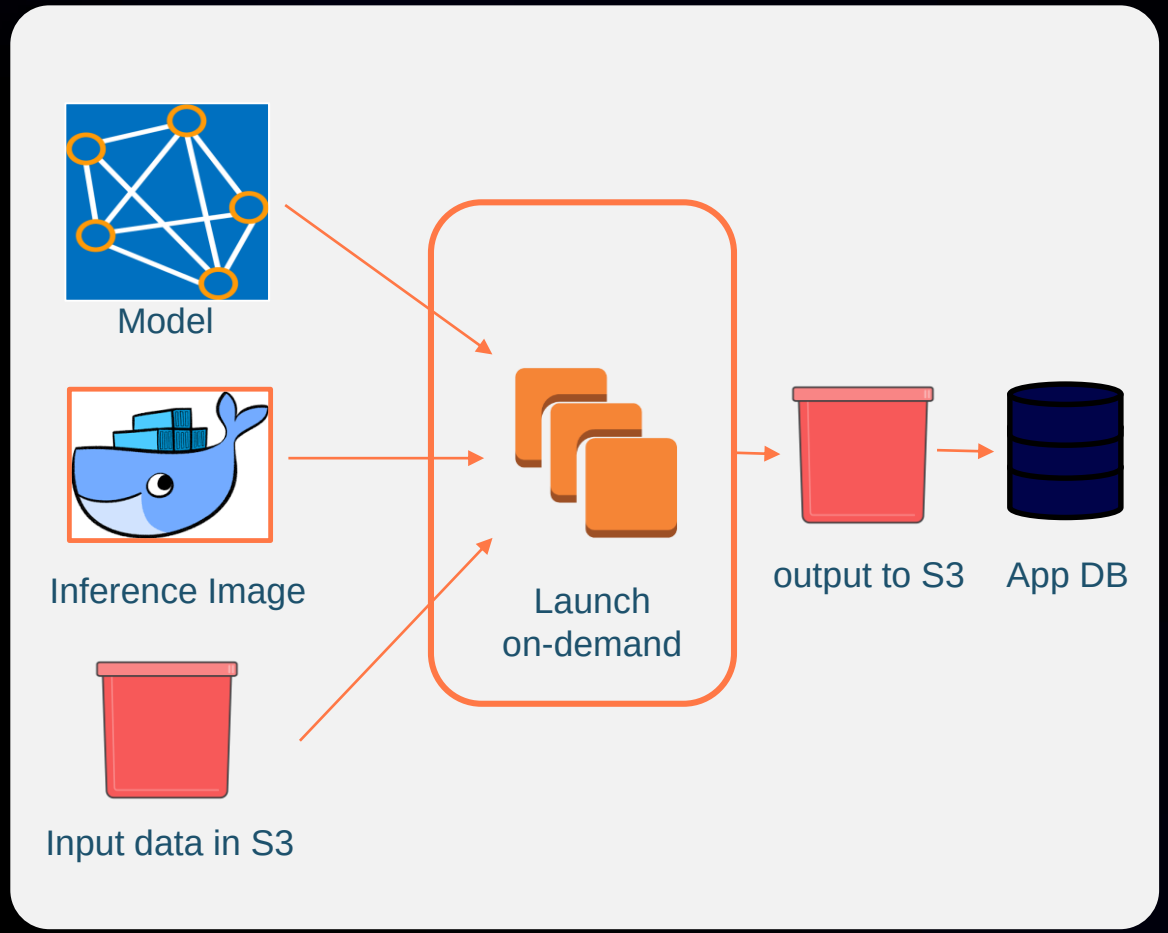
Amazon SageMaker – Real-time inference



- Quick deployment of Restful API endpoint
- Auto scaling across multiple AZs for variable loads
- A/B testing support
- VPC PrivateLink support
- Endpoint invocation and resource usage metrics
- No model re-engineering

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

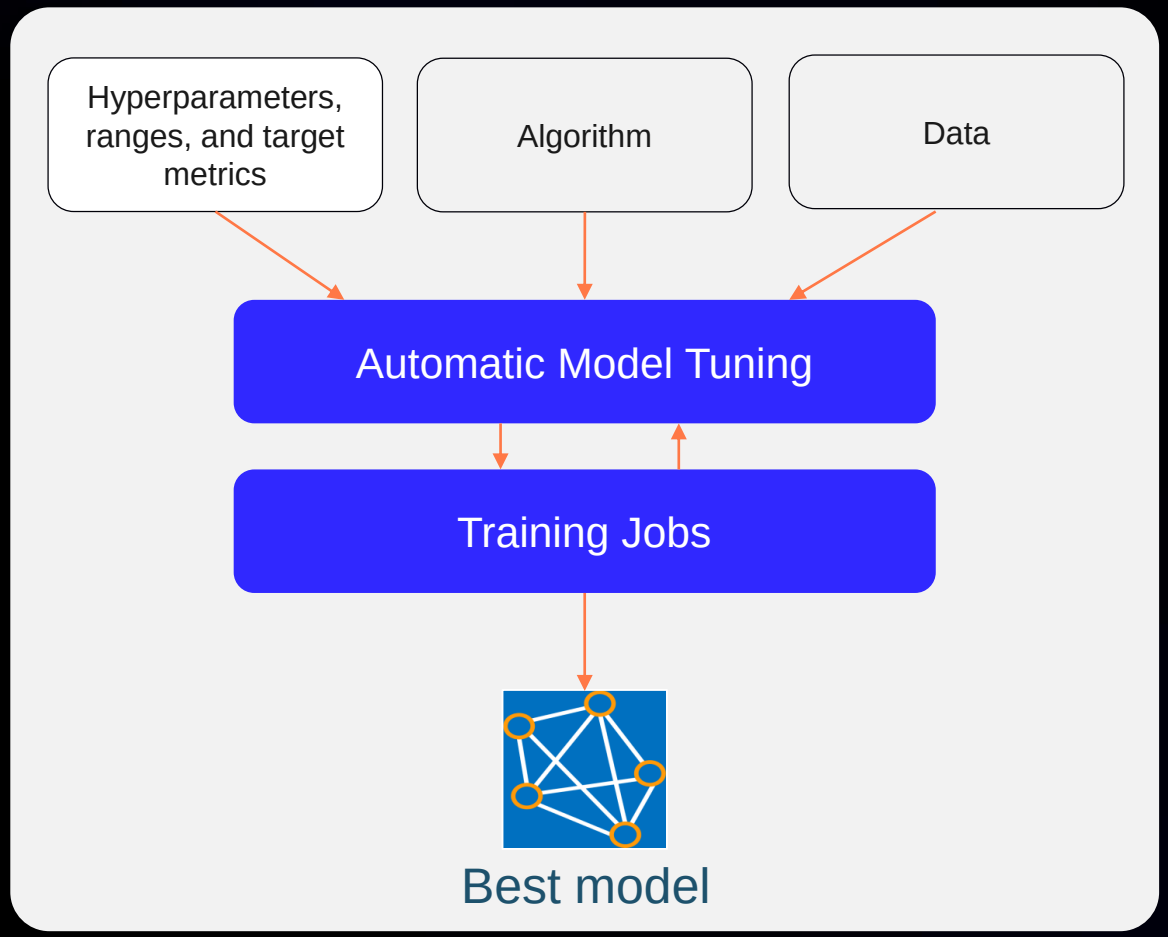
Amazon SageMaker – Batch inference



- Use trained model to get inference on data in S3
- Launch fully managed transient inference resources
- Seamless integration with S3 for both input and output

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

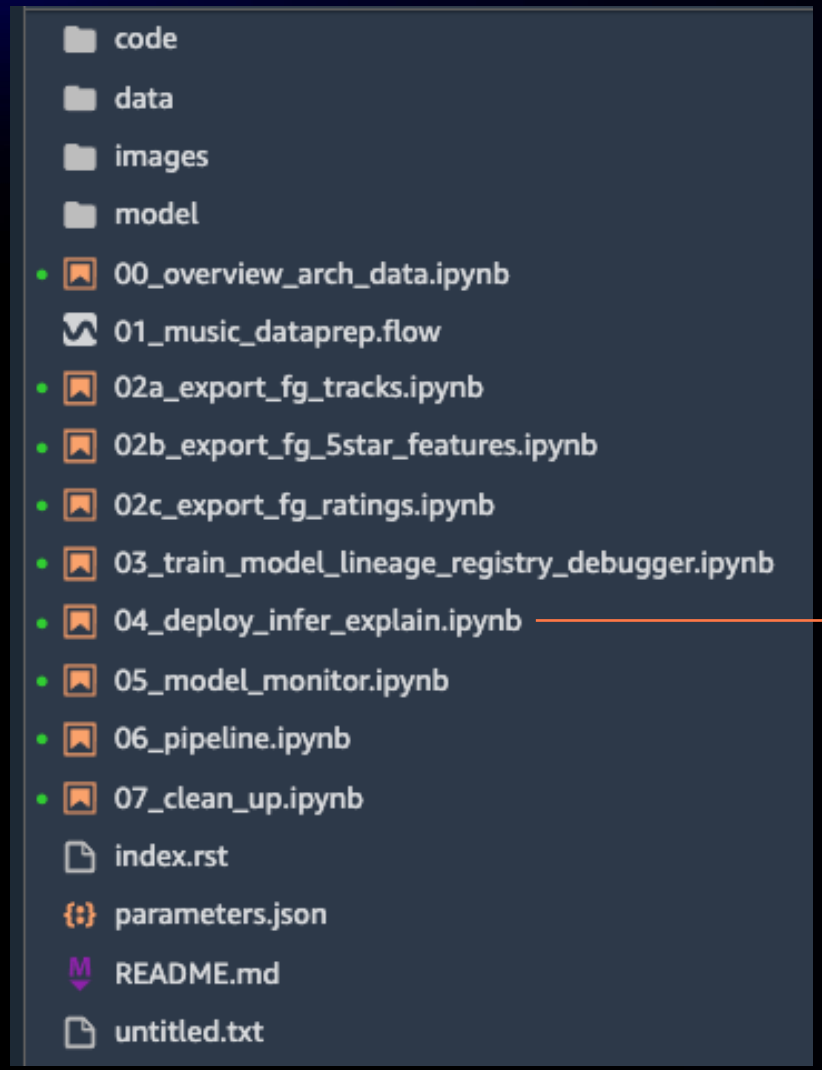
Tuning Model Hyperparameters



- Use Bayesian Optimization strategy to reduce hyperparameter search space
- Flexible control over tuning speed and model accuracy
- Cost control through max job configuration
- Support both build-in algorithms and custom algorithms

