AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

ARC323

Designing Well-Architected machine learning workloads

Haleh Najafzadeh (she/her)
Principal AI/ML Solutions Architect Well-Architected
AWS

Ali Arsanjani (he/him)
Principal AI/ML Specialist Solutions Architect
AWS

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

Agenda

1. Introduction to the AWS Well-Architected Framework
2. The Well-Architected Framework for machine learning (ML lens)
3. The Well-Architected machine learning design principles
4. The Well-Architected machine learning lifecycle
5. Where should you apply the ML lens?
6. Complete use case implementation of ML lens on AWS

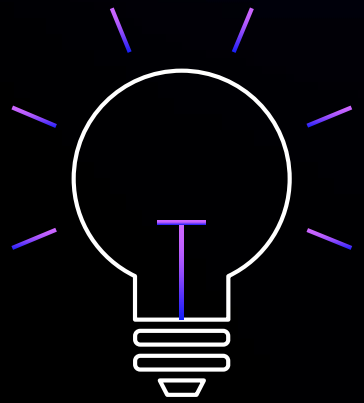
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why Well-Architected Framework?

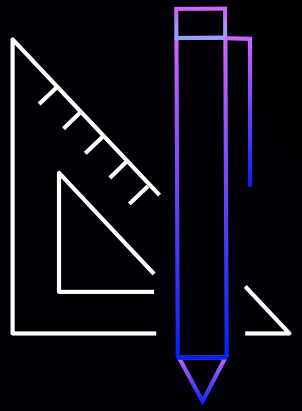
Helps you understand the benefits and risks of your decisions for building workloads

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

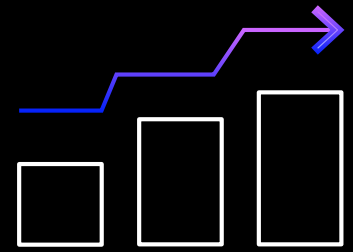
Why Well-Architected Framework?



Learn
best practices



Measure
architecture



Improve
architecture

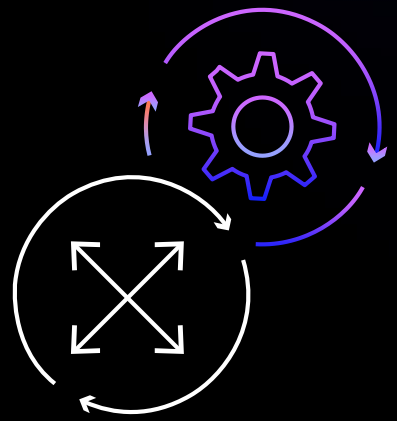
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is the Well-Architected Framework?

Based on five pillars – Provides **architectural best practices** as **guidance**

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

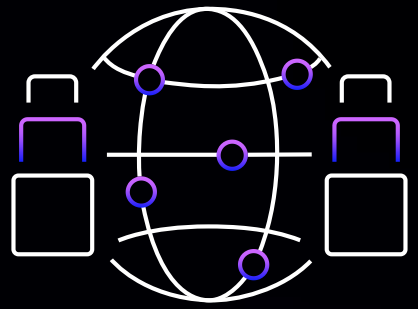
What is the Well-Architected Framework?



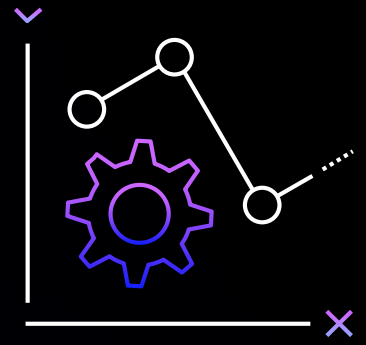
Operational excellence



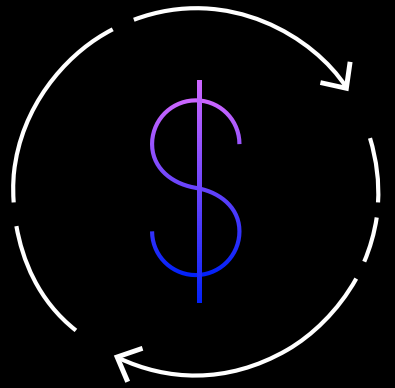
Security



Reliability



Performance efficiency



Cost optimization

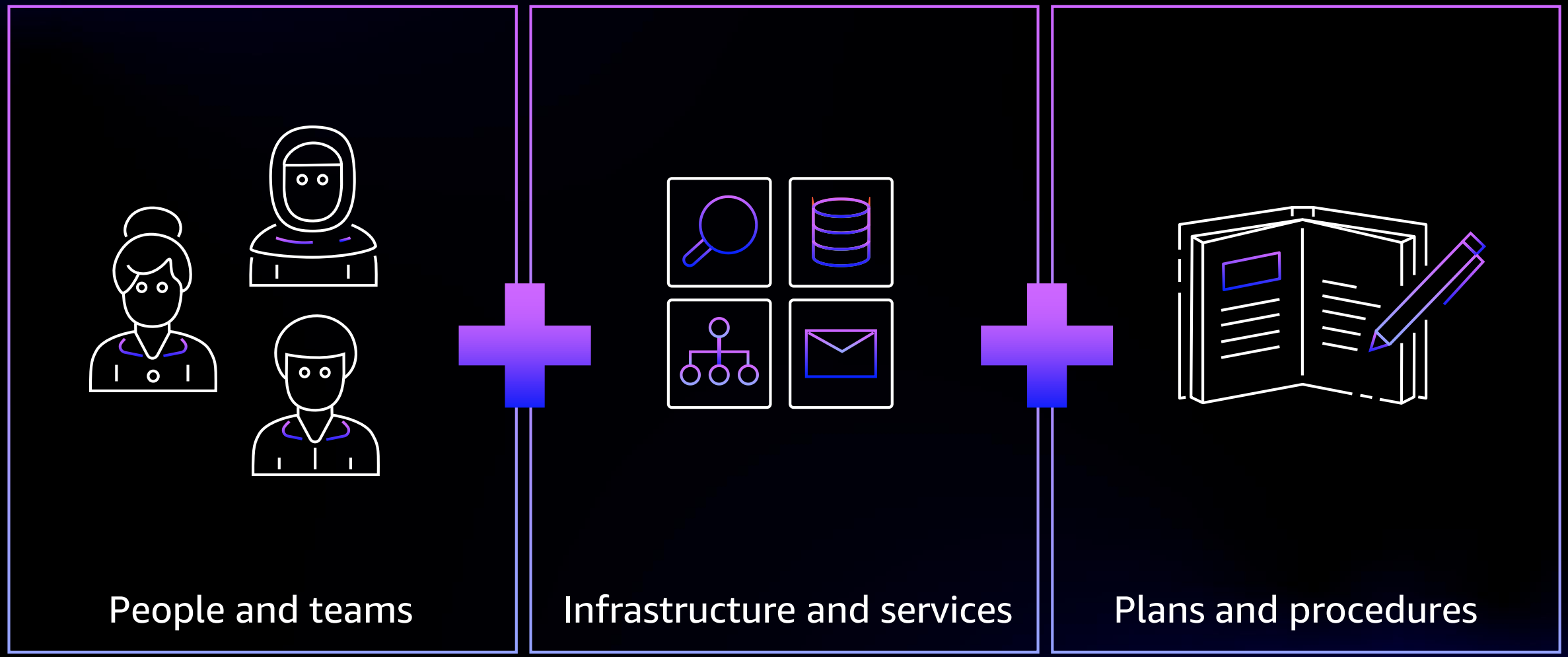
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is a workload?

Set of components that together deliver business value

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

What is a workload?



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

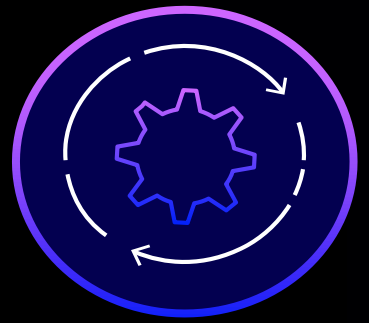
Why do we need the Well-Architected Framework for ML?

Unique characteristics of machine learning workloads

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

Why do we need the Well-Architected Framework for ML?

Algorithm learning



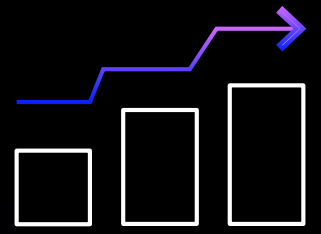
Continuous
Iterative, cyclic

Data dependency



Continuous
Change
Quality, drift

Model monitoring

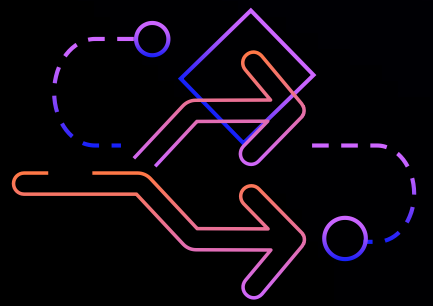


Continuous
Improvement

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

ML lens: Well-Architected Framework for ML

Machine learning domain



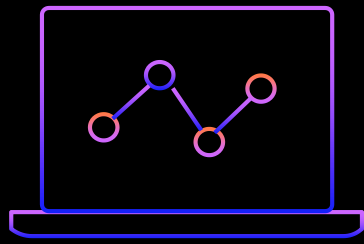
Extends the Well-Architected guidance

Enables reviewing existing workload



Best practices
Implementation guidance

Makes informed decisions for new workloads



Improves workloads throughout the lifecycle

aws

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

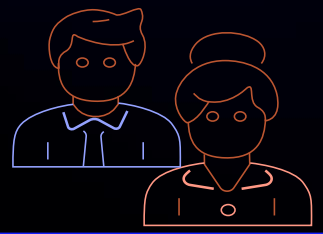
Well-Architected ML design principles

Considerations used as the basis for Well-Architected ML workloads

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

Well-Architected ML design principles

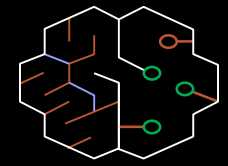
Assign ownership



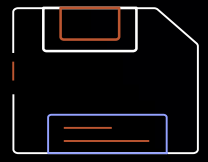
Enable automation



Optimize resources



Provide protection



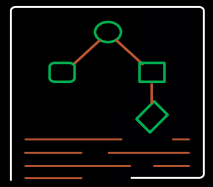
Enable reproducibility



Reduce cost



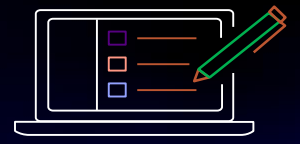
Enable resiliency



Enable reusability



Enable continuous improvement



aws

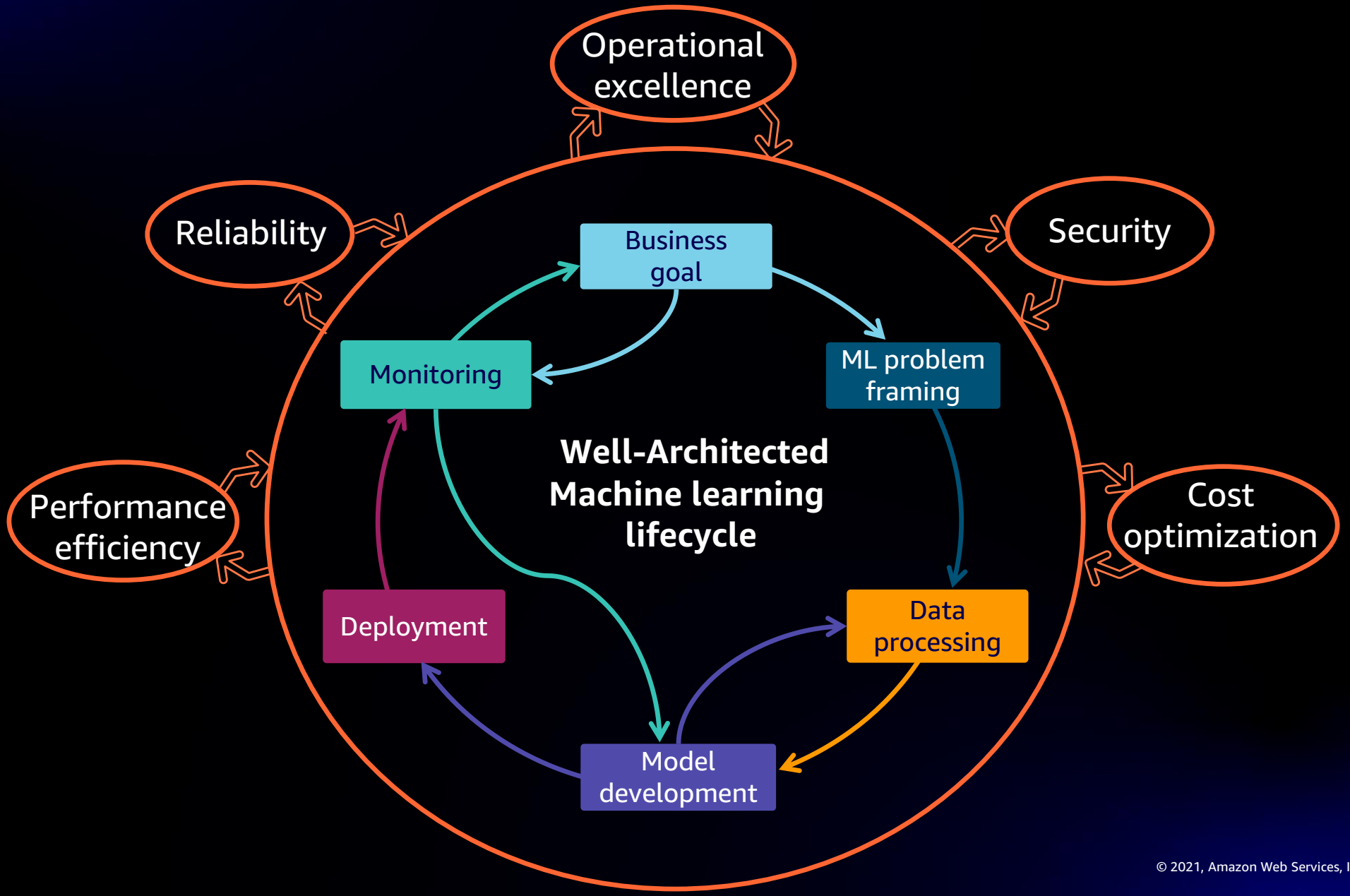
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Well-Architected machine learning lifecycle

Cyclic iterative process with guidance for developing ML workloads

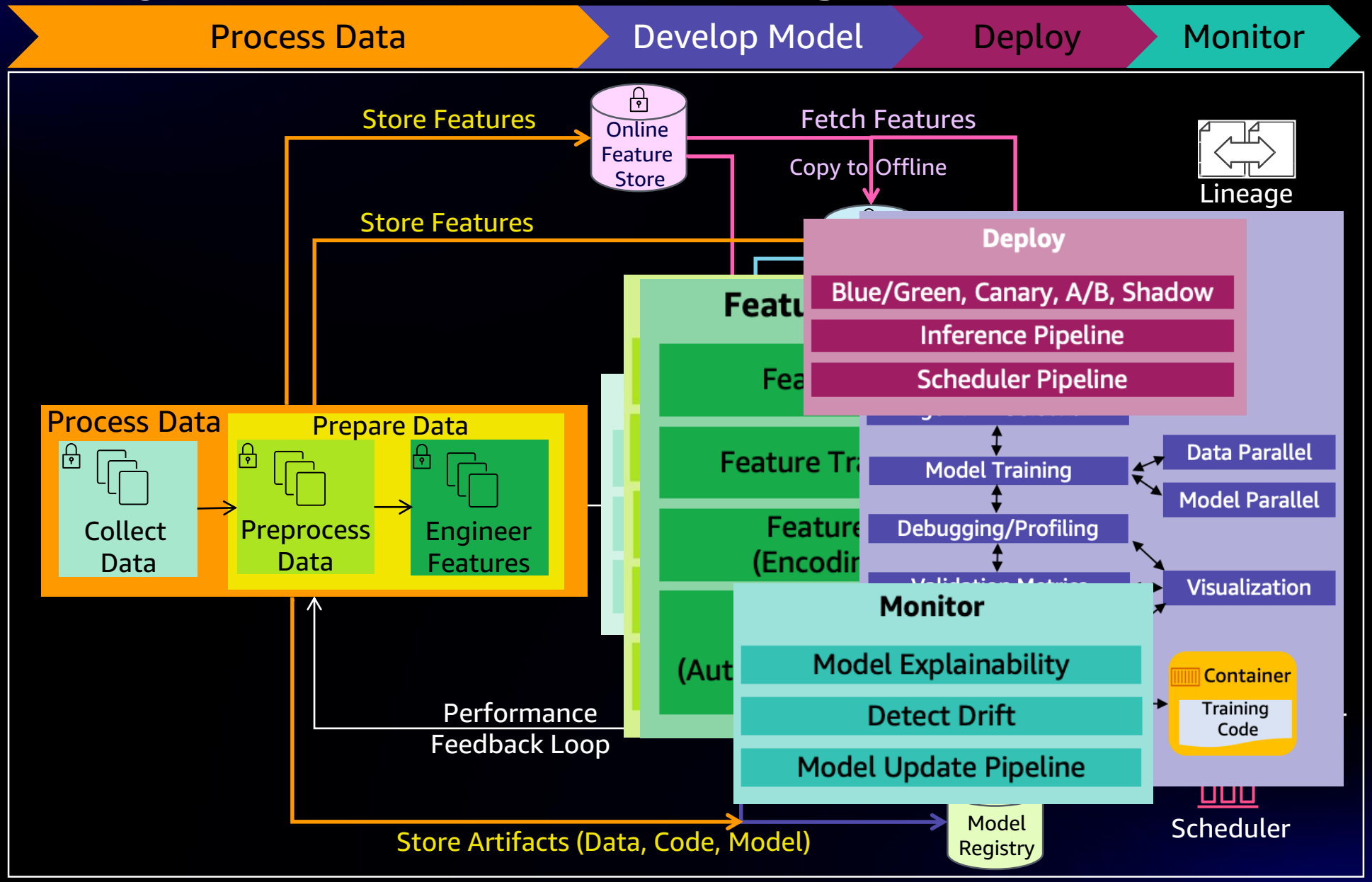
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Well-Architected machine learning lifecycle