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Inference: Predictor and payload



- Part 4: Deploy Model & Inference using Online Feature Store
  - Deploy model
  - Create predictor
  - Infer new songs
  - Explain model predictions

Create a predictor

back to top

```
predictor = sagemaker.predictor.Predictor(
    endpoint_name=endpoint_name,
    sagemaker_session=sagemaker_session)
```

Pull user data from feature group

```
# random user ID. You can try any other ID
sample_user_id = 11005
```

```
featurestore_runtime = boto_session.client(service_name='sagemaker-fe

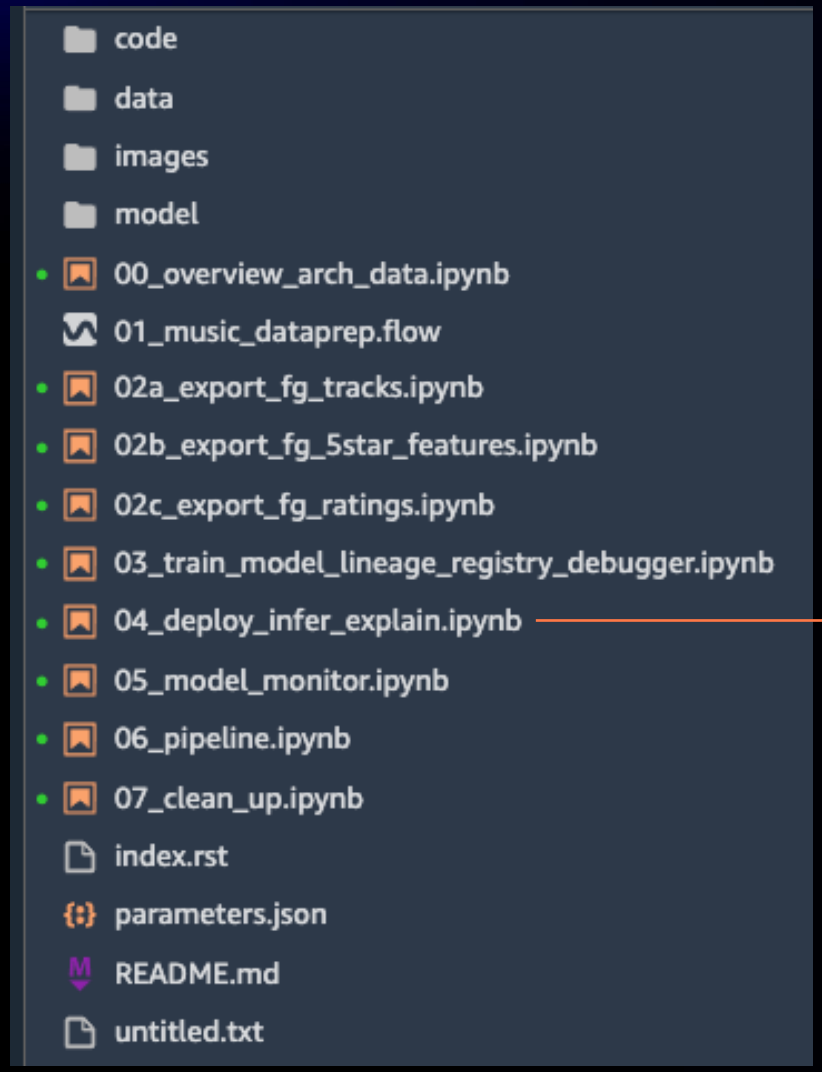
feature_store_session = sagemaker.Session(
    boto_session=boto_session,
    sagemaker_client=sagemaker_client,
    sagemaker_featurestore_runtime_client=featurestore_runtime
)
```

```
# pull the sample user's 5 star preferences record from the feature s
fg_response = featurestore_runtime.get_record(
    FeatureGroupName=fg_name_user_preferences,
    RecordIdentifierValueAsString=str(sample_user_id)
)
```

```
record = fg_response['Record']
df_user = pd.DataFrame(record).set_index('FeatureName')
```

or its affiliates. All rights reserved.

Inference: Recommend a curated playlist



- Part 4: Deploy Model & Inference using Online Feature Store
  - Deploy model
  - Create predictor
  - Infer new songs
  - Explain model predictions

```
predictions = []
for data_input in data_inputs:
    results = predictor.predict(data_input, initial_args = {"ContentType": "text/csv"})
    prediction = json.loads(results)
    predictions.append(prediction)
print(f'Predicted rating for user {int(sample_user_id)}:', prediction)
```

```
Predicted rating for user 11005: 4.857813358306885
```

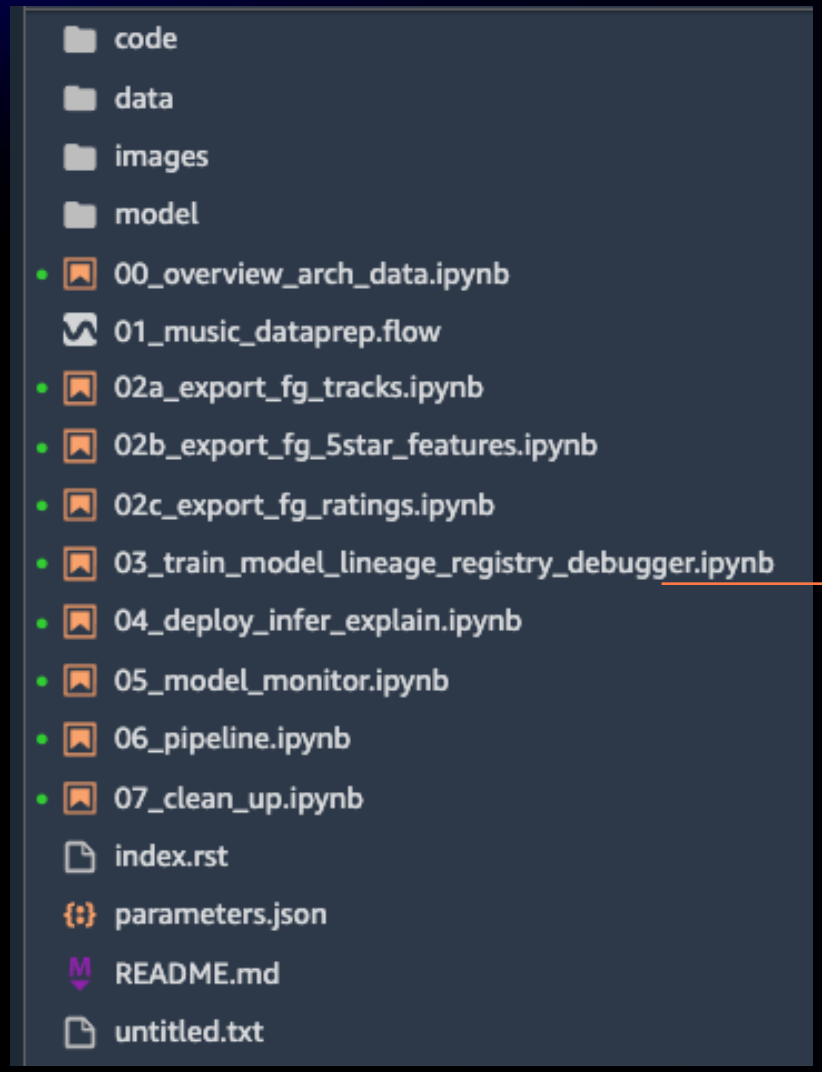
```
playlist_length = 10  # number of songs to recommend in playlist
playlist = inference_df.sort_values(by='rating', ascending=False).head(playlist_length)
print('Curated Playlist:\n', playlist['trackid'])
```

```
Curated Playlist:
 588    ec704261-69fd-4f94-b68b-76346b28bc4e
706    0acc9c7c-e89a-416b-8968-eb650e5a4282
944    9df507e4-9000-4f23-945a-b0684365a0a0
62     369e89cd-f59d-4caf-a29d-d011d8afeb95
75     31bceeb9-89ec-46f9-9640-77bcc0e5f8d1
254    b819a845-7ad8-454a-9f47-1715424957b7
757    4db1bbd8-3771-46ae-9f2e-dece0e8a0e98
40     1540c680-83f3-417e-a19e-227eaa02f7bb
917    bc185762-d112-43f3-b234-b8828b471da8
213    a20ced93-5f07-47c3-9fa2-db8478beaf04
Name: trackid, dtype: object
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

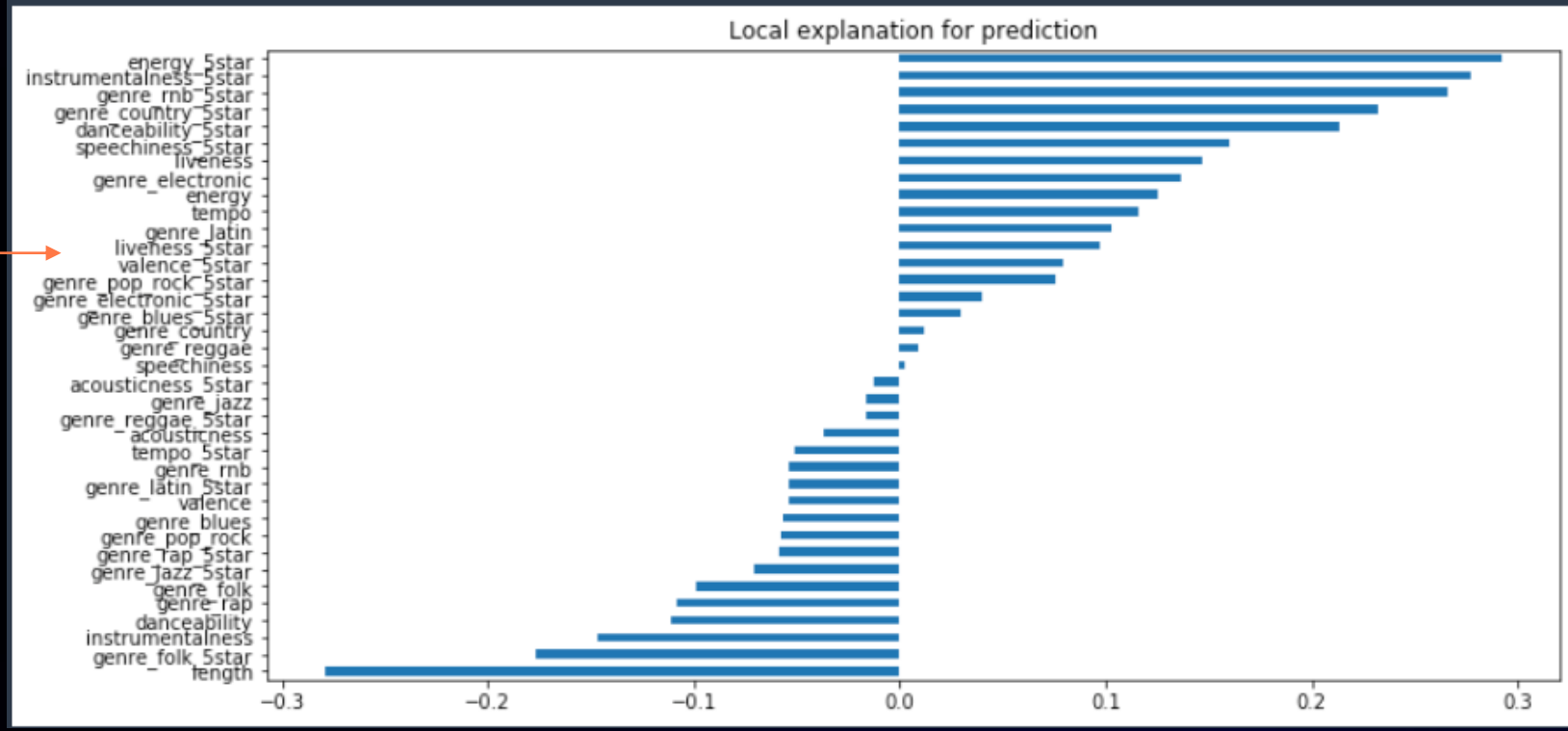
SageMaker Clarify: Model Explainability

- Part 4: Deploy Model & Inference using Online Feature Store
  - Deploy model
  - Create predictor
  - Infer new songs
  - Explain model predictions