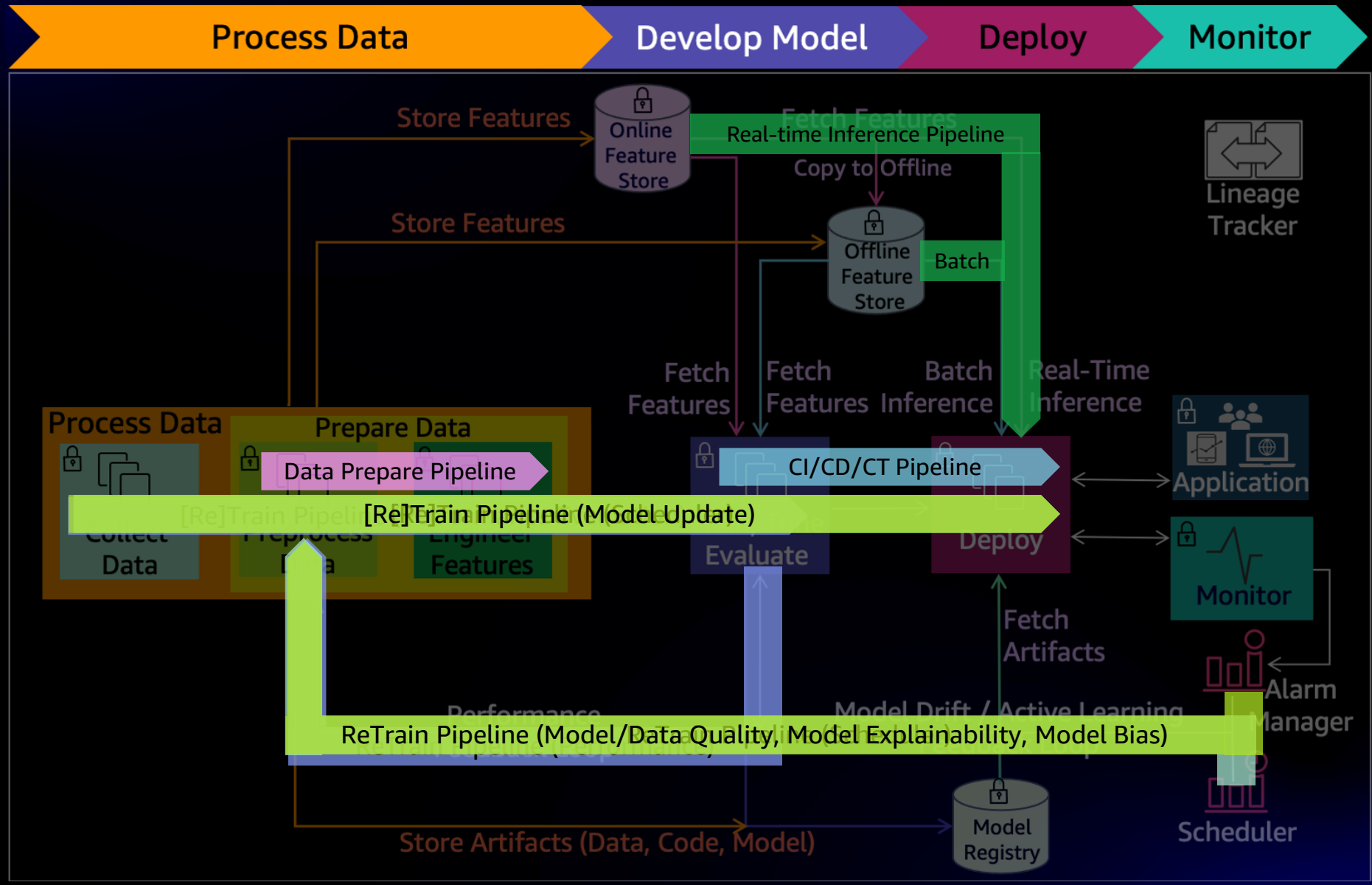
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ML lifecycle architecture diagram components



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

ML lifecycle architecture diagram pipelines



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

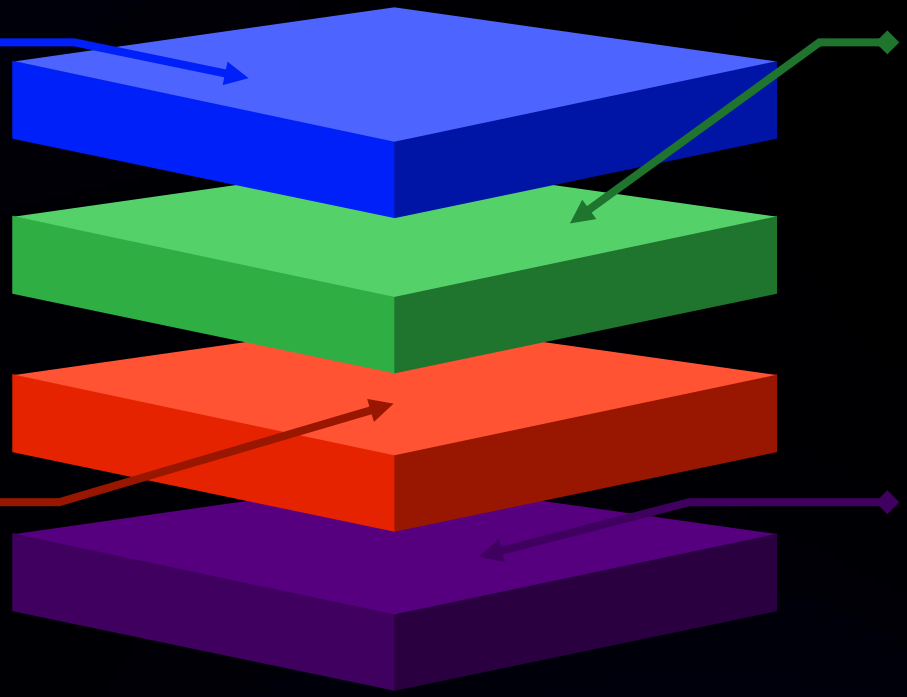
Where to use ML lens

Build and deploy faster

Experiment and release value more often

Lower or mitigate risks

Address risks before they impact your business



Make informed decisions

Awareness of what might impact your business outcomes

Learn AWS best practices

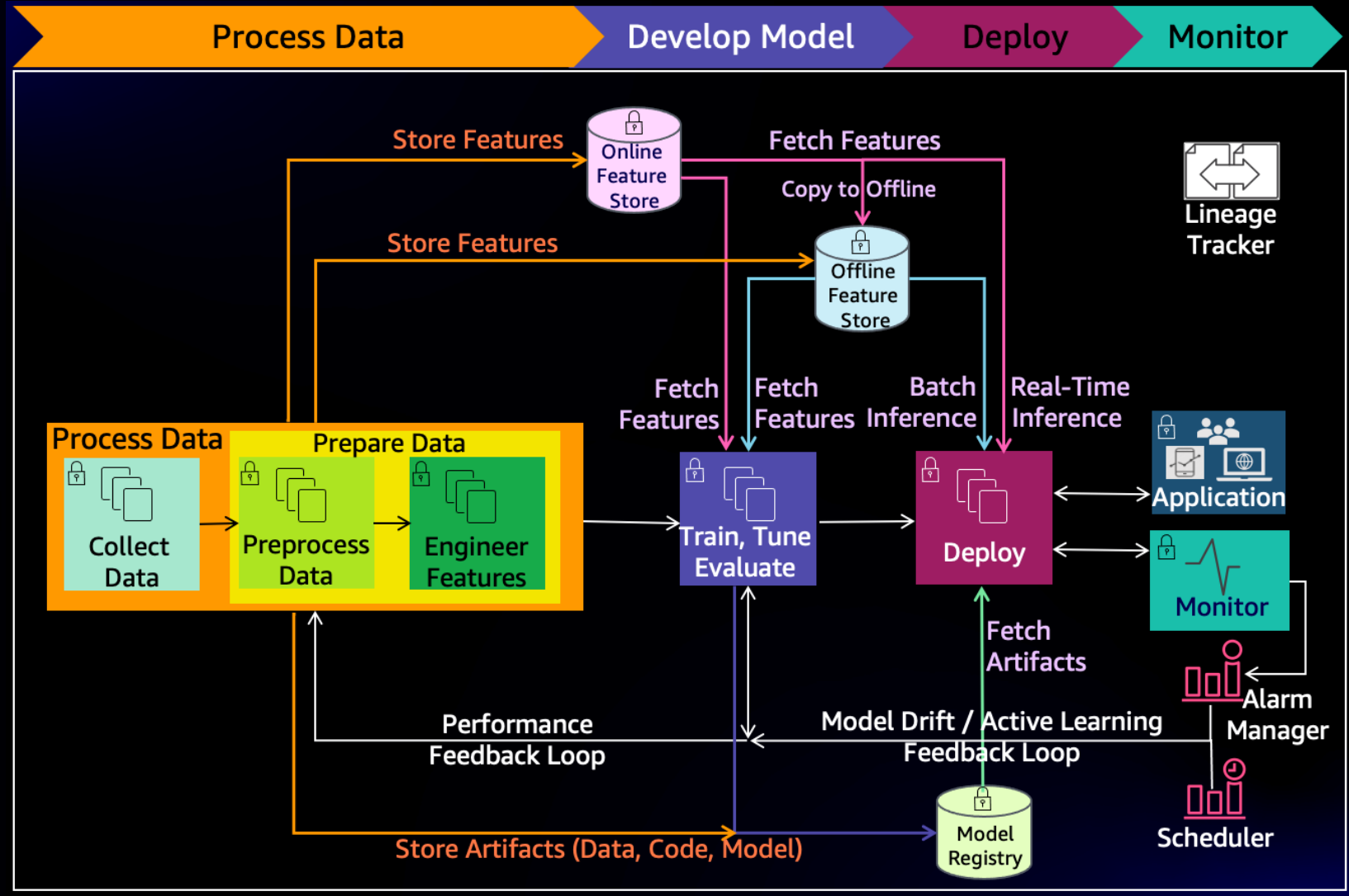
Be aware of the established best practices

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

Use case: Amazon SageMaker music recommendation

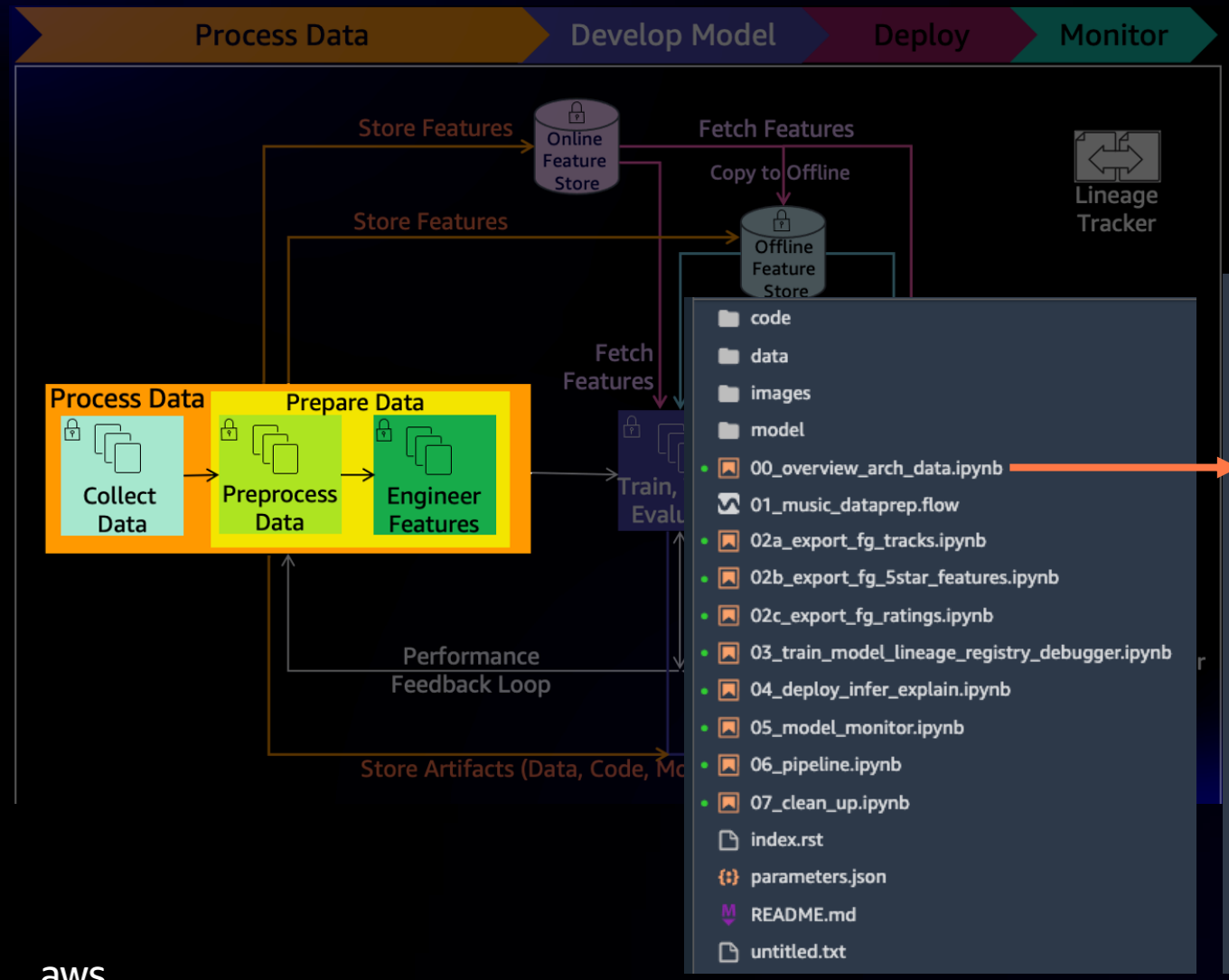
https://github.com/aws/amazon-sagemaker-examples/tree/master/end_to_end/music_recommendation

- 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 example: exploratory data analysis (EDA)



Data Preparation (**Wrangling** during Interactive Data Analysis)
Leverage **Visualization** for Exploratory Data Analysis (EDA)

Data Preprocessing	Feature Engineering
Clean (Replace, Impute, Remove Outliers, Duplicates)	Feature Selection
Partition (Train, Validate, Test)	Feature Transformation
Scale (Normalize, Standardize)	Feature Creation (Encoding, Binning)
Unbias, Balance (Detection & Mitigation)	Feature Extraction (Automated in Deep Learning)
Augment	

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]:

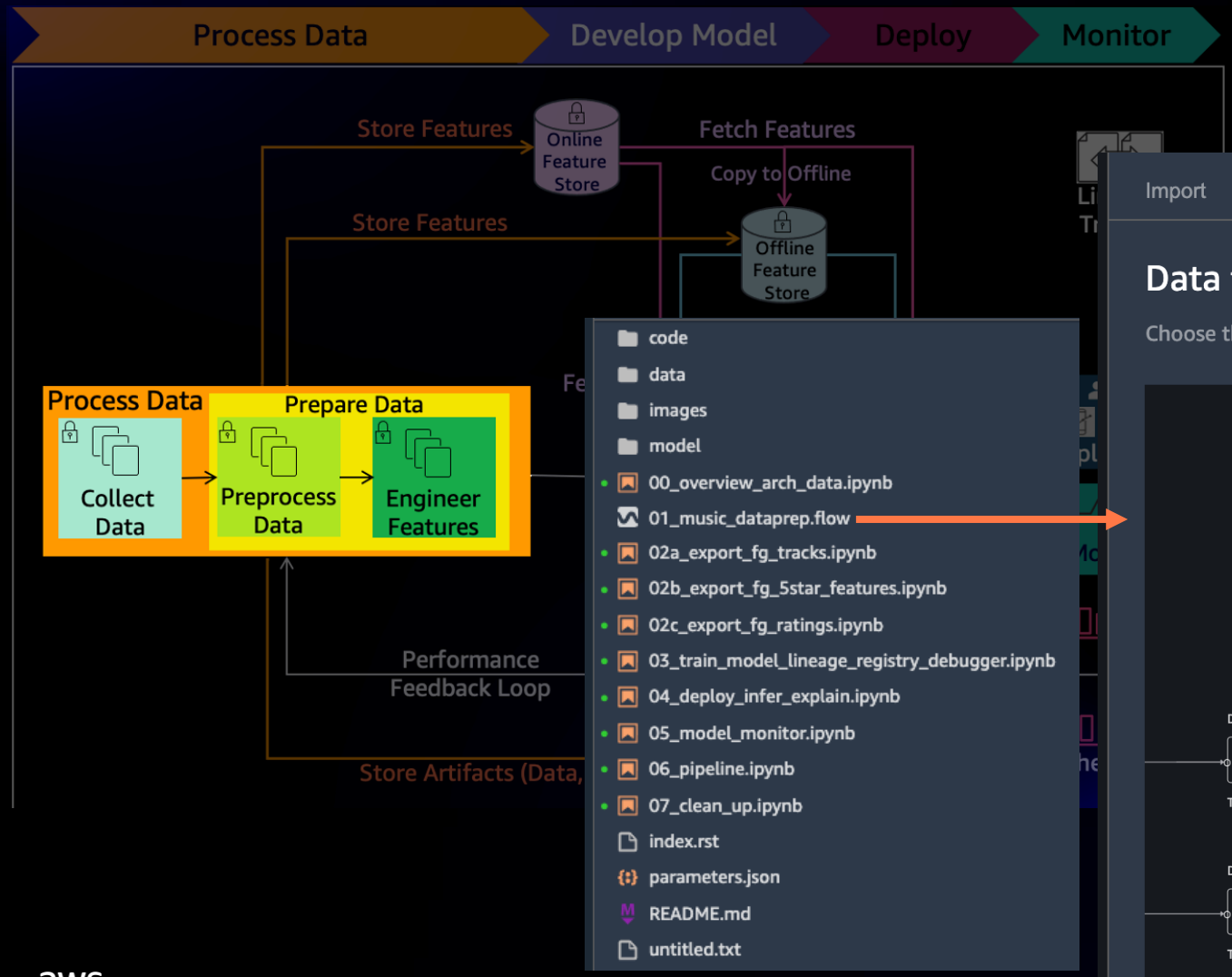
	trackId	length	energy	acousticness	valence	speechiness	instrumentalness	liveness	tempo	ger
0	ef529fb1-37d9-47db-98a9-aeccb75a7481	539.0	0.901144	0.000001	0.104279	0.143046	0.000000	0.021130	107.867115	Ja
1	ec0b785a-e9b0-4573-9823-309aefb99f73	236.0	0.795516	0.470181	0.660053	0.122248	0.935359	0.436054	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

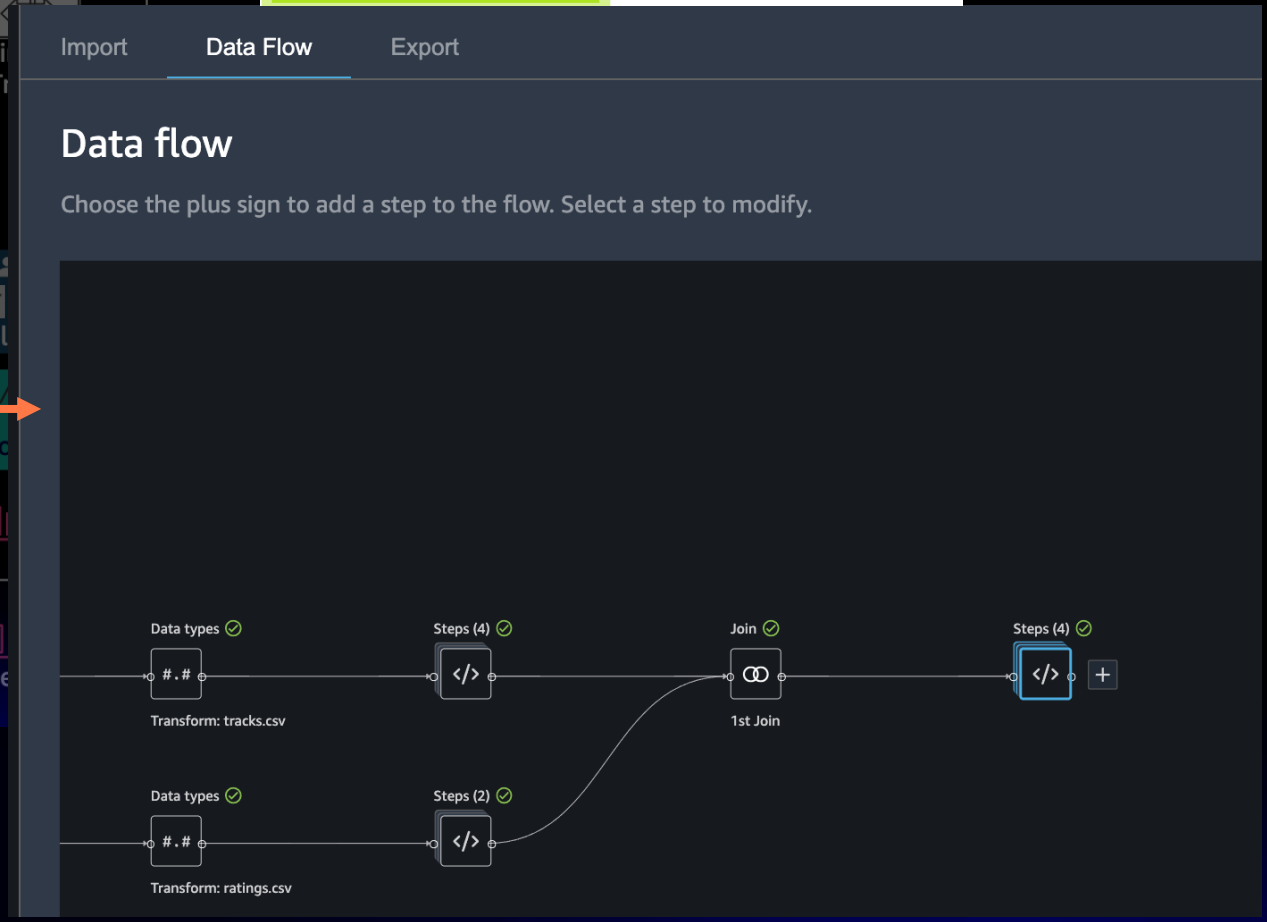
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music Recommendation Example: SageMaker Data Wrangler

Data Preparation (**Wrangling** during Interactive Data Analysis)
Leverage **Visualization** for Exploratory Data Analysis (EDA)

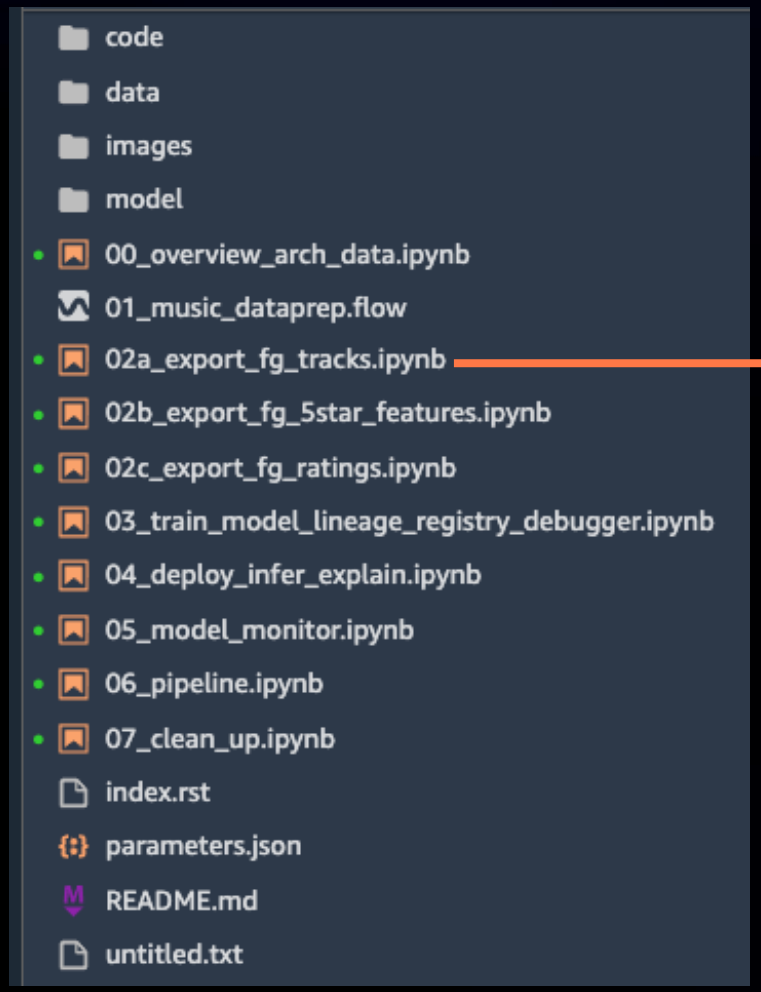
Data Preprocessing	Feature Engineering
Clean (Replace, Impute, Remove Outliers, Duplicates)	Feature Selection
Partition (Train, Validate, Test)	Feature Transformation
Scale (Normalize, Standardize)	Feature Creation (Encoding, Binning)
Unbias, Balance (Detection & Mitigation)	Feature Extraction (Automated in Deep Learning)
Augment	

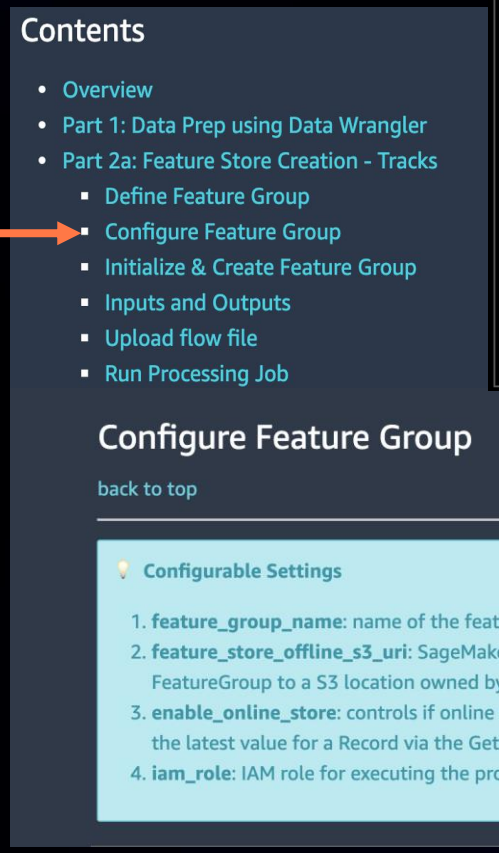


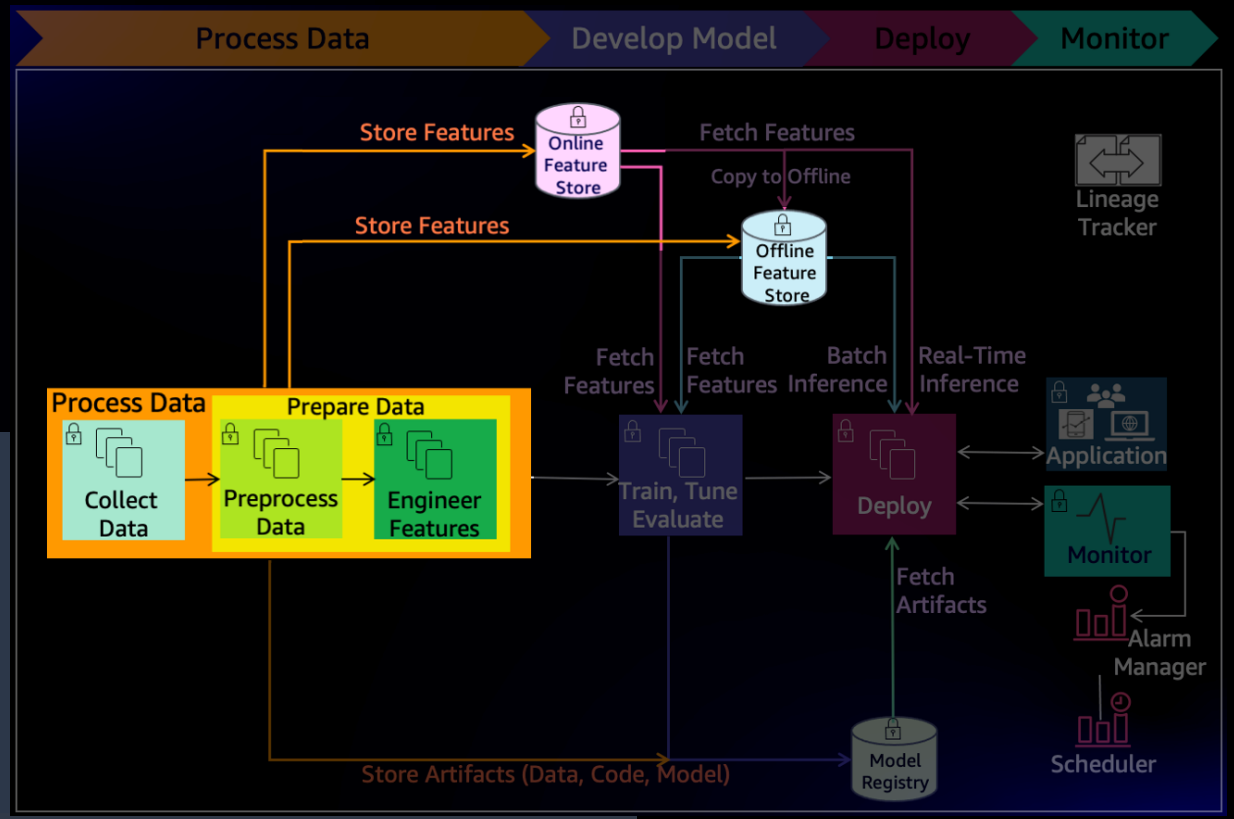


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation example: Configuring the Feature Store

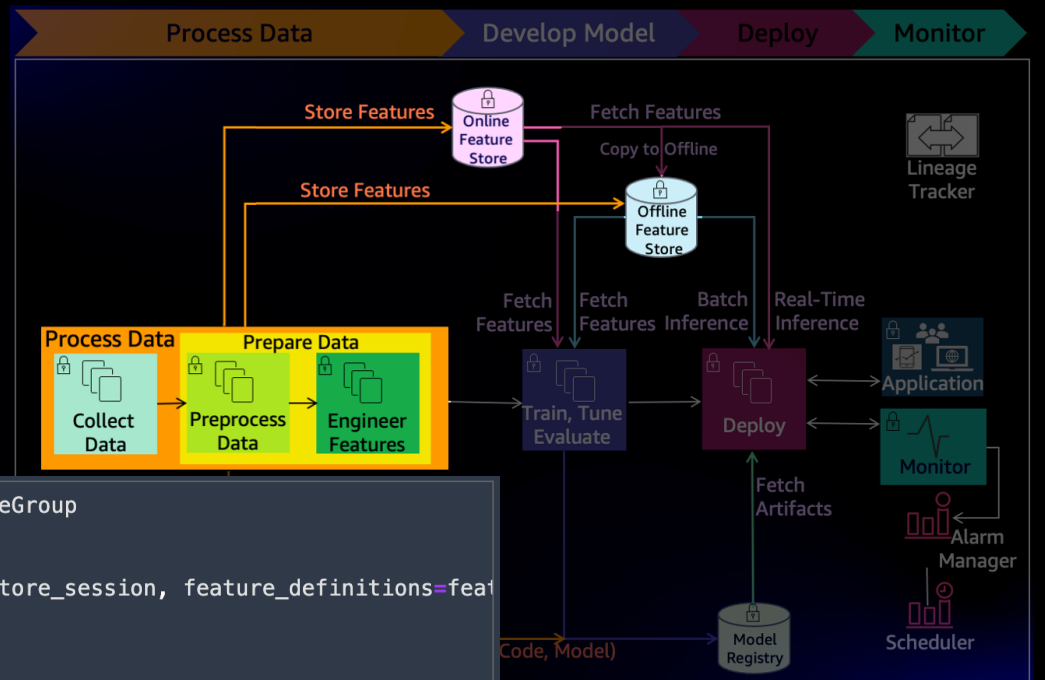






© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation example: Process features and deposit in Feature Store