```
local_e                                                    ns_shap/out.csv')
local_e
print("Model prediction:", playlist.iloc[0, 0])
plt.figure(figsize=(12,6))
local_explanations_out.iloc[0].sort_values().plot.barh(title='Local explanation for prediction')
```

```
Model prediction: 5.347079753875732
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a36657110>
```



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Monitoring models in production



Automatic data collection



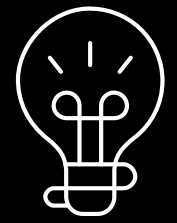
Continuous monitoring



Flexibility with rules



Visual data analysis

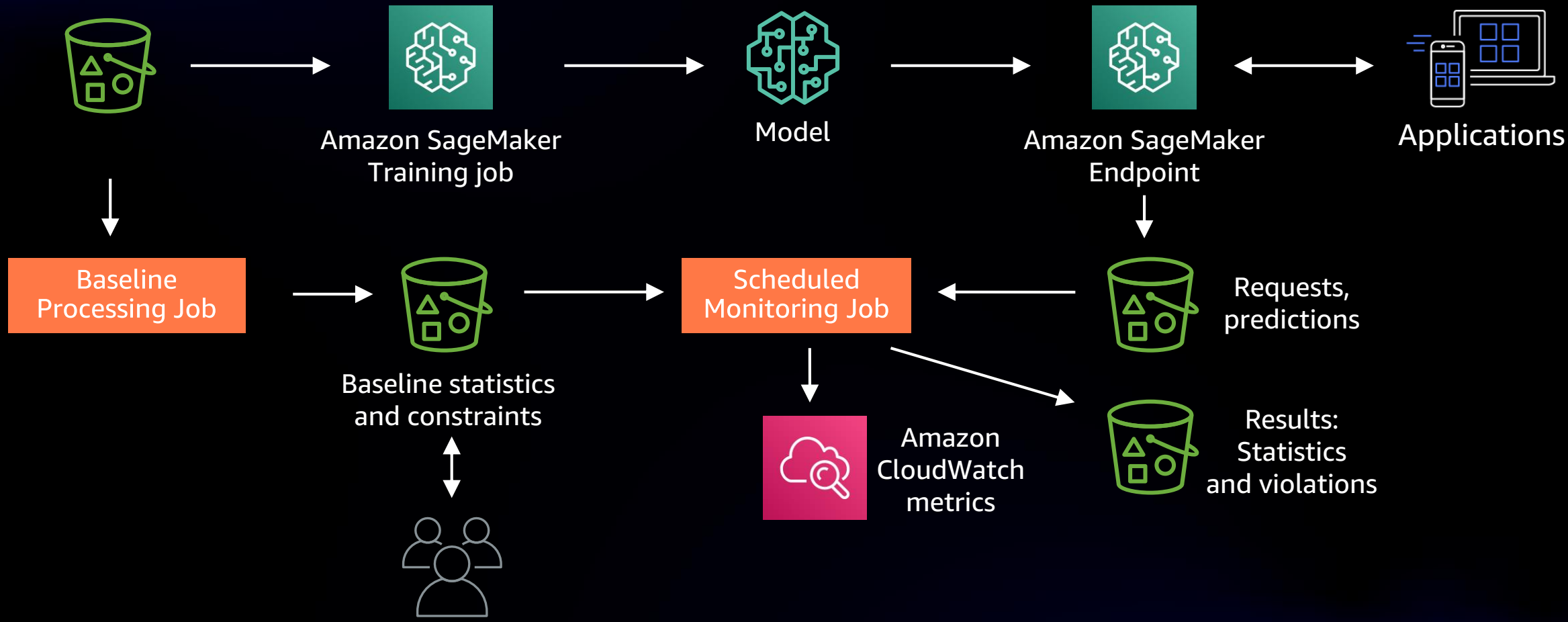


Amazon CloudWatch integration

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SageMaker Model Monitor



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Model Monitoring: Config

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

- Part 5: Model Monitor
  - Enable data capture
  - Baselining
  - Enable continous monitoring

```
%%time
# Change parameters as you would like - adjust sampling percentage,
#  chose to capture request or response or both
data_capture_config = DataCaptureConfig(
    enable_capture = True,
    sampling_percentage=25,
    destination_s3_uri=s3_capture_upload_path,
    kms_key_id=None,
    capture_options=["REQUEST", "RESPONSE"],
    csv_content_types=["text/csv"],
    json_content_types=["application/json"]
)

# Now it is time to apply the new configuration and wait for it to be applied
predictor = Predictor(endpoint_name=endpoint_name)
predictor.update_data_capture_config(data_capture_config=data_capture_config)
sm_session.wait_for_endpoint(endpoint=endpoint_name)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Model Monitoring: Baselining

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

- Part 5: Model Monitor
  - Enable data capture
  - Baselining
  - Enable continous monitoring

Create a baselining job with the validation dataset

Now that we have the training data ready in S3, let's kick off a job to `suggest` constraints. `DefaultModelMonitor.suggest_baseline(..)` kicks off a `ProcessingJob` using a SageMaker container to generate the constraints. Please edit the configurations to fit your needs.

```
]: from sagemaker.model_monitor import DefaultModelMonitor
from sagemaker.model_monitor.dataset_format import DatasetFormat
from sagemaker import get_execution_role
import datetime

role = get_execution_role(sagemaker_session=sm_session)

datetime_stamp = datetime.datetime.now().strftime("%Y-%m-%d-%H%M%S")

my_default_monitor = DefaultModelMonitor(
    role=role,
    instance_count=2,
    instance_type='ml.m5.xlarge',
    volume_size_in_gb=20,
    max_runtime_in_seconds=1800,
    base_job_name=f"{prefix}-monitor-{datetime_stamp}"
)
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Model Monitoring: Continuous Monitoring

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

- Part 5: Model Monitor
  - Enable data capture
  - Baselining
  - Enable continous monitoring

Create a schedule

We are ready to create a model monitoring schedule for the Endpoint created earlier with the baseline resources (constraints and statistics).