- 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

```
from sagemaker.feature_store.feature_group import FeatureGroup

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

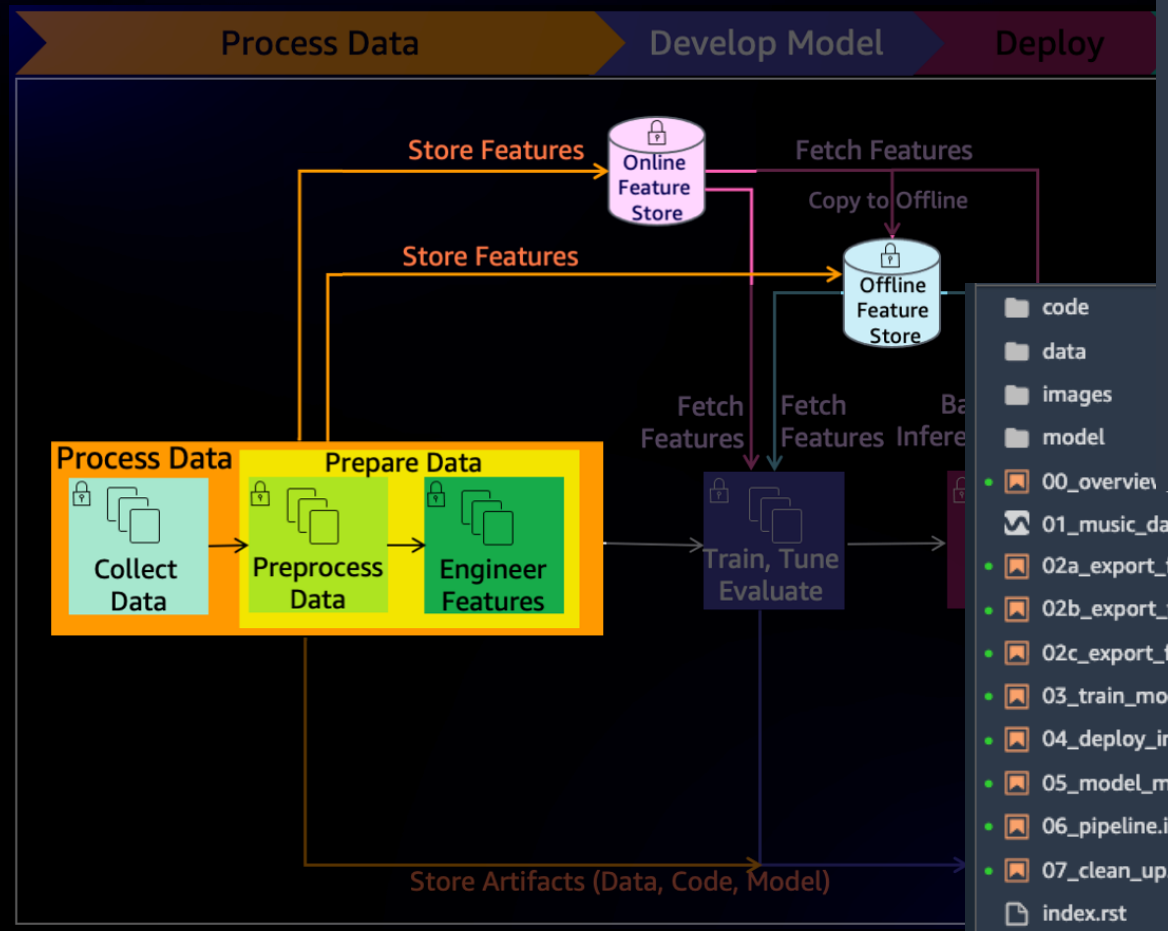
# only create feature group if it doesn't already exist
try:
    sagemaker_client.describe_feature_group(FeatureGroupName=feature_group_name, NextToken='st
    feature_group_exists=True
    print("Feature Group {0} already exists. Using {0}".format(feature_group_name))
except Exception as e:
    error = e.response.get('Error').get('Code')
    if error == "ResourceNotFound":
        feature_group_exists=False
        print("Creating Feature Group {}".format(feature_group_name))
        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))
```

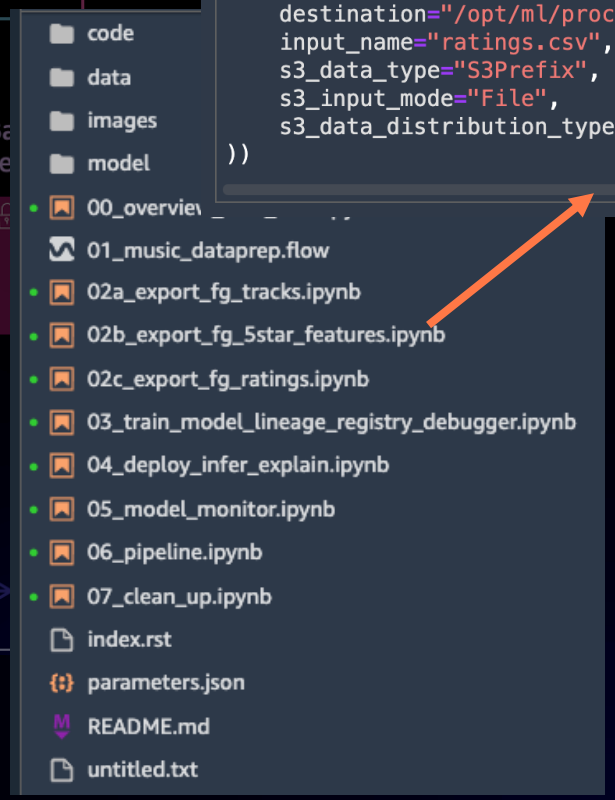
```
Feature Group user-5star-track-features-music-rec already exists. Using user-5star-track-feat
ures-music-rec
```

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.

Music recommendation example: Set up input dataset sources of the processing job





Input - S3 Source: tracks.csv

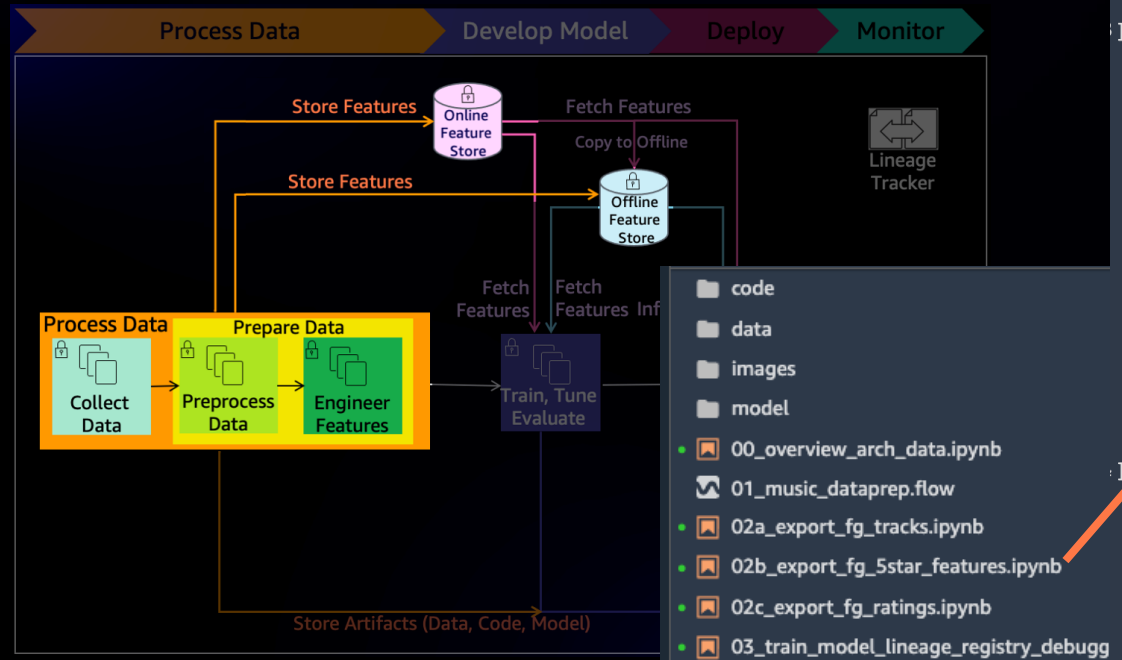
```
data_sources.append(ProcessingInput(
    source=f"{tracks_data_source}", # You could override this to point to another dataset on S
    destination="/opt/ml/processing/tracks.csv",
    input_name="tracks.csv",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"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

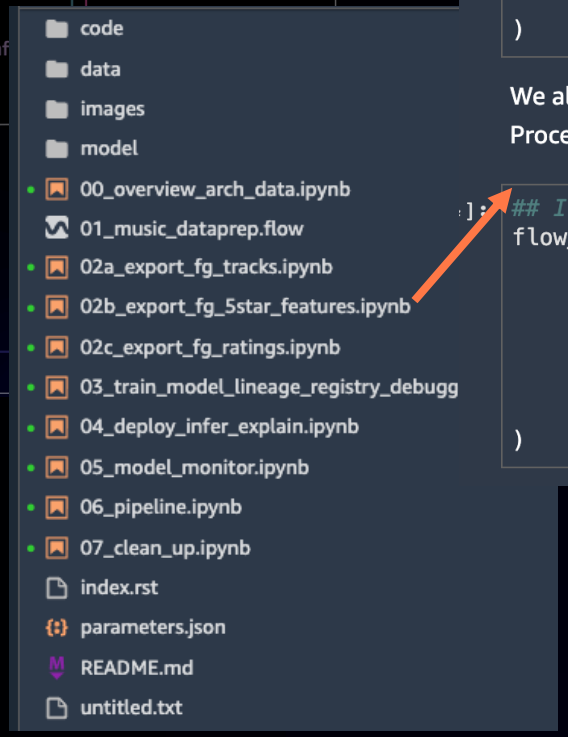
Input - S3 Source: ratings.csv

```
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.

Music recommendation example: 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),
)
```

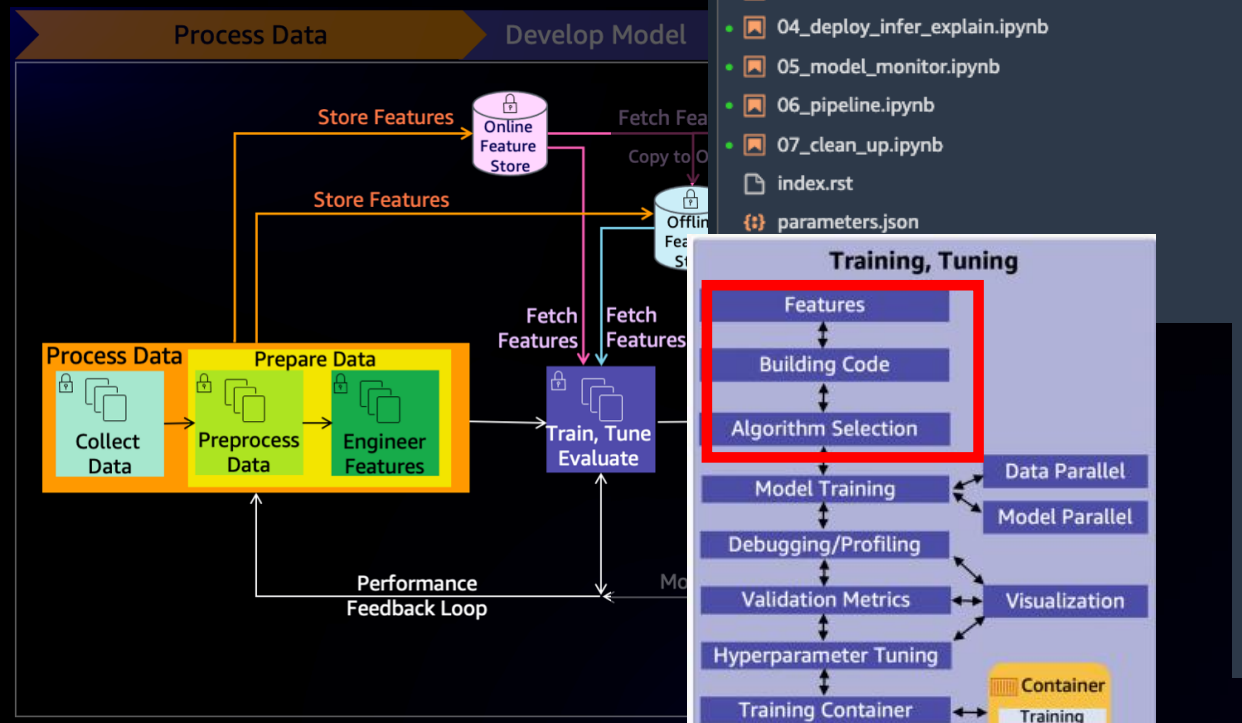
We already uploaded our flow file in the `02a` notebook. Here the Data Wrangler Flow is also provided to the Processing Job as an input source which we configure below.

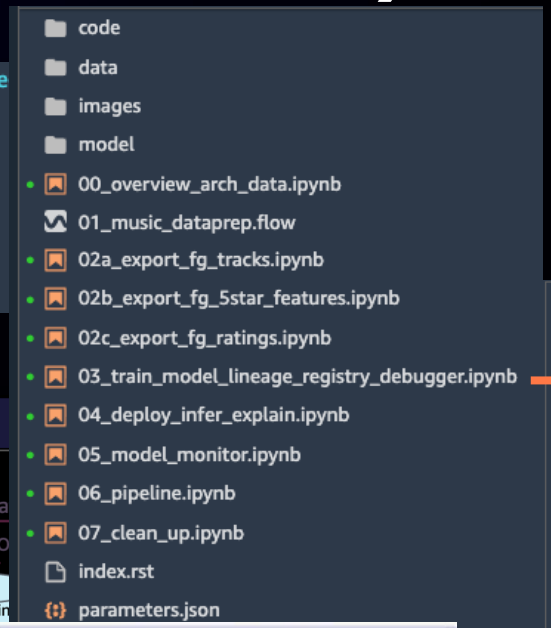
```
## Input - Flow: 01_music_dataprep.flow
flow_input = ProcessingInput(
    source=flow_s3_uri,
    destination="/opt/ml/processing/flow",
    input_name="flow",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
)
```

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

Music recommendation example: Model training (fetch features)

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






```python
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)

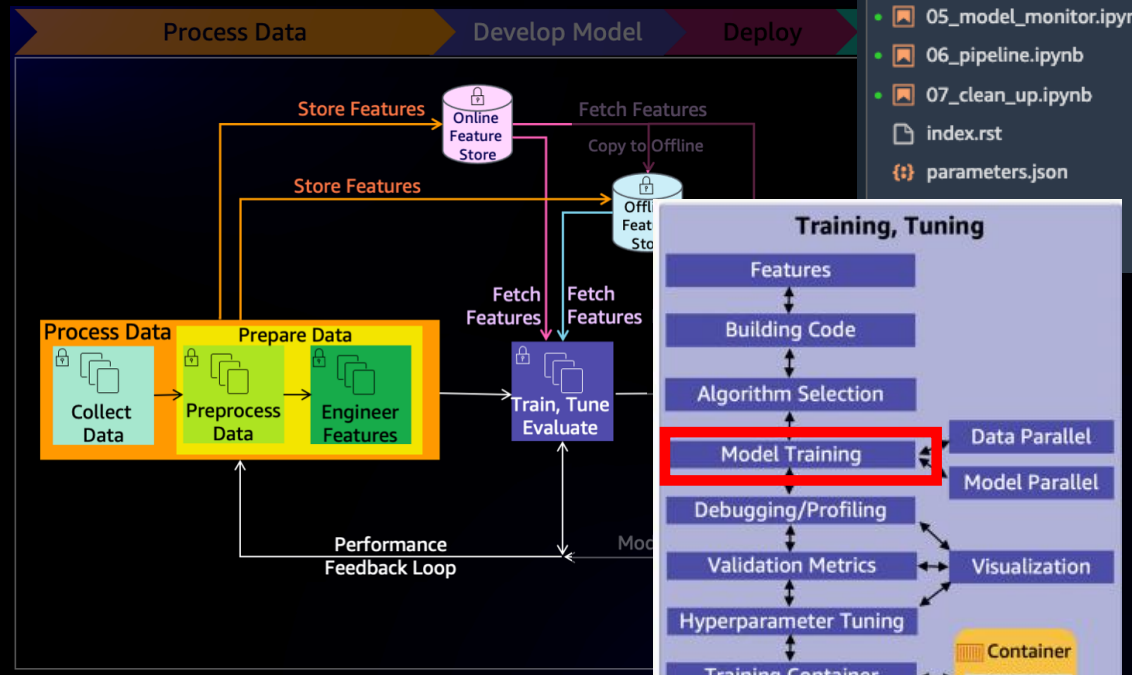
    # split data
    from sklearn.model_selection import train_test_split
    train, val = train_test_split(dataset, test_size=0.2, random_state=42)
    print("Training dataset shape: {}\nValidation dataset shape: {}\n".format(train.shape, val.shape))

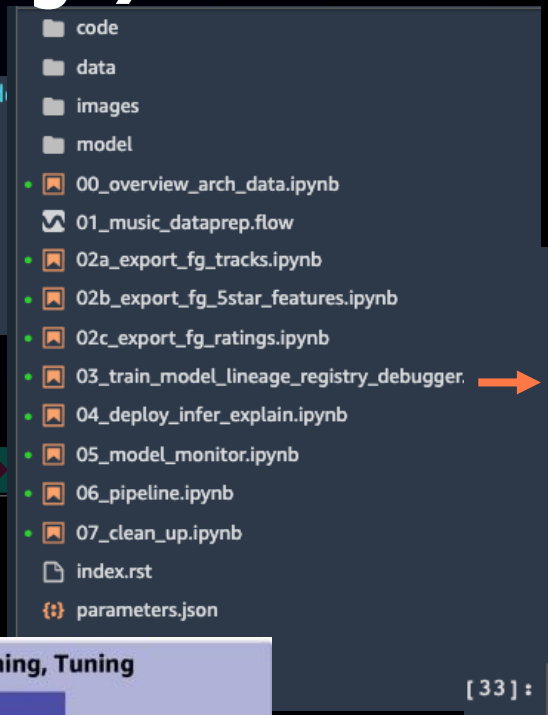
    return train, val
```

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

Music recommendation example: Model training (set lineage)

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





Examine Lineage

back to top

Though you already know the training job details from above, if we were just given the m
Lineage to produce the training job details which produced the model.

Data Lineage and Metrics for Best Model

```
[33]: from sagemaker.lineage import context, artifact, association, action
```

Training data artifact

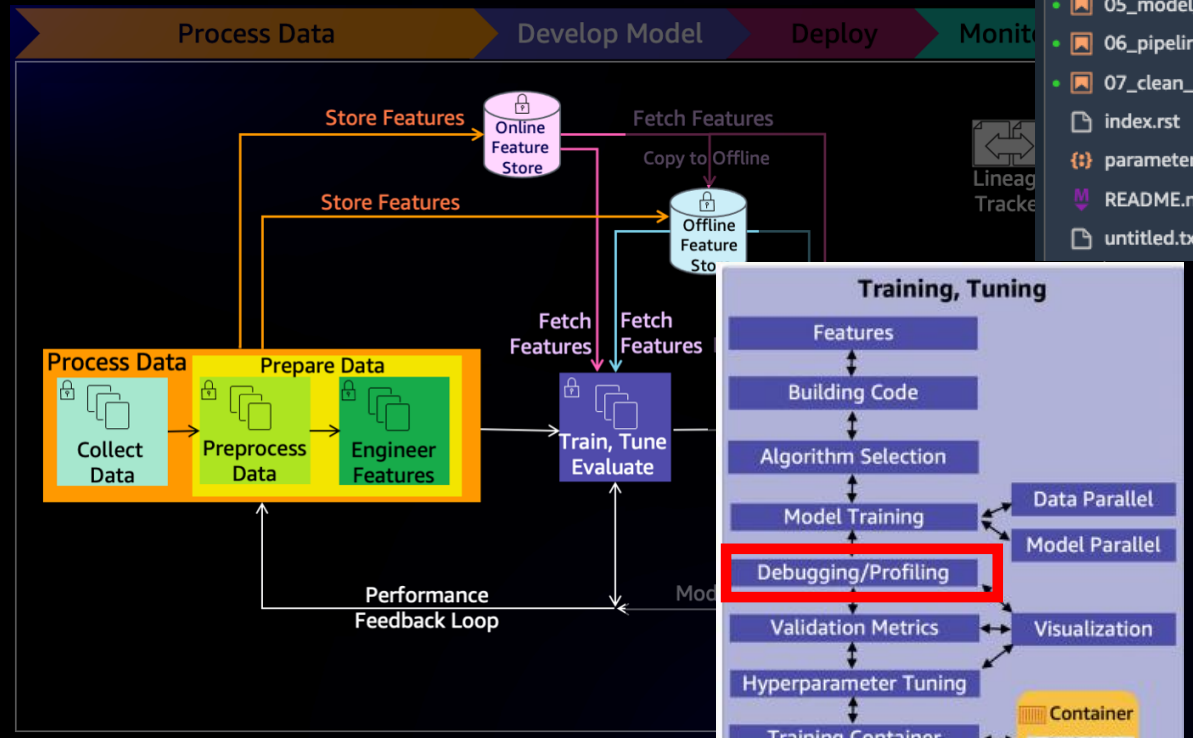
```
[34]: data_artifact_list = []
for data_input in training_job_info['InputDataConfig']:
    channel = data_input['ChannelName']
    data_s3_uri = data_input['DataSource']['S3DataSource']['S3Uri']

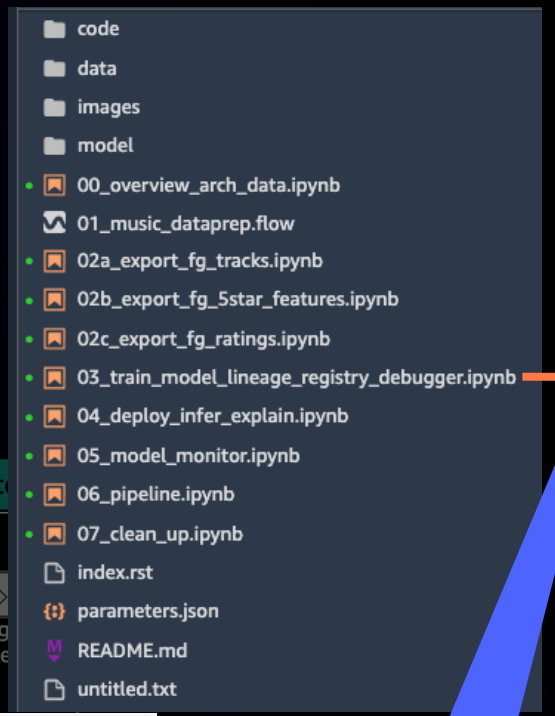
    matching_artifacts = list(artifact.Artifact.list(
        source_uri=data_s3_uri,
        sagemaker_session=sagemaker_session)
    )
```

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

Music recommendation example: Model training (Amazon SageMaker Debugger)

- 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
  - Set Lineage Artifacts
  - Register Model





An estimator with debugger hooks and rules

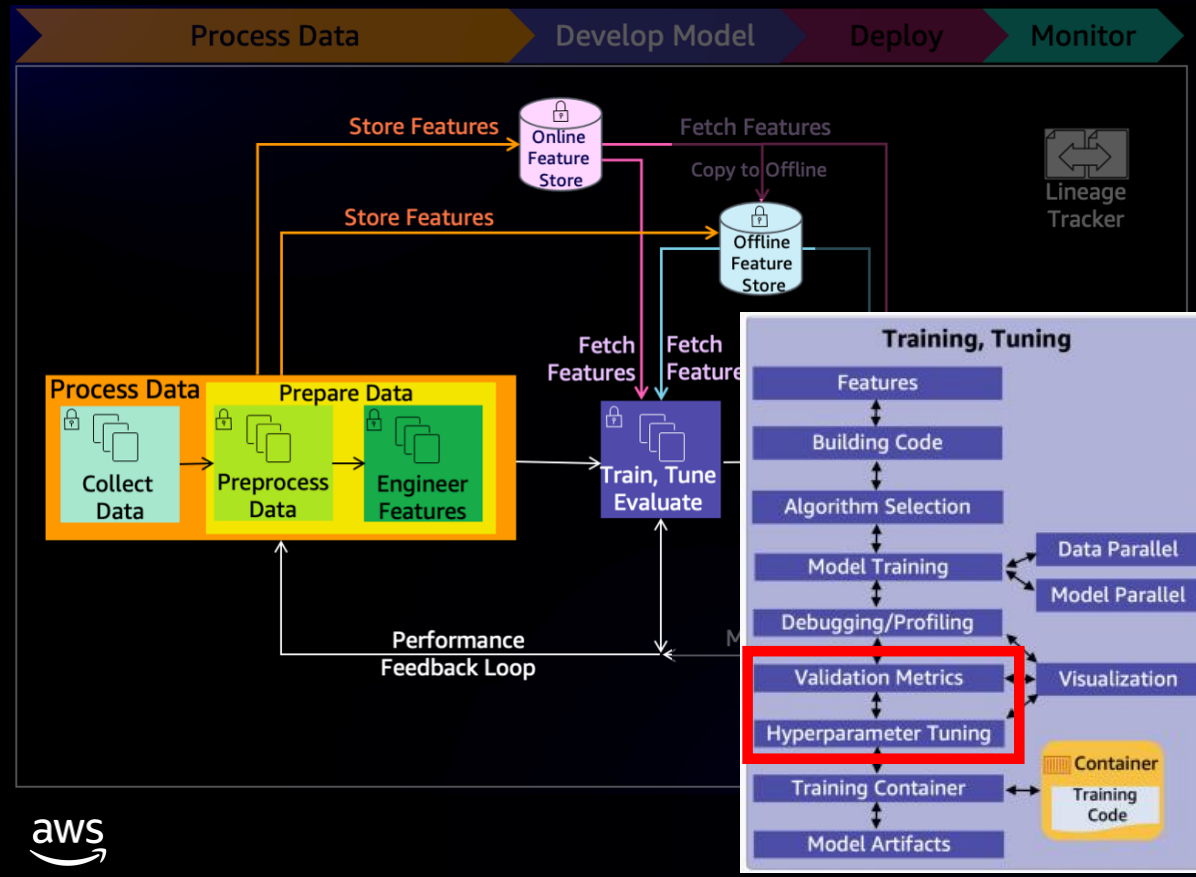
```
xgb_estimator = Estimator(
    role=sagemaker_role,
    instance_count=train_instance_count,
    instance_type=train_instance_type,
    image_uri=image,
    hyperparameters=hyperparameters,
    base_job_name=model_name,
    output_path=estimator_output_path,

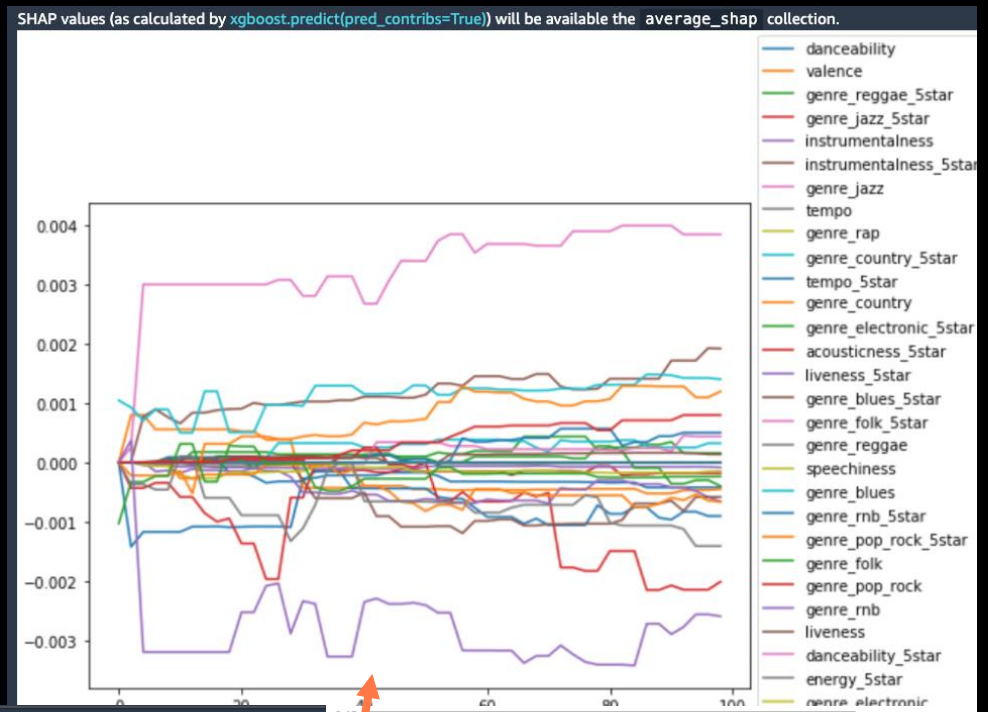
    debugger_hook_config=DebuggerHookConfig(
        s3_output_path=estimator_output_path+'/debugger',
        collection_configs=[
            CollectionConfig(
                name="metrics",
                parameters={
                    "save_interval": str(save_interval)
                }
       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

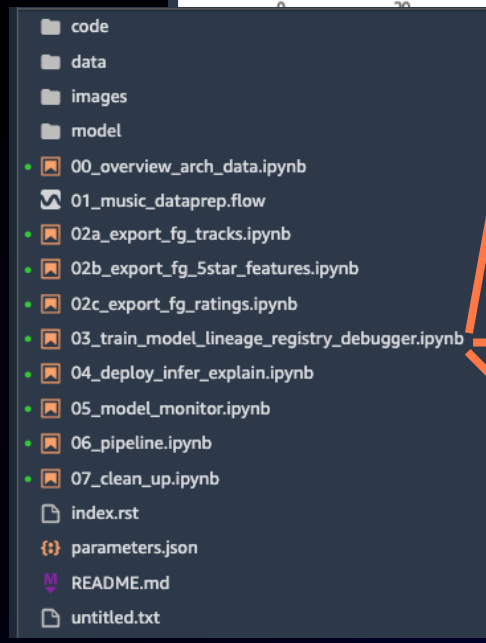
    rules=[
        Rule.sagemaker(
            rule_configs.loss_not_decreasing(),
```

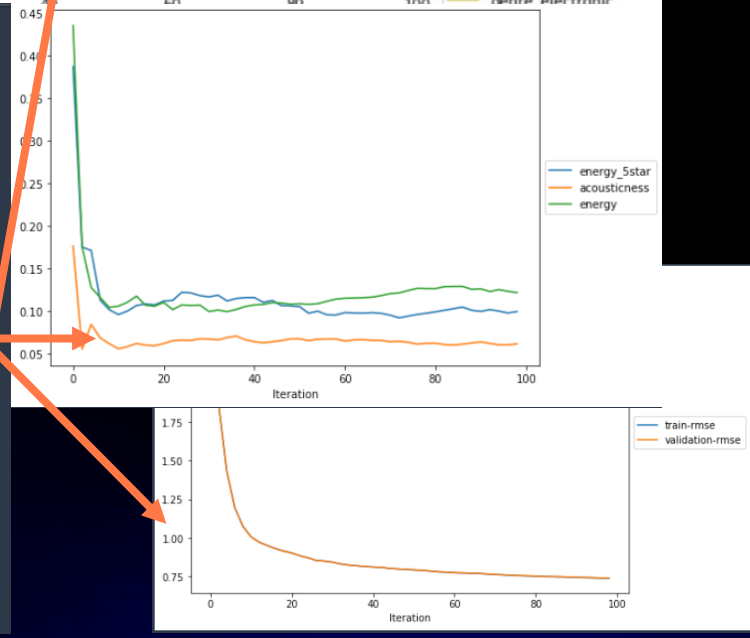
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation example: Model tuning, evaluation (explainability, feature engineering)









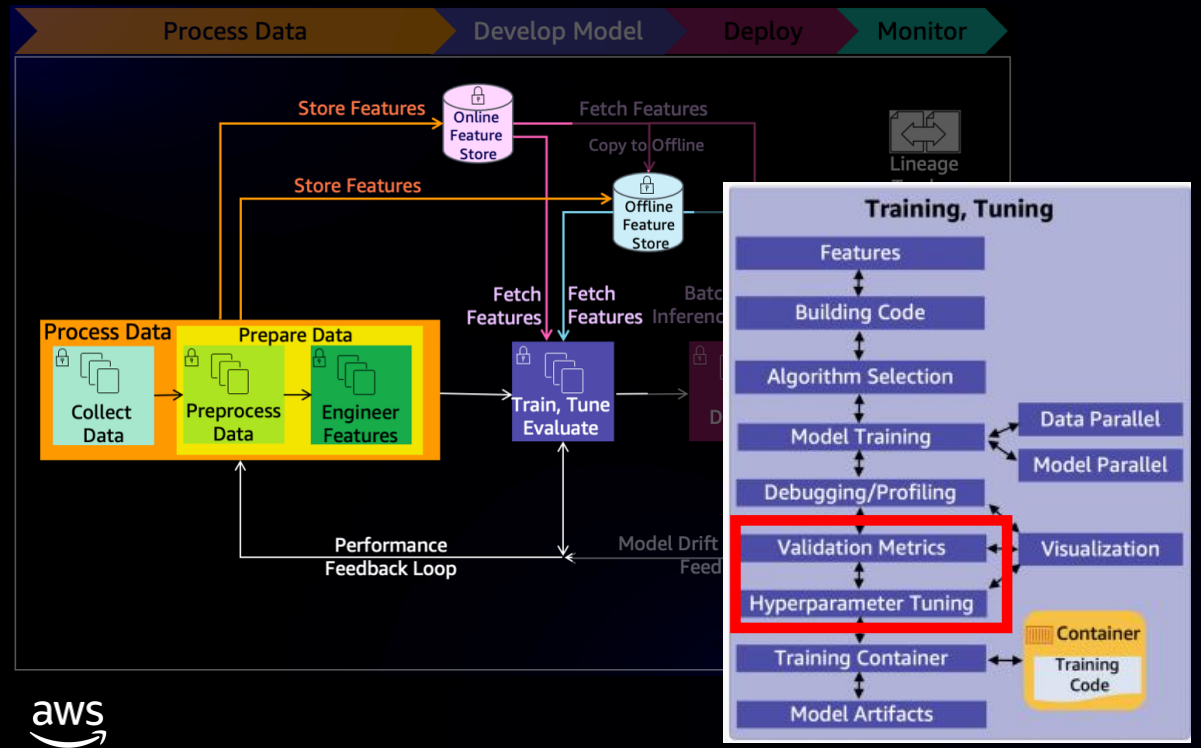
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

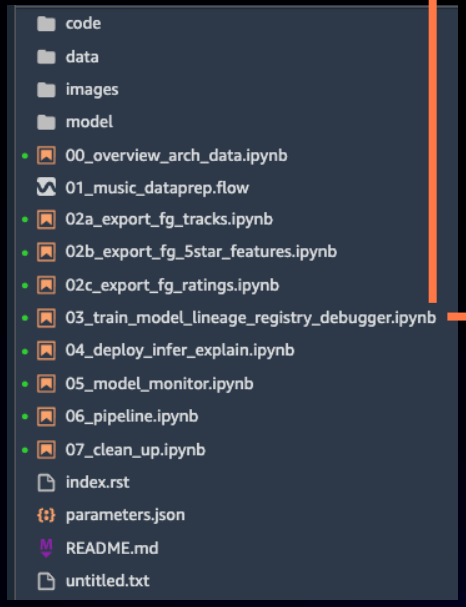
Music recommendation example: Model tuning, evaluation (explainability, feature engineering)