```
from sagemaker.model_monitor import CronExpressionGenerator
import datetime as datetime
from time import gmtime, strftime


mon_schedule_name = 'music-rec-monitor-schedule-{}'.format(datetime.datetime.now().strftime("%Y-%m-%d-%H%M%S"))
s3_report_path = f's3://{bucket}/{prefix}/monitor/report'

try:
    my_default_monitor.create_monitoring_schedule(
        monitor_schedule_name=mon_schedule_name,
        endpoint_input=endpoint_name,
        output_s3_uri=s3_report_path,
        statistics=my_default_monitor.baseline_statistics(),
        constraints=my_default_monitor.suggested_constraints(),
        schedule_cron_expression=CronExpressionGenerator.daily(),
        enable_cloudwatch_metrics=True,
    )
    print(f"Created monitoring schedule {mon_schedule_name}")
except:
    my_default_monitor.update_monitoring_schedule(
        endpoint_input=endpoint_name,
        schedule_cron_expression=CronExpressionGenerator.daily(),
        enable_cloudwatch_metrics=True,
    )
    print(f"Updated monitoring schedule {my_default_monitor.monitoring_schedule_name}")
```

```
Created monitoring schedule music-rec-monitor-schedule-2021-09-19-225419
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Model Monitoring: Continuous Monitoring

- code
- data
- images
- model
- 00_overview_arch_data.ipynb
- 01_music_dataprep.flow
- 02a_export_fg_tracks.ipynb
- 02b_export_fg_5star_features.ipynb
- 02c_export_fg_ratings.ipynb
- 03_train_model_lineage_registry_debugger.ipynb
- 04_deploy_infer_explain.ipynb
- 05_model_monitor.ipynb
- 06_pipeline.ipynb
- 07_clean_up.ipynb
- index.rst
- parameters.json
- README.md
- untitled.txt

- Part 5: Model Monitor
  - Enable data capture
  - Baselining
  - Enable continous monitoring

Explore the generated constraints and statistics

```
import pandas as pd

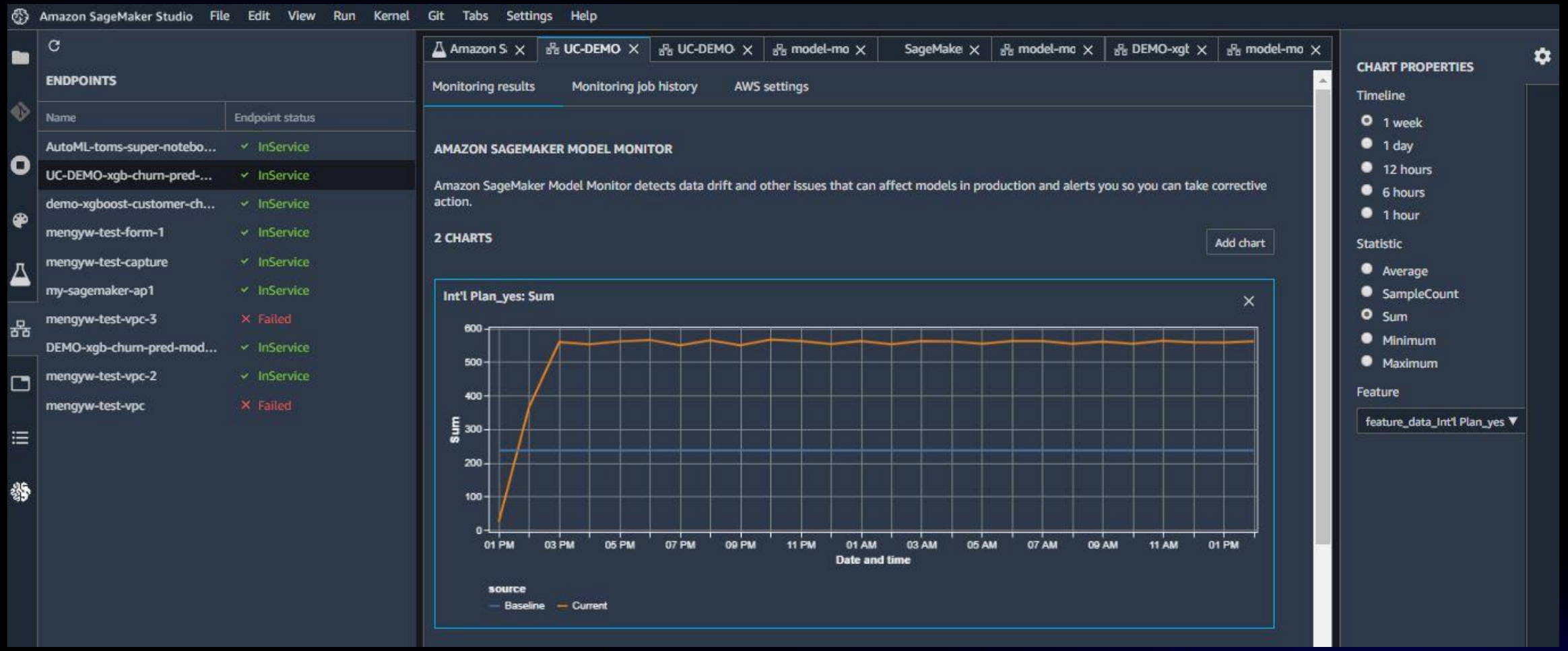
baseline_job = my_default_monitor.latest_baselining_job
schema_df = pd.io.json.json_normalize(baseline_job.baseline_statistics().body_dict["features"])
schema_df.head(10)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:4: FutureWarning: pandas.io.json.json_normalize is de
precated, use pandas.json_normalize instead
  after removing the cwd from sys.path.
```

	name	inferred_type	numerical_statistics.common.num_present	numerical_statistics.common.num_missing	numerical_statistics.mean	numerical_st
0	_c0	Fractional	68525.0	0.0	3.469901	2.3
1	_c1	Fractional	68525.0	0.0	248.140752	1.7
2	_c2	String	NaN	NaN	NaN	
3	_c3	String	NaN	NaN	NaN	
4	_c4	String	NaN	NaN	NaN	
5	_c5	Fractional	68525.0	0.0	0.065646	4.4
6	_c6	String	NaN	NaN	NaN	
7	_c7	Fractional	68525.0	0.0	0.170376	1.1
8	_c8	Fractional	68525.0	0.0	123.429822	8.4
9	_c9	Fractional	68525.0	0.0	0.231588	1.5

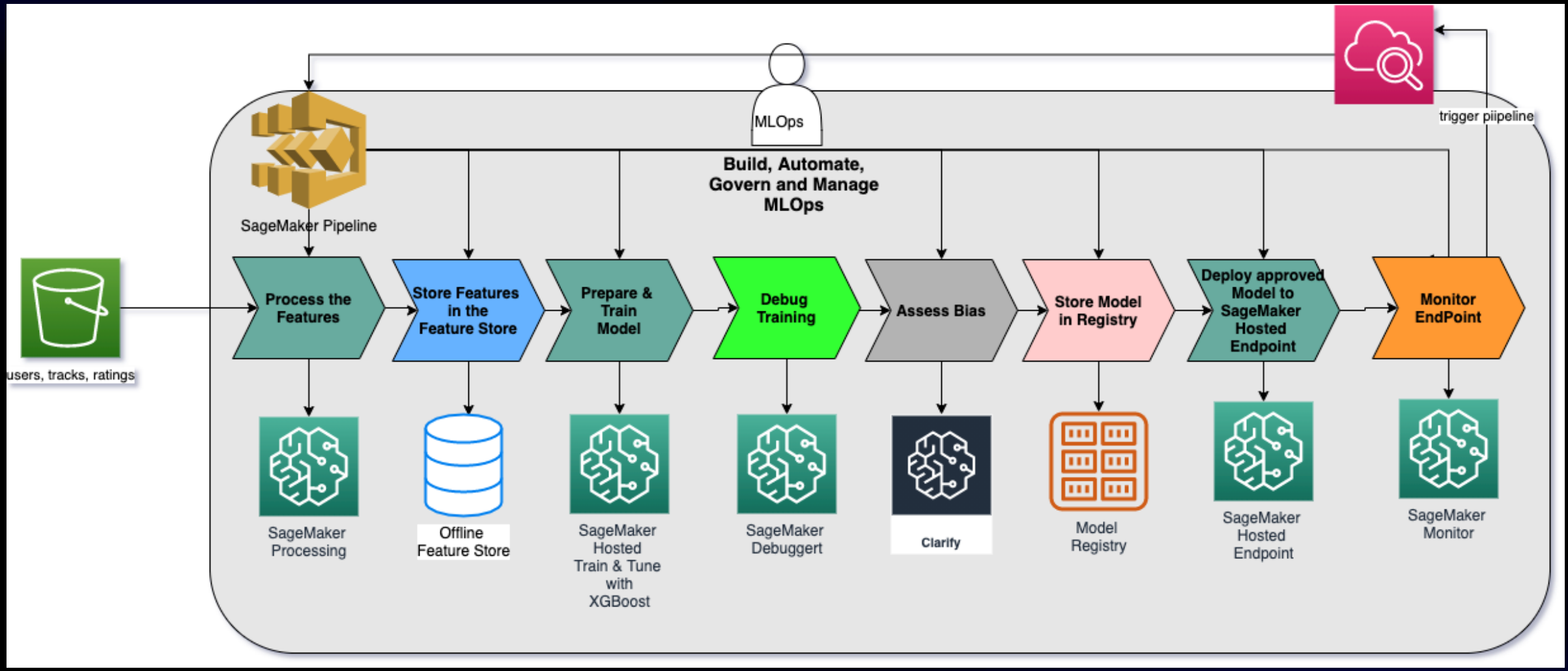
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using Amazon SageMaker Model Monitor, you can identify model drift and take action



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Automated workflow with pipelines for MLOps



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon SageMaker Pipelines

Build fully automated machine learning workflows at scale



Python SDK for quickly and easily building ML Workflows

Define parameters, pipeline steps (Data processing, Training, Tuning, Batch Inference, Model Registration), and pipeline DAG for execution. Pipeline steps are data aware and output of one step can become the input to the next.



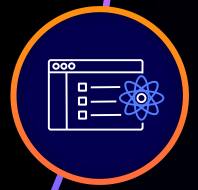
SageMaker managed workflow executions

Pipeline execution runs, associated metrics, and logs are retained and viewable within SageMaker Studio in real-time for easy debugging of pipelines. Cache pipeline executions for reuse and troubleshooting.



Catalog model versions in Model Registry

Register trained models to a central model registry. Built-in model versioning and cross-account support. View model metrics & approve model deployments.



Fully managed MLOps with built-in support for CI/CD

Built-in MLOps templates for model building and deployment automate ML workflows using AWS Code* Services. DevOps can bring custom templates published in AWS Service Catalog. Automate ML Pipelines based on scheduled events, on code and data changes, and drift detection with Model Monitor.