```
import shap

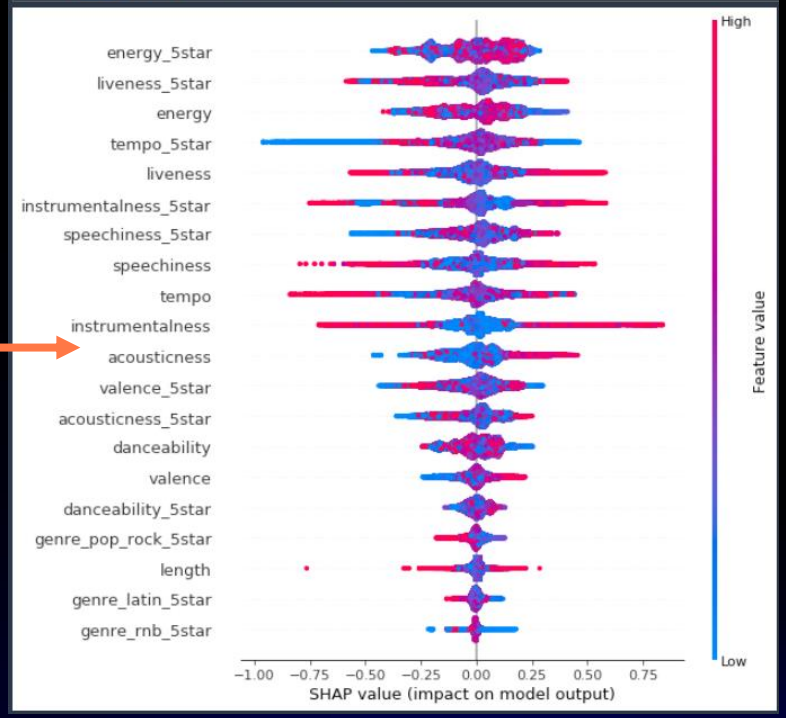
shap_values = trial.tensor("full_shap/f0").value(trial.last_complete_step)
shap_no_base = shap_values[:, :-1]
shap_base_value = shap_values[0, -1]
shap.summary_plot(shap_no_base, plot_type='bar', feature_names=feature_names)
```





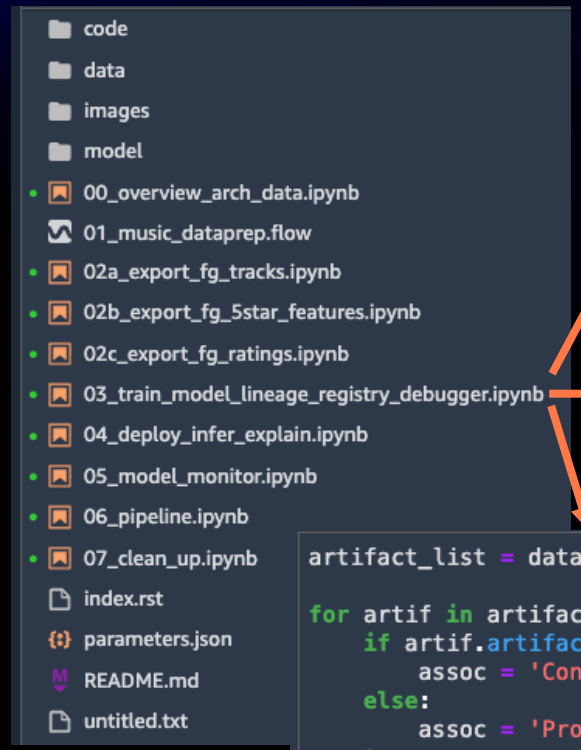



```
X_train = train.iloc[:, 1:]
shap.summary_plot(shap_no_base, X_train)
```



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

Music recommendation example: Lineage for data, algorithm code, and models

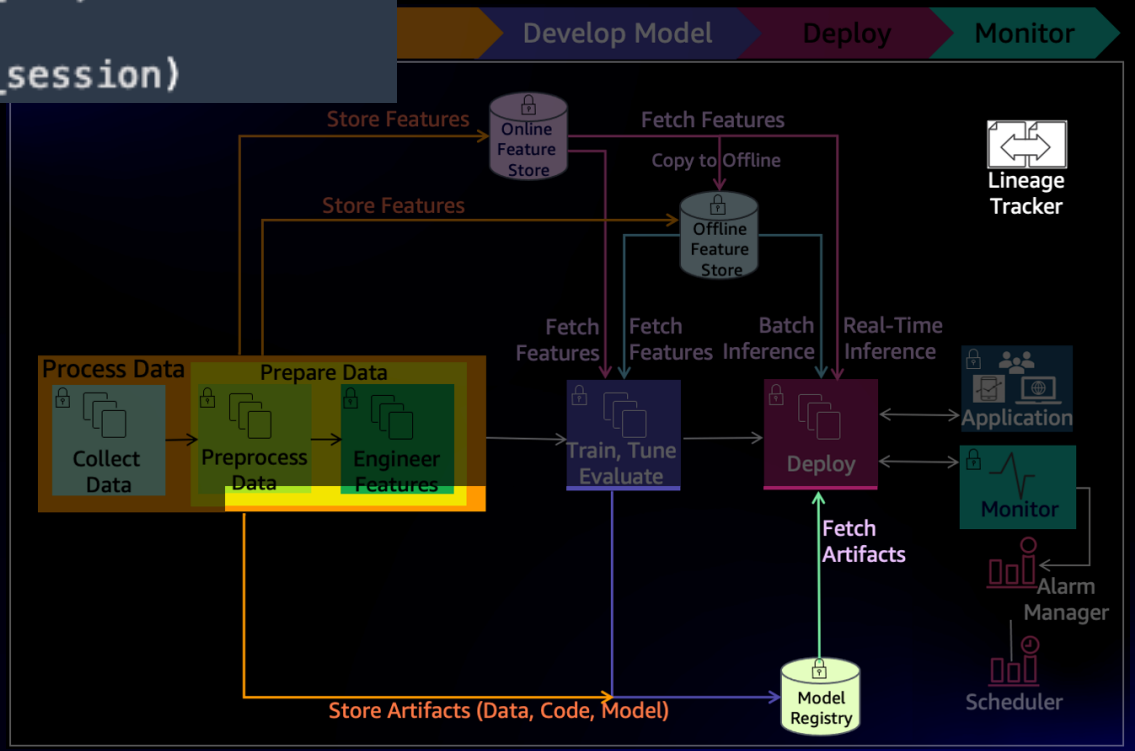



```
data_artifact = artifact.Artifact.create(
    artifact_name=channel,
    source_uri=data_s3_uri,
    artifact_type='Dataset',
    sagemaker_session=sagemaker_session)
print(f'Create artifact {data_artifact.artifact_arn}: SUCCESSFUL')
```

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

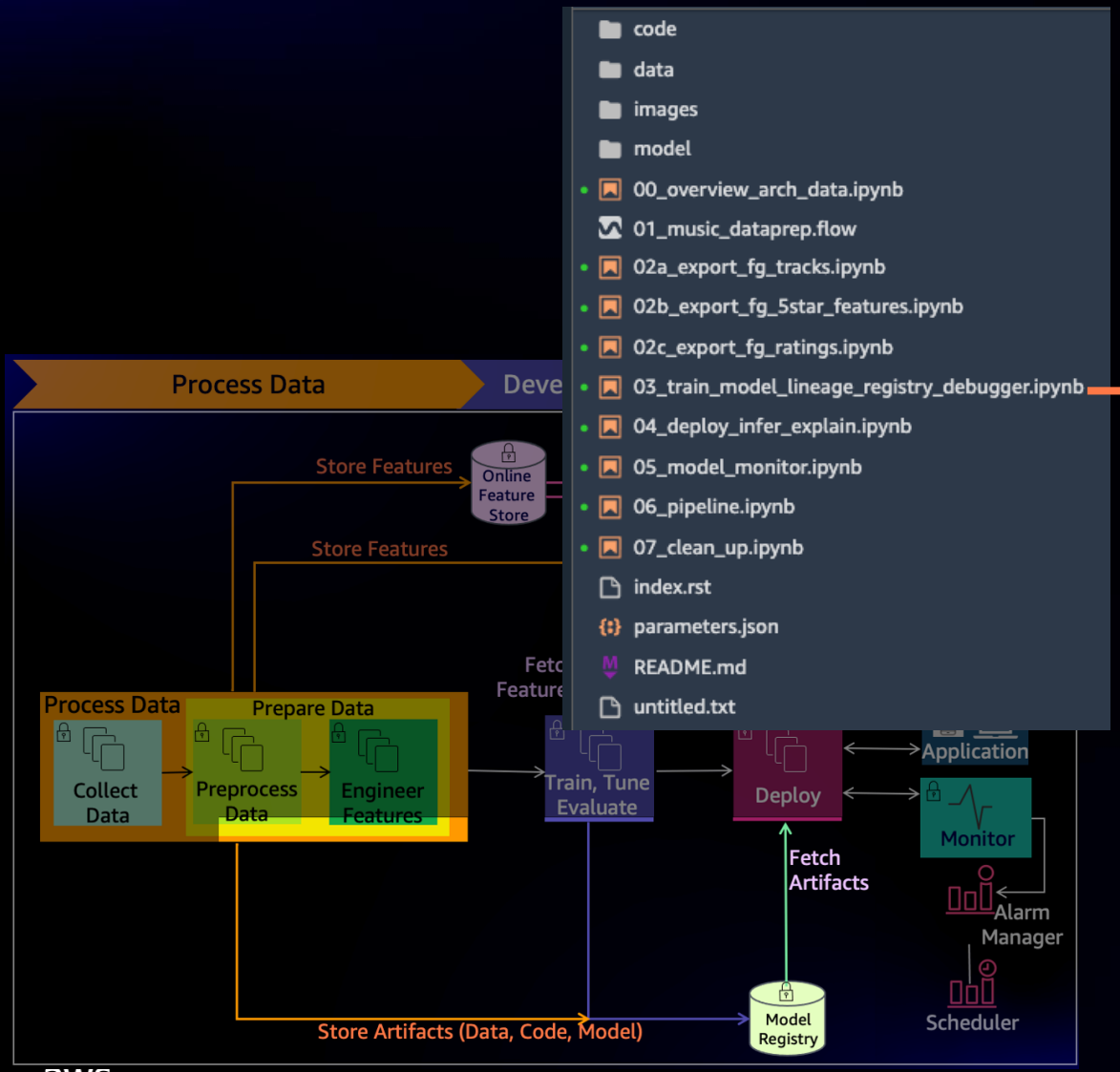
```
artifact_list = data_artifact_list + [model_artifact]

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 example: Lineage for data, algorithm code, and models




```
sys.path.insert(1, './code')
from inference_specification import InferenceSpecification

model_uri = training_job_info.get('ModelArtifacts', {}).get('S3ModelArtifacts')
training_image = training_job_info['AlgorithmSpecification']['TrainingImage']

mp_inference_spec = InferenceSpecification().get_inference_specification_dict(
    ecr_image=training_image,
    supports_gpu=False,
    supported_content_types=['text/csv'],
    supported_mime_types=['text/csv'])

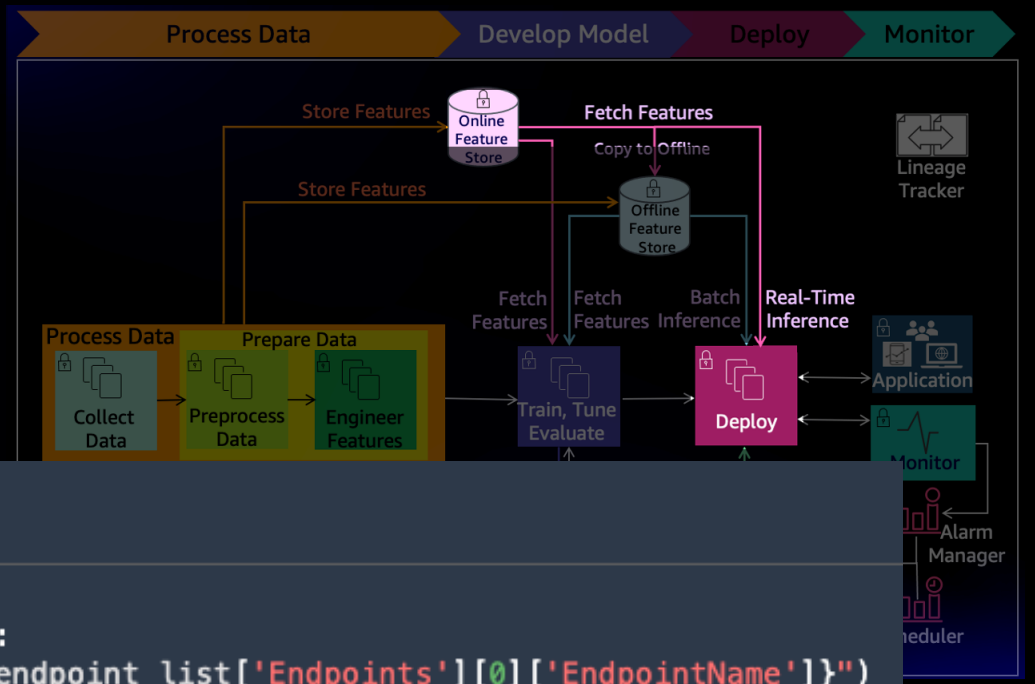
mp_inference_spec['InferenceSpecification']['Containers'][0]['ModelDataUrl'] = model_uri
mp_input_dict = {
    'ModelPackageGroupName': mpg_name,
    'ModelPackageDescription': 'SageMaker Music Recommender',
    'ModelApprovalStatus': 'PendingManualApproval'
}

mp_input_dict.update(mp_inference_spec)
mp_response = sagemaker_client.create_model_package(**mp_input_dict)

model_packages = sagemaker_client.list_model_packages(ModelPackageGroupName=mpg_name)['Model
model_packages
```