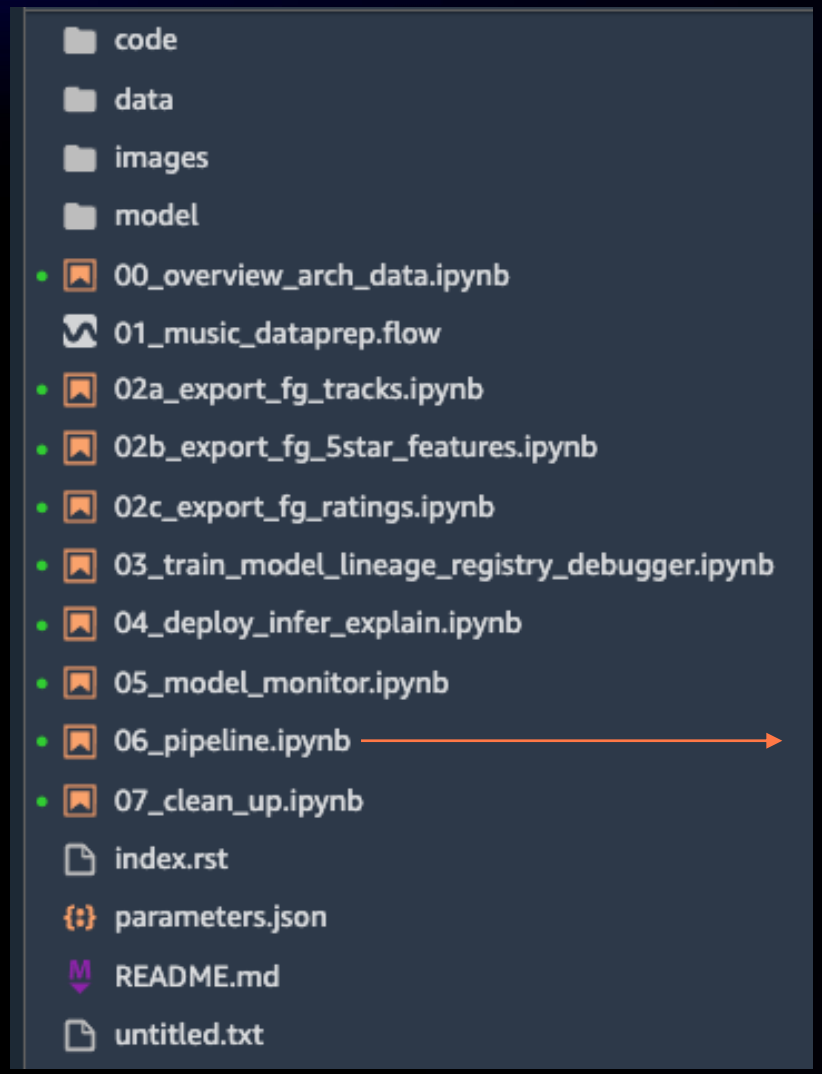


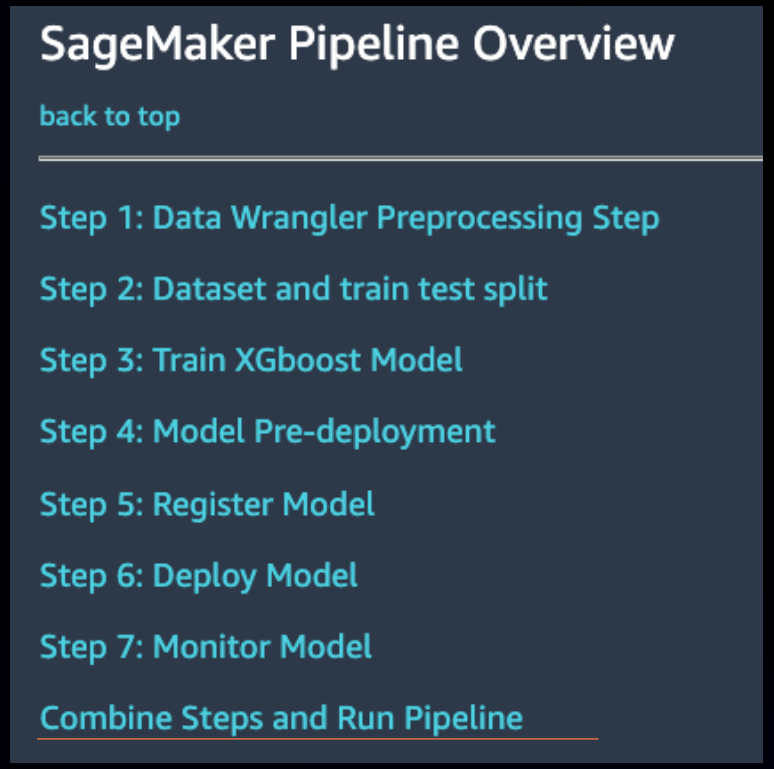
End-to-end Lineage Tracking for governance and audits

Built-in support for tracking lineage for data, code, metrics, and model artifacts.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

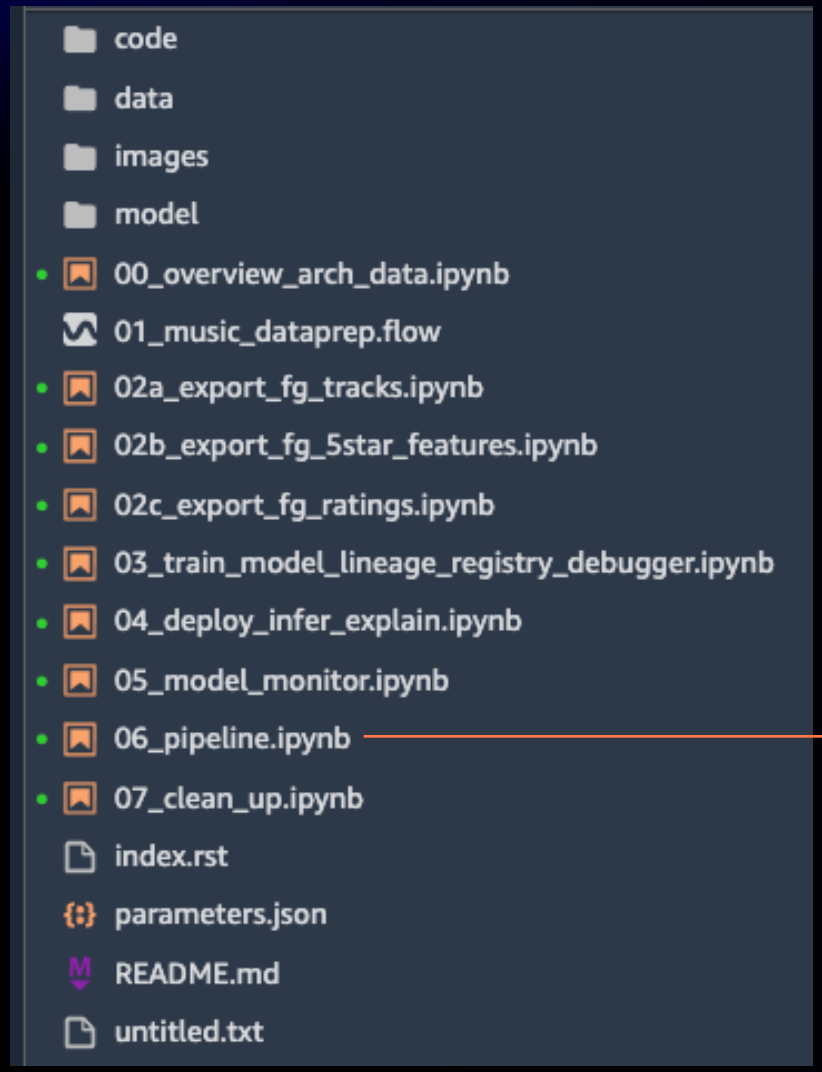
MLOps : SageMaker Pipelines





© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MLOps: SageMaker Pipelines




```
pipeline_name = f'MusicRecommendationPipeline'
dataprep_pipeline_name = f'MusicRecommendationDataPrepPipeline'
train_deploy_pipeline_name = f'MusicRecommendationTrainDeployPipeline'

ps.add({'pipeline_name':pipeline_name, 'dataprep_pipeline_name':dataprep_pipeline_name,
        'train_deploy_pipeline_name':train_deploy_pipeline_name,
        'pipeline_endpoint_name':pipeline_endpoint_name},
       namespace='music-rec'
)
ps.store()
```

```
date and time = 19/09/2021 23:00:12
```

```
### Option 1: The Entire Pipeline End to end
```

```
pipeline_name = f'MusicRecommendationPipeline'

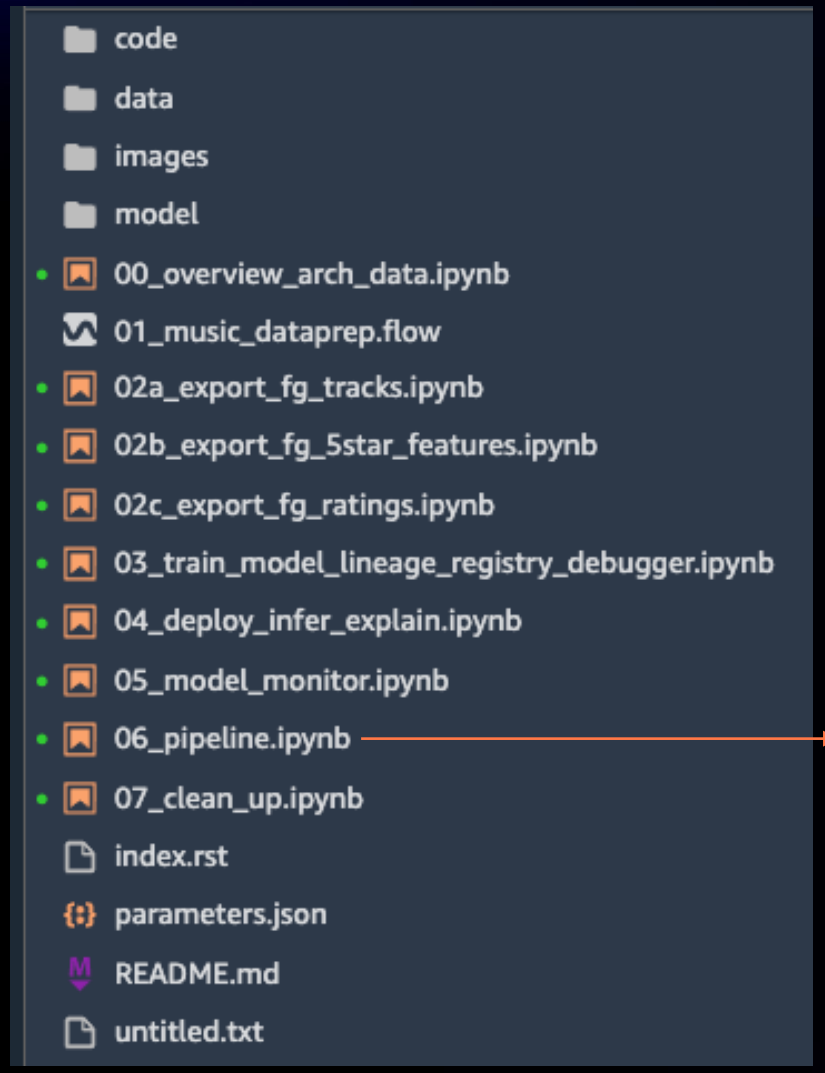
ps.add({'pipeline_name':pipeline_name}, namespace='music-rec')
ps.store()
```

```
date and time = 19/09/2021 23:00:13
```

```
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[
        train_instance_param,
        model_approval_status],
    steps=[
        flow_step_tracks,
        flow_step_user_preferences,
        flow_step_ratings,
        create_dataset_step,
        train_step,
        create_model_step,
        register_step,
        deploy_step,
        monitor_model_step
    ])
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MLOps : SageMaker Pipelines