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

Music recommendation example: Create endpoint



- 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

Create endpoint

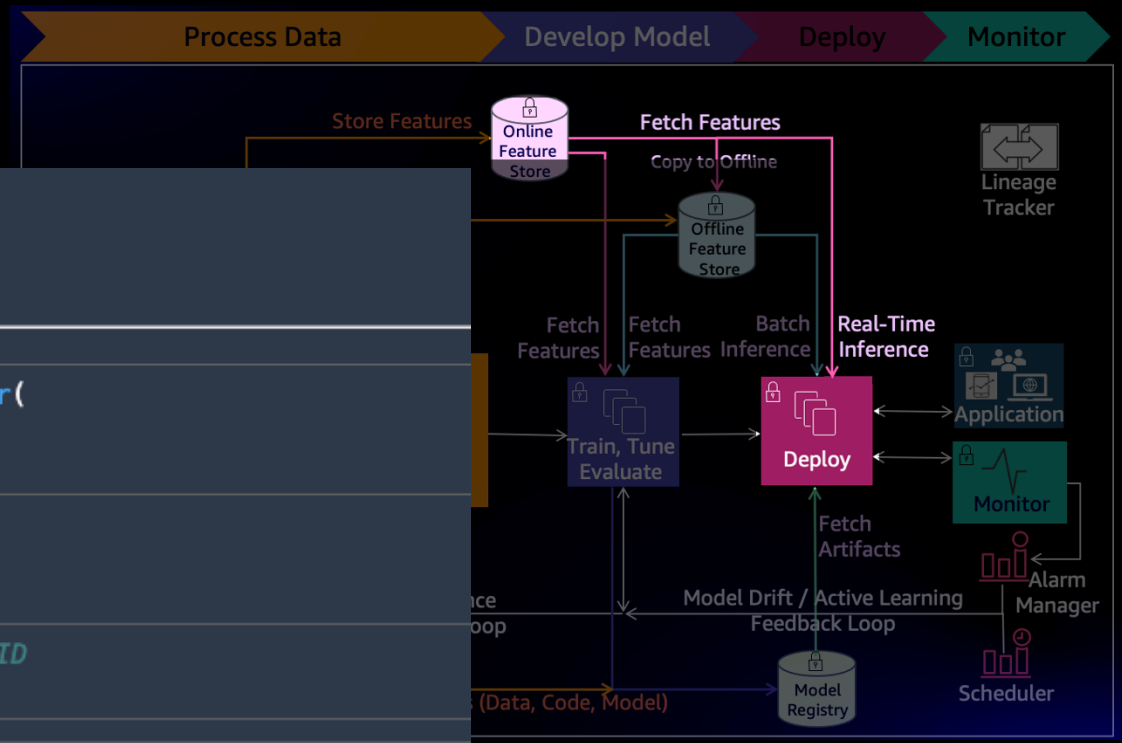
```
%%time
if len(endpoint_list['Endpoints']) > 0:
    print(f"Using existing endpoint: {endpoint_list['Endpoints'][0]['EndpointName']}")
else:
    # deploy endpoint for model if it doesn't already exist
    xgb_estimator.deploy(initial_instance_count=1,
                         instance_type='ml.m4.xlarge',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

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

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

Music recommendation example: Create predictor



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

- 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

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

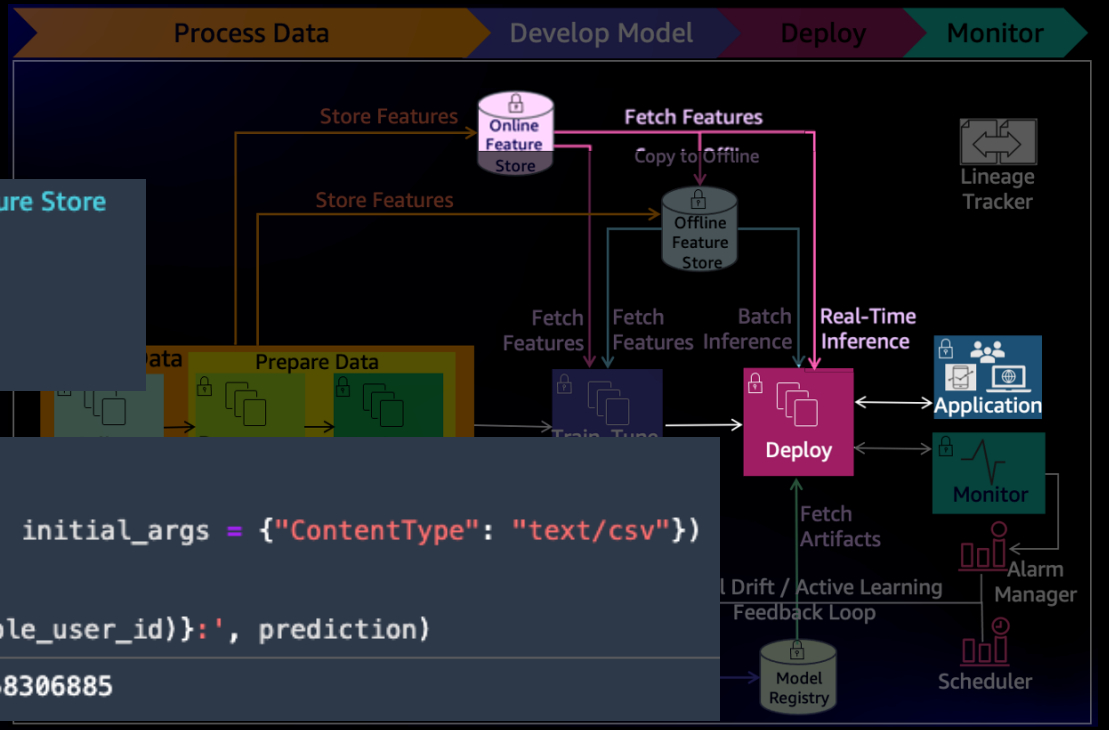
feature_store_session = sagemaker.Session(
    boto_session=boto_session,
    sagemaker_client=sagemaker_client,
    sagemaker_featurestore_runtime_client=featurestore_runtime
)
```

```
# 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')
```

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

Music recommendation example: Real-time inference



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

- 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

```
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
```

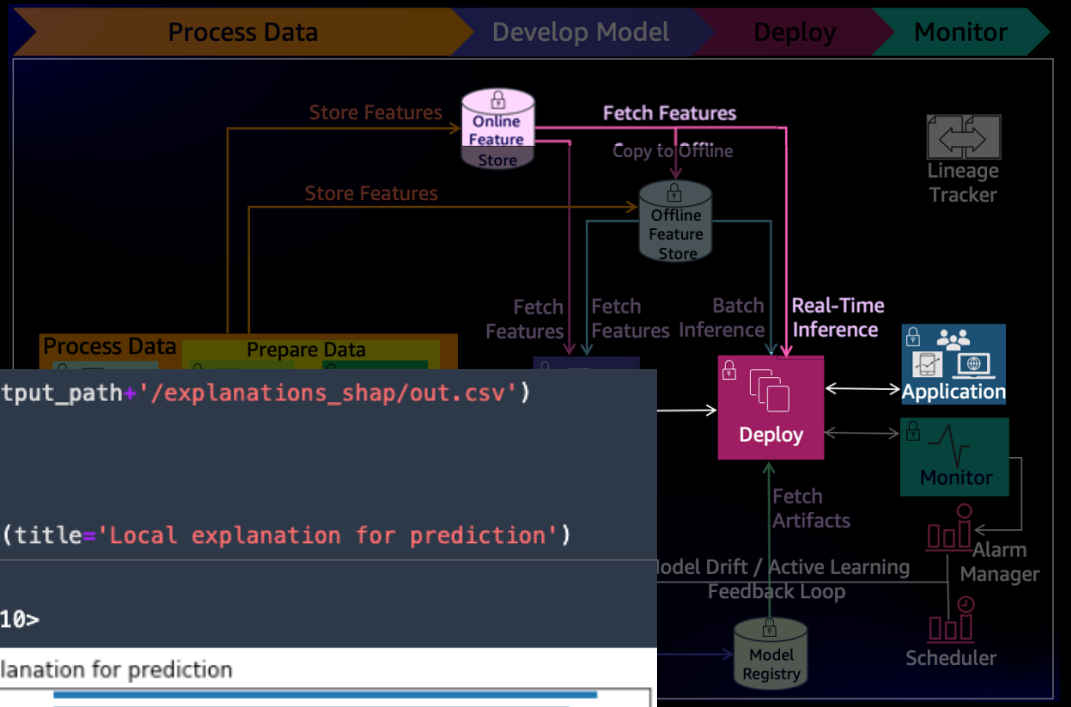
```
playlist_length = 10  # number of songs to recommend in playlist
playlist = inference_df.sort_values(by='rating', ascending=False).head(playlist_length)
print('Curated Playlist:\n', playlist['trackid'])
```

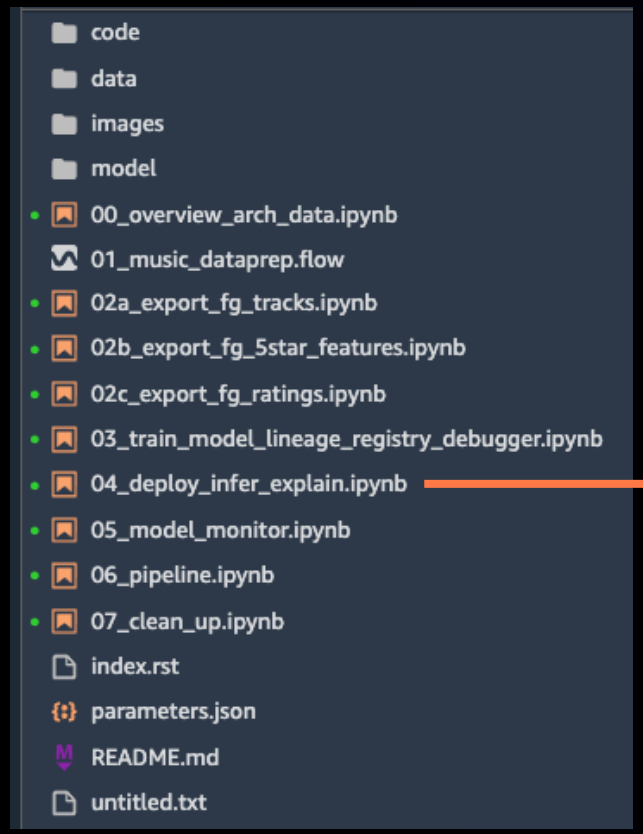
```
Curated Playlist:
 588    ec704261-69fd-4f94-b68b-76346b28bc4e
706    0acc9c7c-e89a-416b-8968-eb650e5a4282
944    9df507e4-9000-4f23-945a-b0684365a0a0
62     369e89cd-f59d-4caf-a29d-d011d8afeb95
75     31bceeb9-89ec-46f9-9640-77bcc0e5f8d1
254    b819a845-7ad8-454a-9f47-1715424957b7
757    4db1bbd8-3771-46ae-9f2e-dece0e8a0e98
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.

Music recommendation example: Model explainability

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

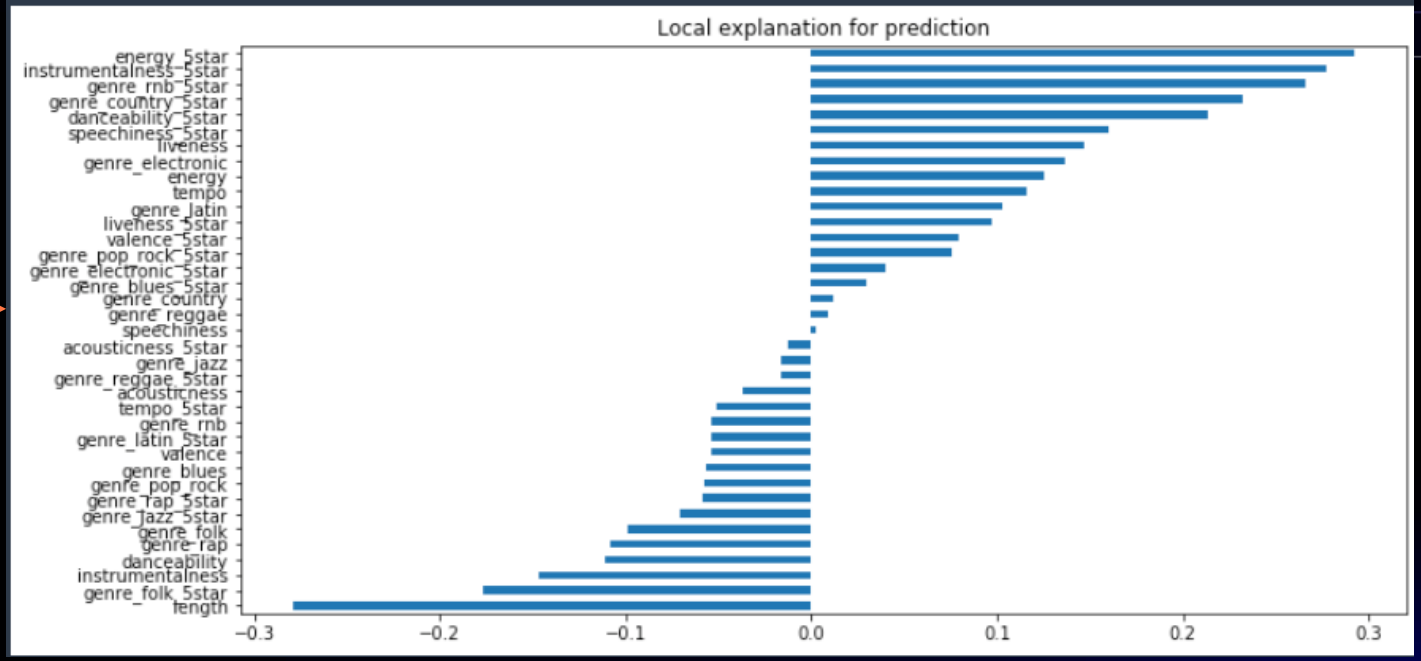





```
local_explanations_out = pd.read_csv(explainability_output_path+'/explanations_shap/out.csv')
local_explanations_out.columns = feature_names

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.

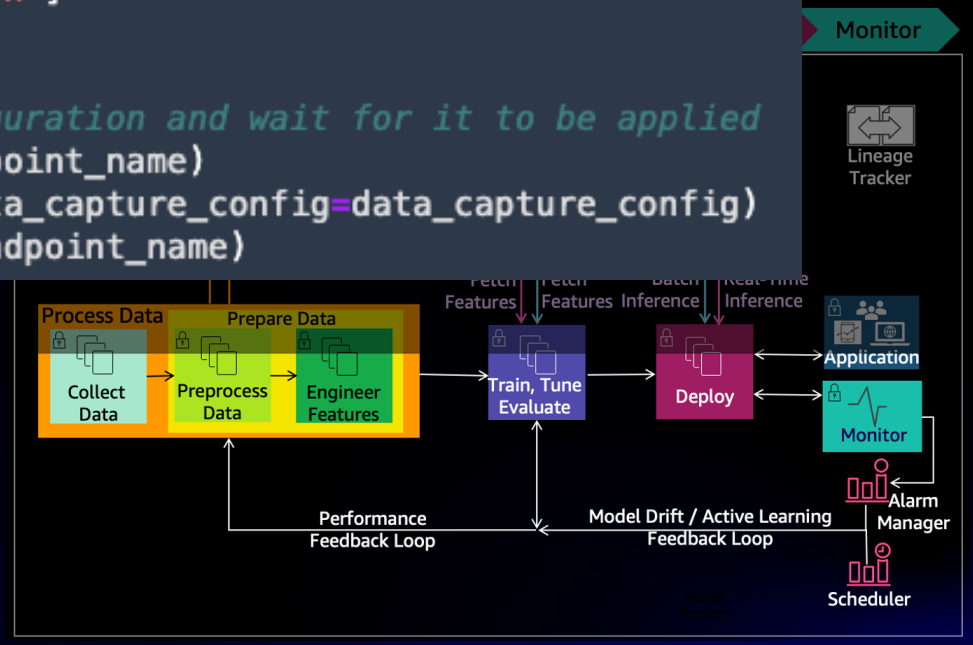
Music recommendation example: Monitor – Enable data capture

- 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

```
%%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"]
)

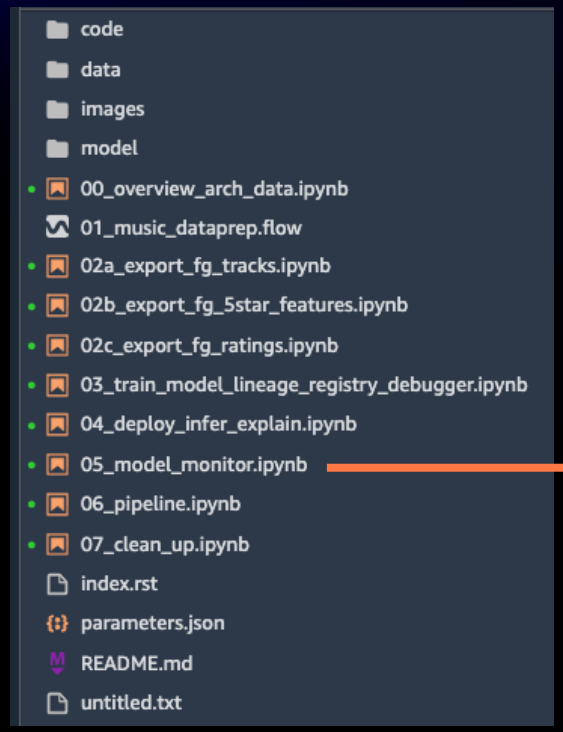
# Now it is time to apply the new configuration and wait for it to be applied
predictor = Predictor(endpoint_name=endpoint_name)
predictor.update_data_capture_config(data_capture_config=data_capture_config)
sm_session.wait_for_endpoint(endpoint=endpoint_name)
```

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



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

Music recommendation example: Monitor – Baselining



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.