```
data_sources = []

## Input - S3 Source: tracks.csv
data_sources.append(ProcessingInput(
    source=f"s3://{bucket}/{prefix}/data/tracks_new.csv", # Y
    destination=f"{processing_dir}/data/tracks_new.csv",
    input_name="tracks_new.csv",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
))

## Input - S3 Source: ratings.csv
data_sources.append(ProcessingInput(
    source=f"s3://{bucket}/{prefix}/data/ratings_new.csv", # You can
    destination=f"{processing_dir}/data/ratings_new.csv",
    input_name="ratings_new.csv",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
))

## Input - Flow: 01_music_dataprep.flow
flow_input = ProcessingInput(
    source=flow_s3_uri,
    destination=f"{processing_dir}/flow",
    input_name="flow",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
)
```

SageMaker Pipeline Overview

back to top

Step 1: Data Wrangler Preprocessing Step

Step 2: Dataset and train test split

Step 3: Train XGboost Model

Step 4: Model Pre-deployment

Step 5: Register Model

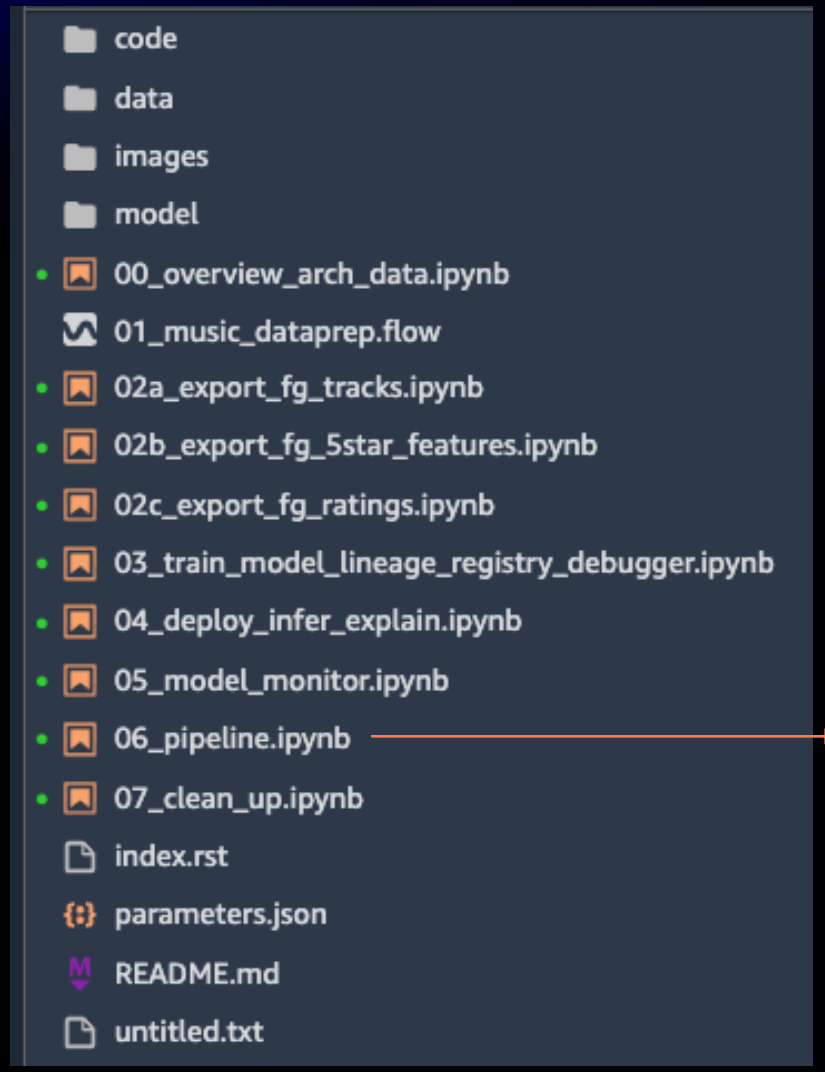
Step 6: Deploy Model

Step 7: Monitor Model

Combine Steps and Run Pipeline

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MLOps: SageMaker Pipelines



SageMaker Pipeline Overview

back to top

Step 1: Data Wrangler Preprocessing Step

Step 2: Dataset and train test split

Step 3: Train XGboost Model

```
hyperparameters = {
    "max_depth": "4",
    "eta": "0.2",
    "objective": "reg:squarederror",
    "num_round": "100"
}

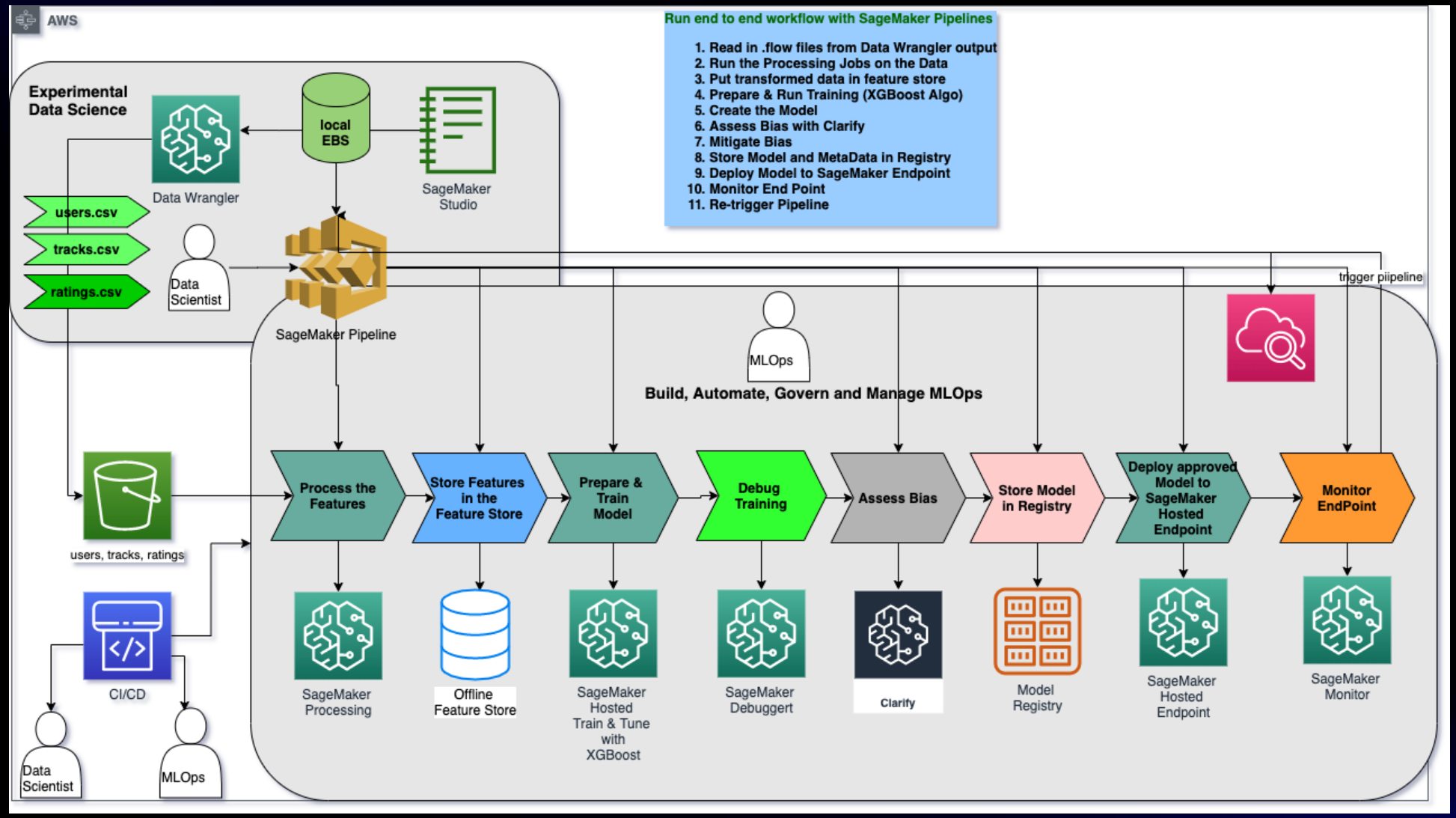
save_interval = 5
```

```
xgb_estimator = Estimator(
    role=sagemaker_role,
    instance_count=2,
    instance_type='ml.m5.4xlarge',
    volume_size=60,
    image_uri=sagemaker.image_uris.retrieve("xgboost", region, "0.90-2"),
    hyperparameters=hyperparameters,
    output_path=f's3://{bucket}/{prefix}/training_jobs',
    base_job_name='xgb-music-rec-model-pipeline',
    max_run=1800
)
```

```
train_step = TrainingStep(
    name='TrainStep',
    estimator=xgb_estimator,
    inputs={
        'train': sagemaker.inputs.TrainingInput(
            s3_data=create_dataset_step.properties.ProcessingOutputConfig.Outputs['train_data'].S3Output.S3Uri,
            content_type="text/csv"
        ),
        'validation': sagemaker.inputs.TrainingInput(
            s3_data=create_dataset_step.properties.ProcessingOutputConfig.Outputs['test_data'].S3Output.S3Uri,
            content_type="text/csv"
        )
    }
```

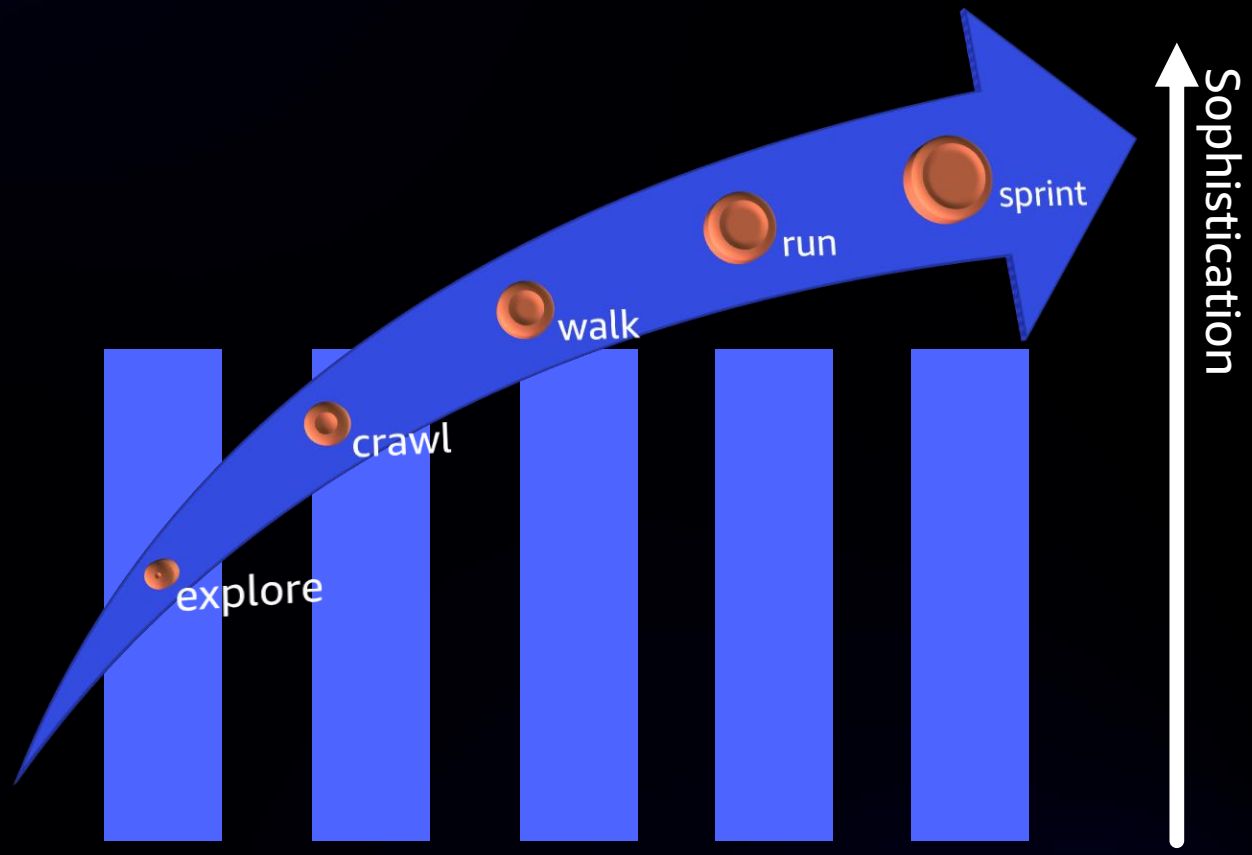
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML Ref Arch with SageMaker instantiation



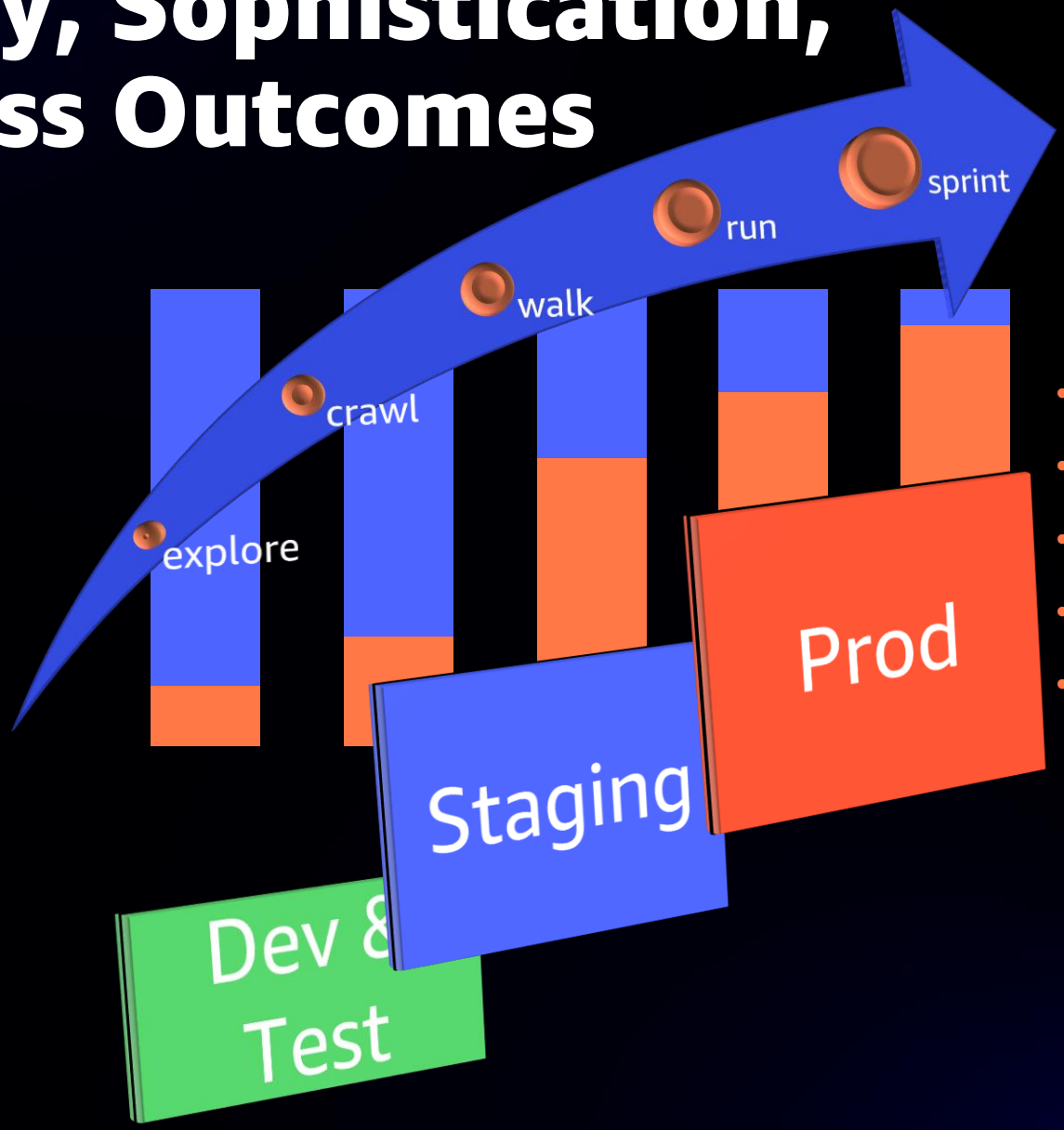
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML Maturity, Sophistication, and Business Outcomes



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML Maturity, Sophistication, and Business Outcomes

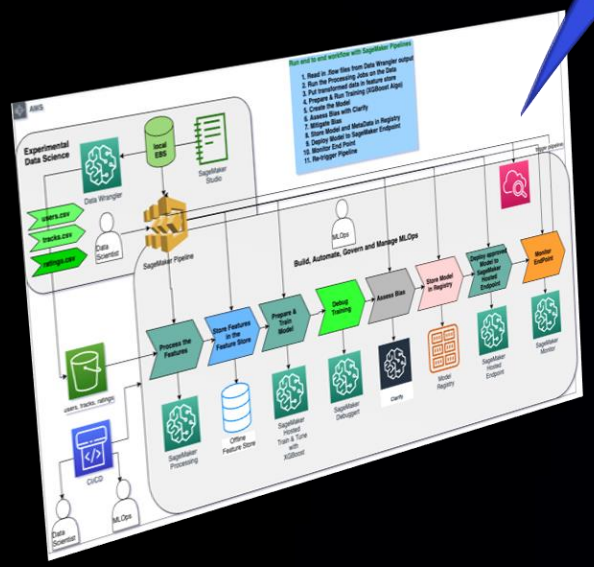


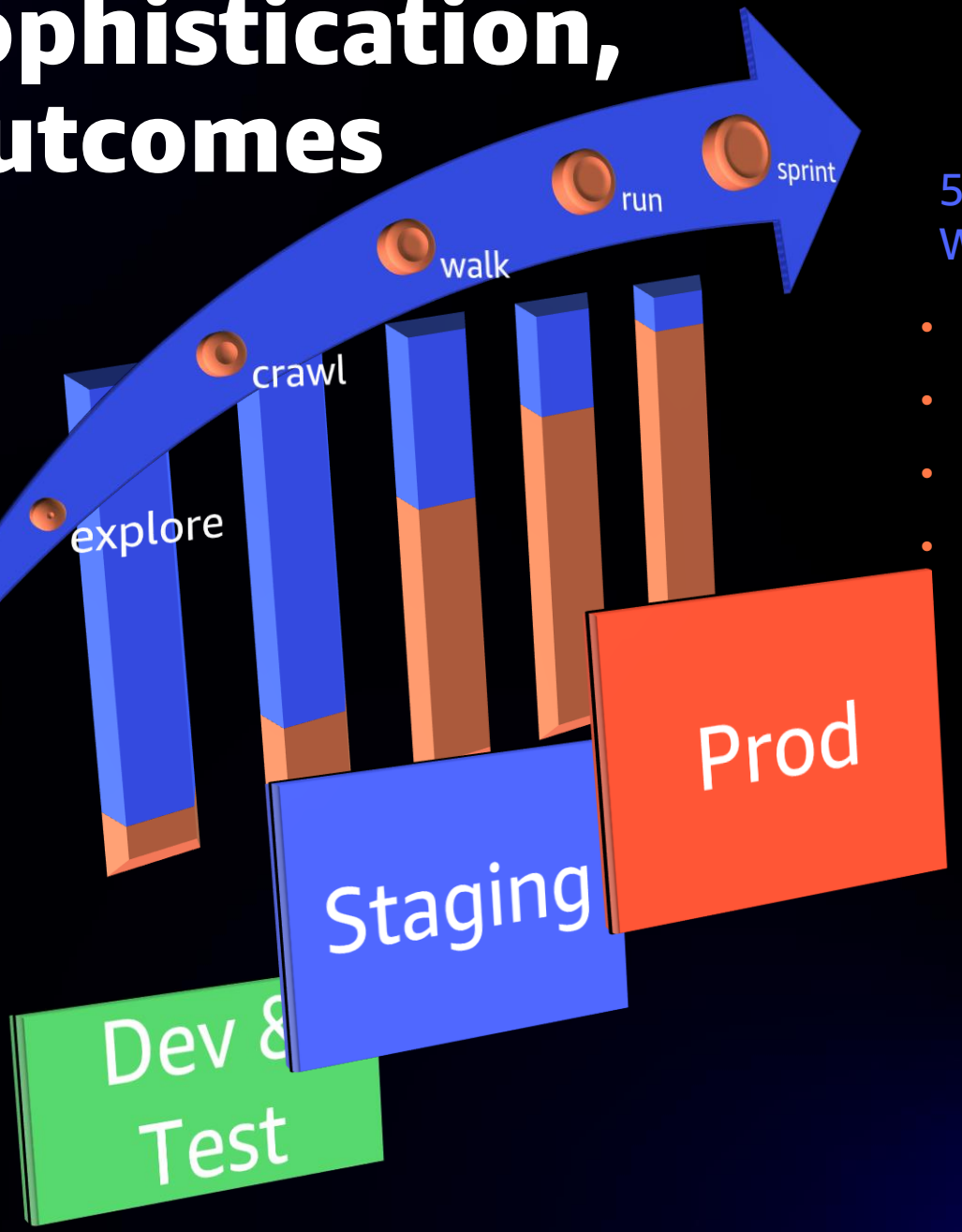
5 Pillars of AWS Well-Architected Framework

- **Operational Excellence**
- **Security**
- **Reliability**
- **Performance Efficiency**
- **Cost Optimization**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML Maturity, Sophistication, and Business Outcomes





5 Pillars of AWS Well-Architected Framework

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics

Samir Joshi

ML Engineer, iQ R&D
Qualtrics

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics

At Qualtrics, we work on Experience Management, building technology that closes experience gaps for customers, employees, brands, and products.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Three inseparable services.
A new operating system for XM.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use Cases

- Natural Language Processing (NLP)
- Predictive Models with Tabular Datasets
- Time-Series Prediction
- … and more!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics X SageMaker

Qualtrics standardized its labeling, data prep, training, and deployment processes using SageMaker.

We leverage SageMaker purpose-built tools such as:

- **Ground Truth**
- **Real-Time Inference**
- **Batch Transform**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Challenges Encountered – Universal Models

- Scaling large neural deep learning models that are dependent on GPUs
- Fragmented ML tools and processes that slowed innovation
- Lack of out-of-the-box labeling tools

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Challenges Encountered – Customer-Specific Models

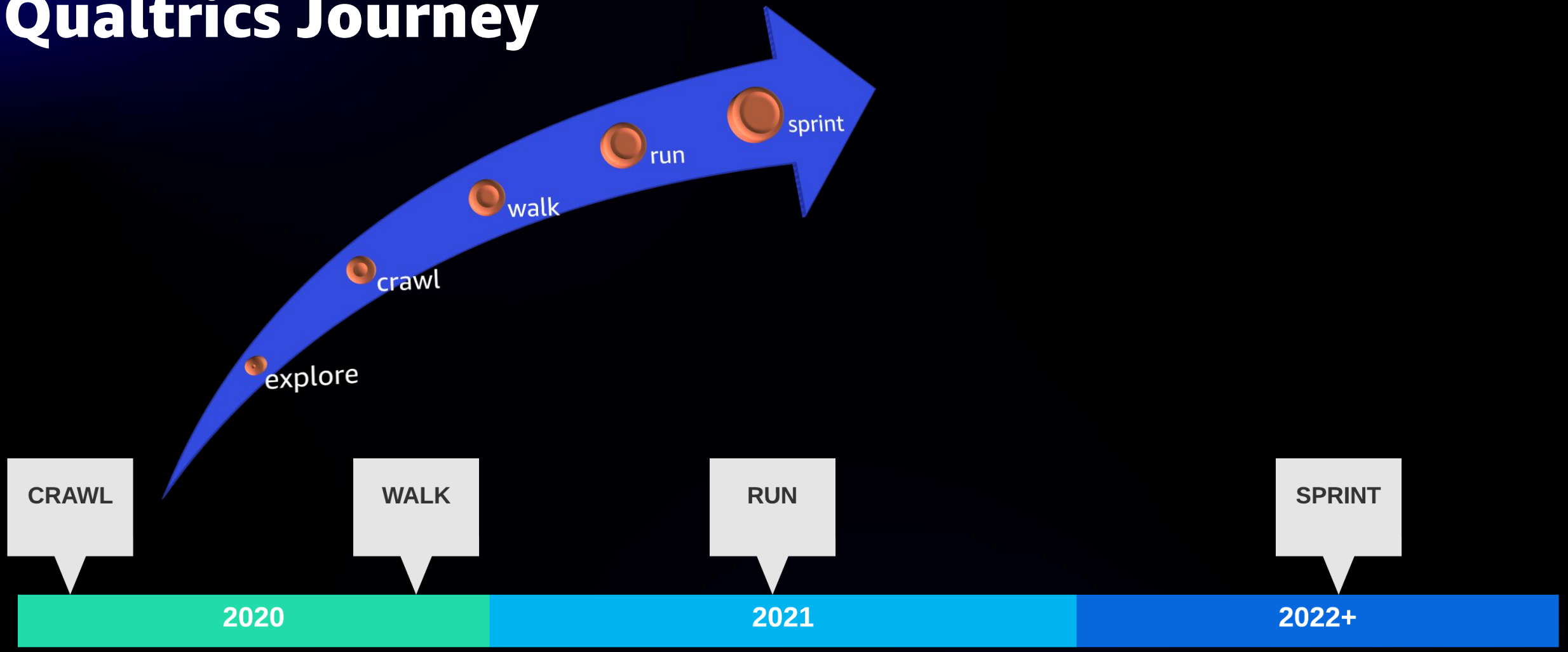
- Workflows that guarantee customer isolation at each step of the ML lifecycle
- Large order of models
- Invoked with widely different load patterns