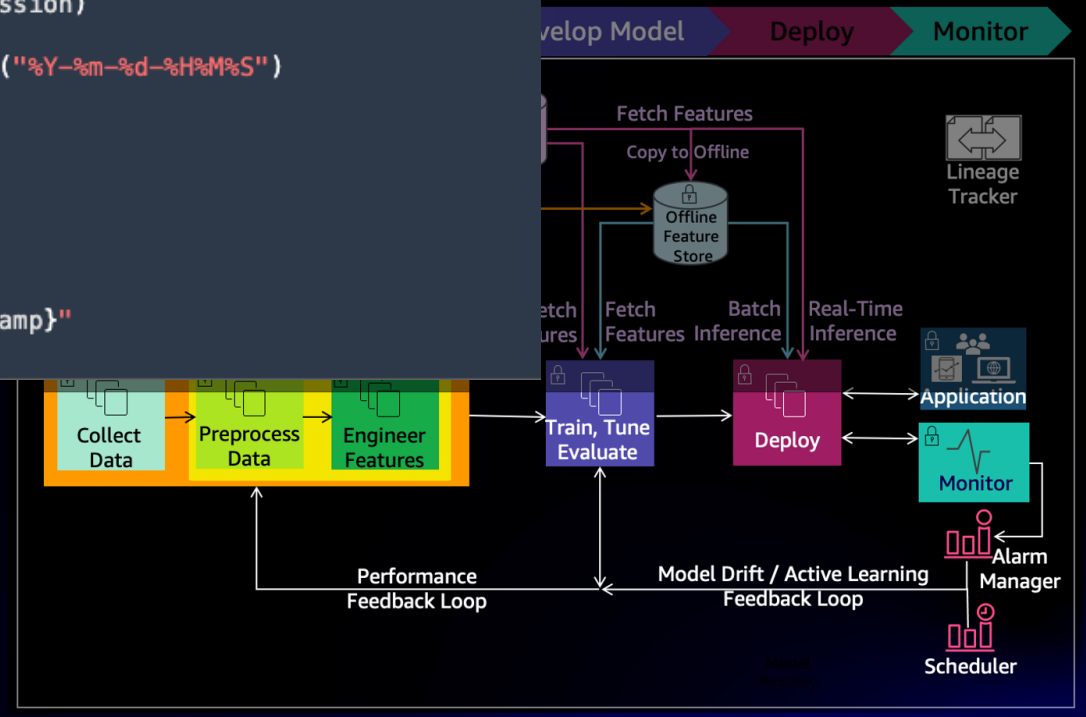
```
]: from sagemaker.model_monitor import DefaultModelMonitor
from sagemaker.model_monitor.dataset_format import DatasetFormat
from sagemaker import get_execution_role
import datetime

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}"
)
```

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



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

Music recommendation example: Monitor – Explore constraints and statistics

- 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

Explore the generated constraints and statistics

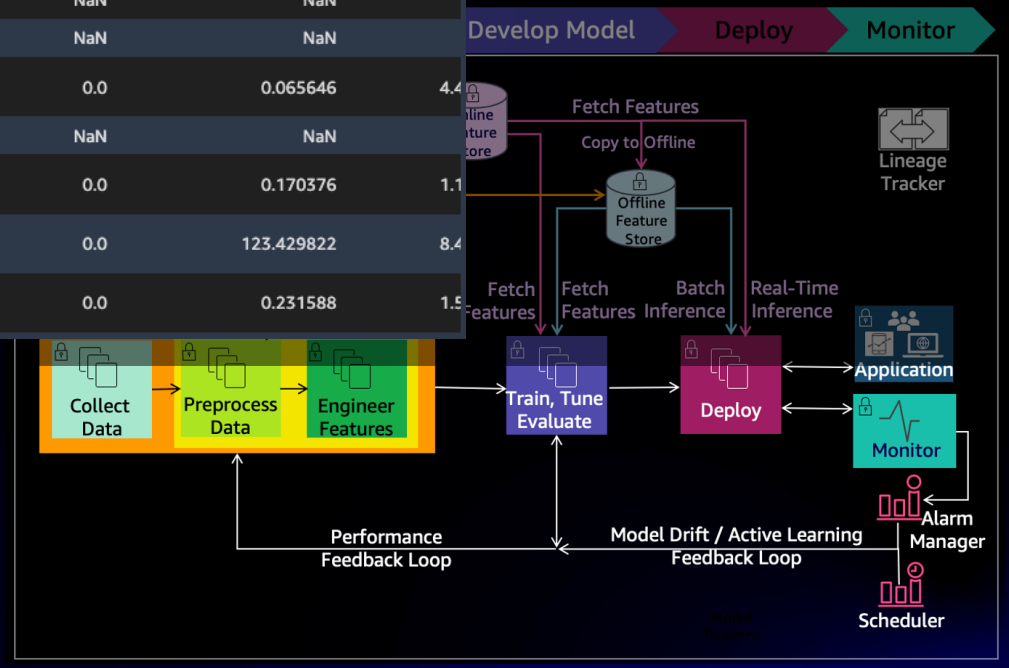
```
import pandas as pd

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.
```

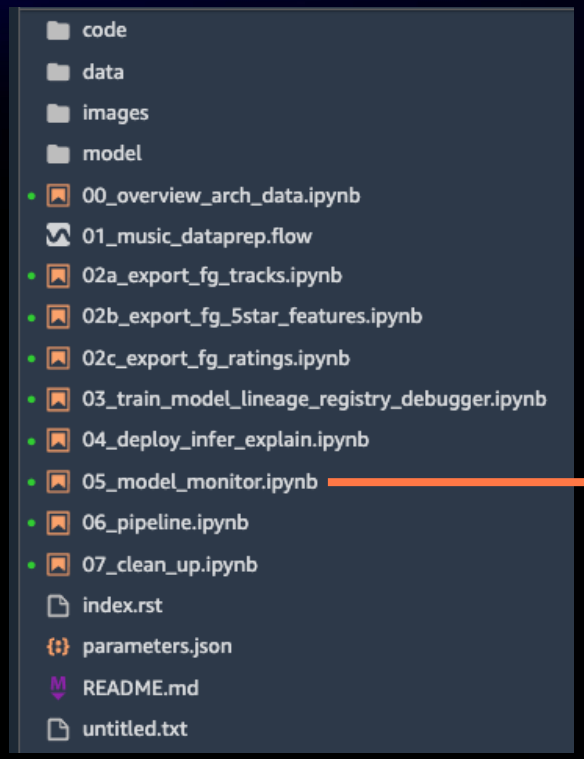
	name	inferred_type	numerical_statistics.common.num_present	numerical_statistics.common.num_missing	numerical_statistics.mean	numerical_st
0	_c0	Fractional	68525.0	0.0	3.469901	2.3
1	_c1	Fractional	68525.0	0.0	248.140752	1.7
2	_c2	String	NaN	NaN	NaN	
3	_c3	String	NaN	NaN	NaN	
4	_c4	String	NaN	NaN	NaN	
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

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



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

Music recommendation example: Monitor – Enable continuous 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).

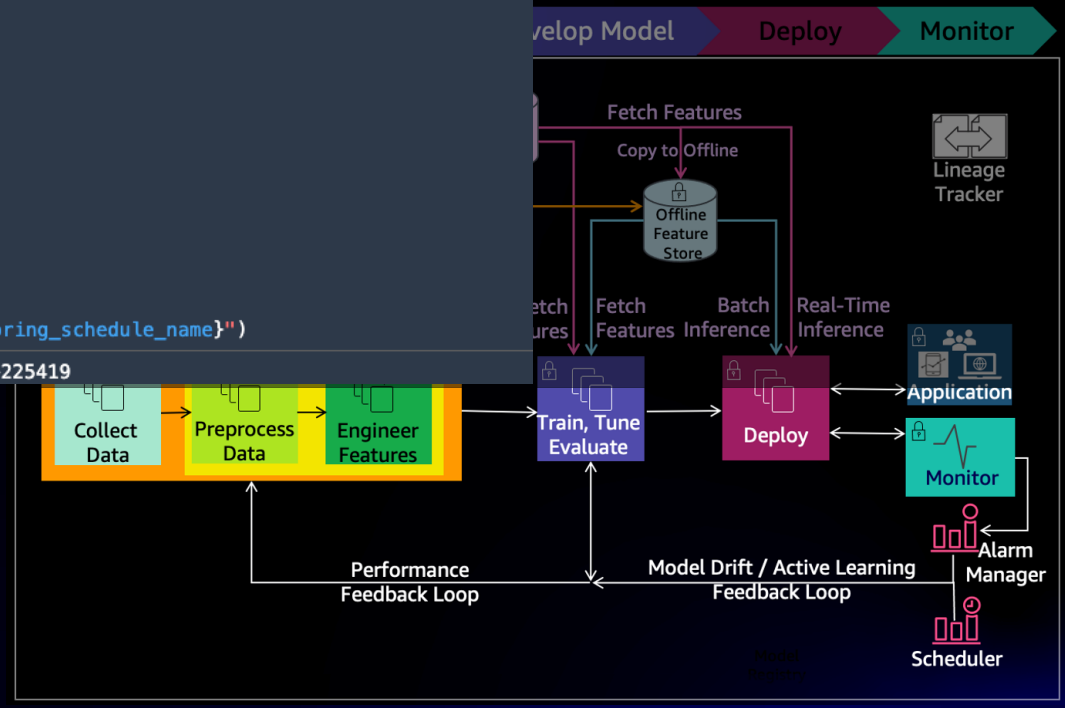
```
from sagemaker.model_monitor import CronExpressionGenerator
import datetime as datetime
from time import gmtime, strftime


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
```

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



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

Music recommendation example: Pipelines (model update, scheduler, data prepare)

- 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

```
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
```

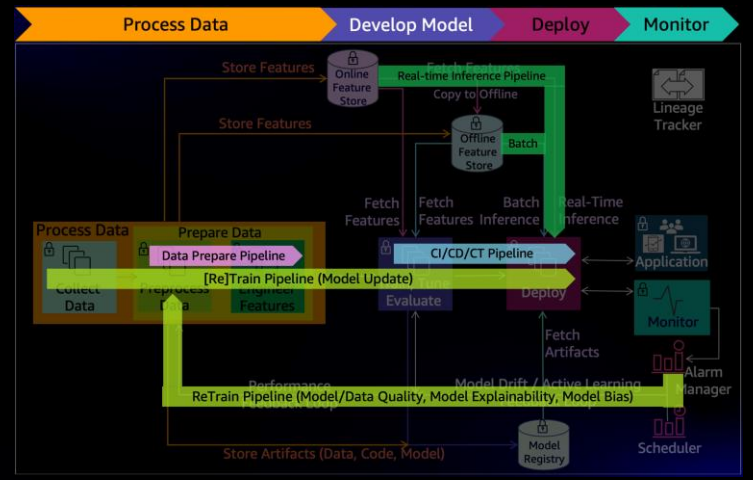
```
### Option 1: The Entire Pipeline End to end
```

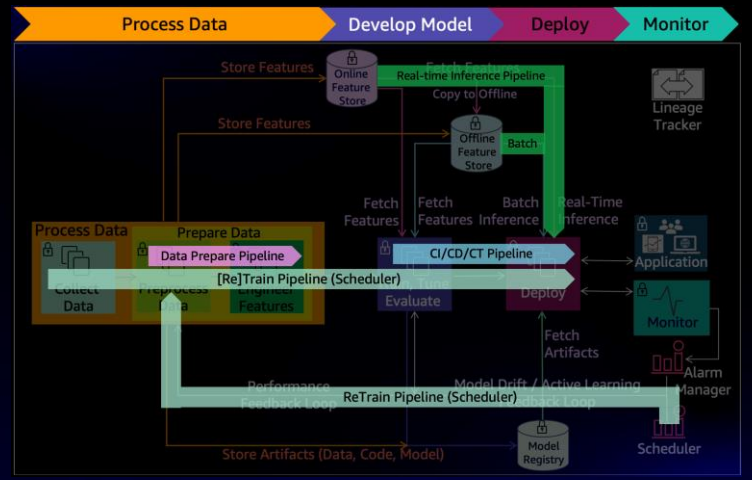
```
pipeline_name = f'MusicRecommendationPipeline'

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

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

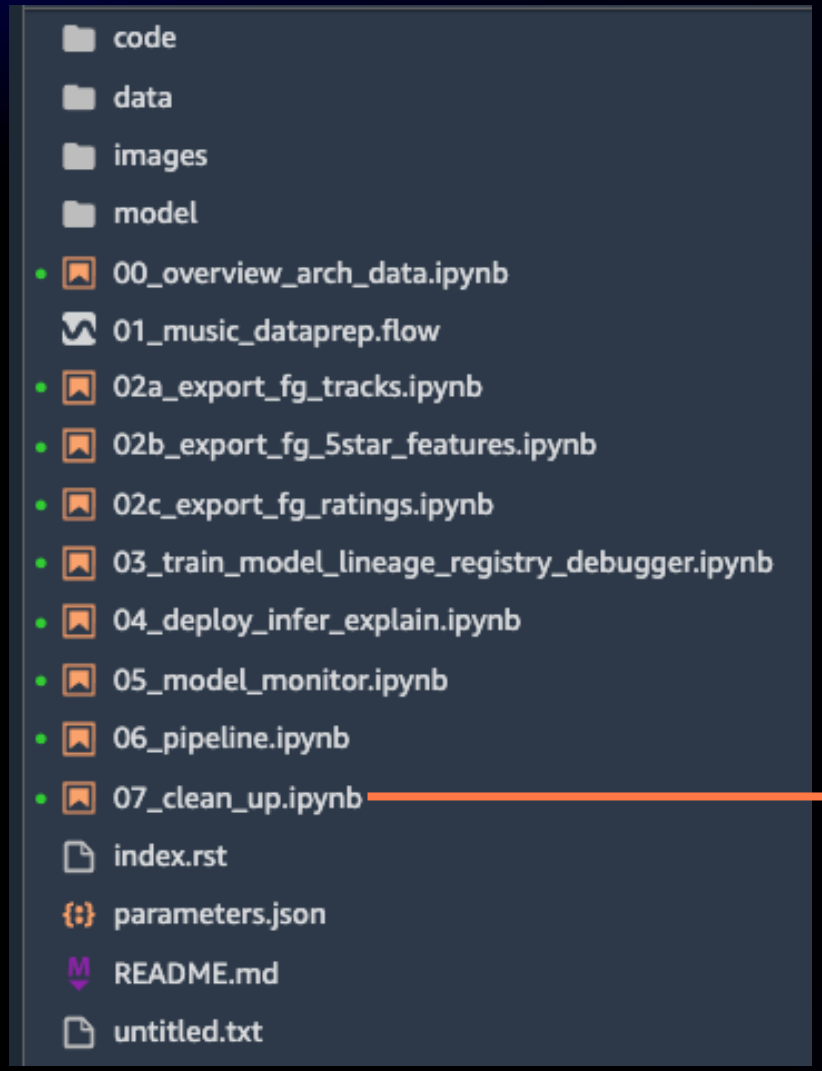
```
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[
        train_instance_param,
        model_approval_status],
    steps=[
        flow_step_tracks,
        flow_step_user_preferences,
        flow_step_ratings,
        create_dataset_step,
        train_step,
        create_model_step,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

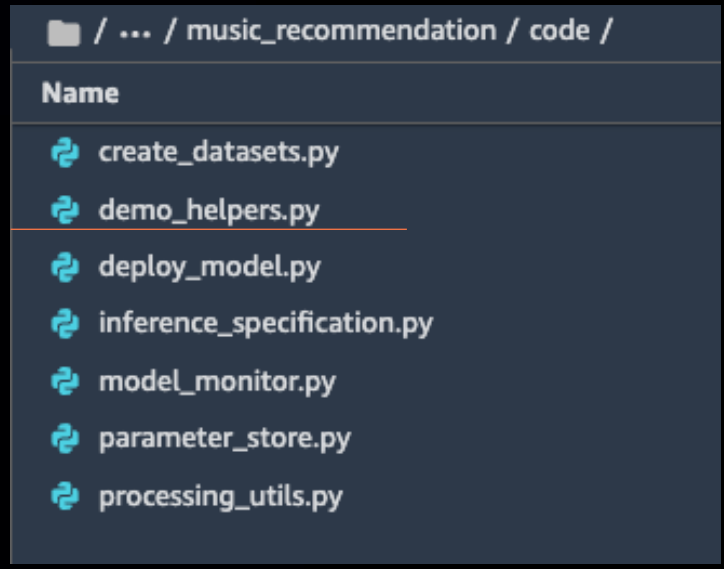




© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Music recommendation example: Cleanup






```
# when demo_helpers.delete_project_resources() is ran it will delete all the resources created by this
sys.path.insert(1, './code')
import demo_helpers  # our custom set of functions


def remove_all_resources():
    demo_helpers.delete_project_resources(
        sagemaker_boto_client=sagemaker_boto_client,
        sagemaker_session=sagemaker_session,
        endpoint_names=[pipeline_endpoint_name, endpoint_name],
        pipeline_names=[pipeline_name, dataprep_pipeline_name, train_deploy_pipeline_name],
        mpg_name=mpg_name,
        feature_groups=[fg_name_ratings, fg_name_tracks, fg_name_user_preferences],
        prefix=prefix,
        delete_s3_objects=True,
        bucket_name=bucket
    )
```

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

Resources

AWS Well-Architected Framework
https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html

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

Architect and build the full machine learning lifecycle with amazon SageMaker (blog)
https://aws.amazon.com/blogs/machine-learning/architect-and-build-the-full-machine-learning-lifecycle-with-amazon-sagemaker/

AWS Sagemaker music recommendation example
https://github.com/aws/amazon-sagemaker-examples/tree/master/end_to_end/music_recommendation

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

Thank you!

ML Lens Whitepaper

Haleh Najafzadeh

Principal AI/ML Solutions Architect Well-Architected
AWS
hnajafza@amazon.com

Ali Arsanjani

Principal AI/ML Specialist Solutions Architect
AWS
arsanjani@amazon.com

aws

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