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – ML Maturity

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Crawl

- Smaller models are run in production on CPU instances inside Qualtrics Co-located infrastructure.
- This presented 2 major challenges:
  - Scalability
  - Time to launch/iteration speed



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Walk

- First phase of models in production running on SageMaker
  - Data analysis & training
  - Real-time inference
- Experimentation platform



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Walk: Leveraging SageMaker

- SageMaker notebooks/SageMaker Studio have become our environment as the scientist's workbench
- SageMaker Ground Truth has allowed us to label data efficiently
- Auto scaled GPU instances through a managed endpoints



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Run

- Automated models created and deployed
- Pipelines for
  - Creating SM models
  - SM training & HPO jobs
  - Validating and verifying model quality
  - Deploy models at production scale
- Currently 10–50 models



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Run

- Variety of large-scale transformer-based models running in production
  - Approx 700 MB (compressed and zipped)
  - Run as both
    - Real-time inference endpoint (latency sensitive) (API latency SLO p95 200 ms, including pre- and post-processing)
    - Batch transform (throughput sensitive)
- Feedback pipeline for model monitoring
- Feedback used for model retraining/model improvements



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Run: Leveraging SageMaker

SageMaker enabled us to horizontally scale

- **HPO**
- **Training jobs**
- **Batch transformation**

… for hundreds of independent models



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Qualtrics Journey – Sprint (Next Steps)

- Model versioning
- Data versioning
- A/B testing – configurable dialup for model rollout
- Feature stores
- Data Processing Pipelines
- Model CI/CD
- Continue to horizontally scale and tackle new use cases!



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why SageMaker?

- Encompasses all aspects of the ML Lifecycle
- Integrated into AWS
- Highly available & scalable
- Feature-rich and continues to release new features
- Partnership and commitment from the AWS team

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Lessons Learned/ Takeaways

- **Auto scaling GPUs allowed 10x throughput with larger models**
- **Time to launch notably reduced**
- **Significant reduction in infrastructure management**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Dr. Ali Arsanjani
arsanjani@amazon.com

Samir Joshi
samirj@qualtrics.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.