aws SUMMIT

ASEAN

COM203

Using IaC with Terraform to provision big data platform on Amazon EMR

Dwi Fahni Denni

Lead DevOps Engineer, ZebraX
AWS Community Builder

Agenda

- Architecture Big Data Platform
- GitOps Flow
- Demo (Deploy Amazon EMR Cluster)

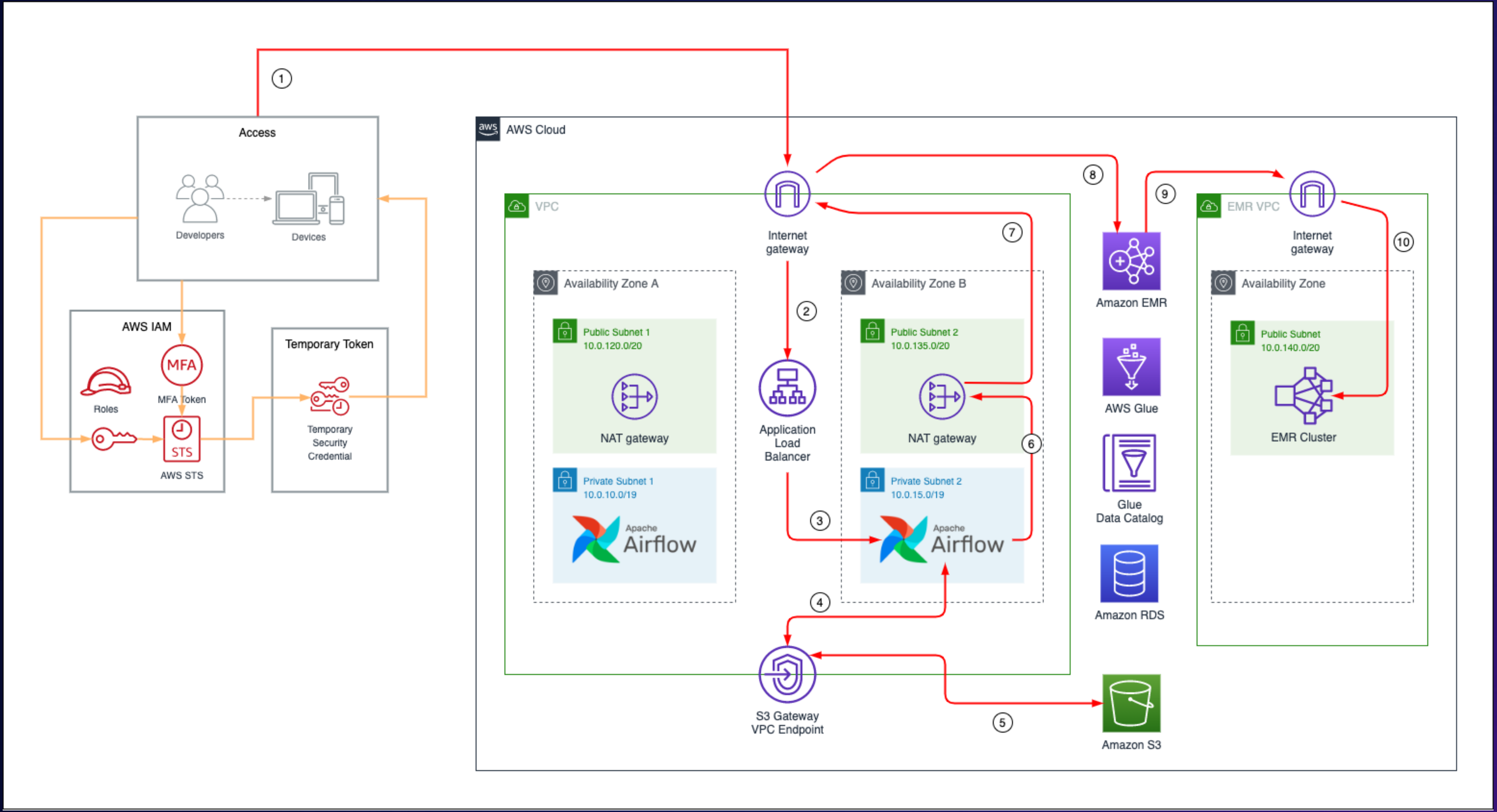
Abstract

USING IAC WITH TERRAFORM TO PROVISION BIG DATA PLATFORM ON AMAZON EMR

- Training multiple Machine Learning (ML) Models in a serial mode can be challenging, time-consuming, and ineffective. Parallel modelling provides tremendous benefits in building a variety of models by speeding up the process through parallelisation so the model building process becomes more efficient.
- Solve the problem of training multiple ML models by using Spark Panda's UDF (a Python library for building ML models) inside Amazon EMR cluster
- Provision Amazon EMR cluster using Terraform as an Infrastructure-as-Code tool

Architecture Big Data Platform

Architecture Big Data Platform



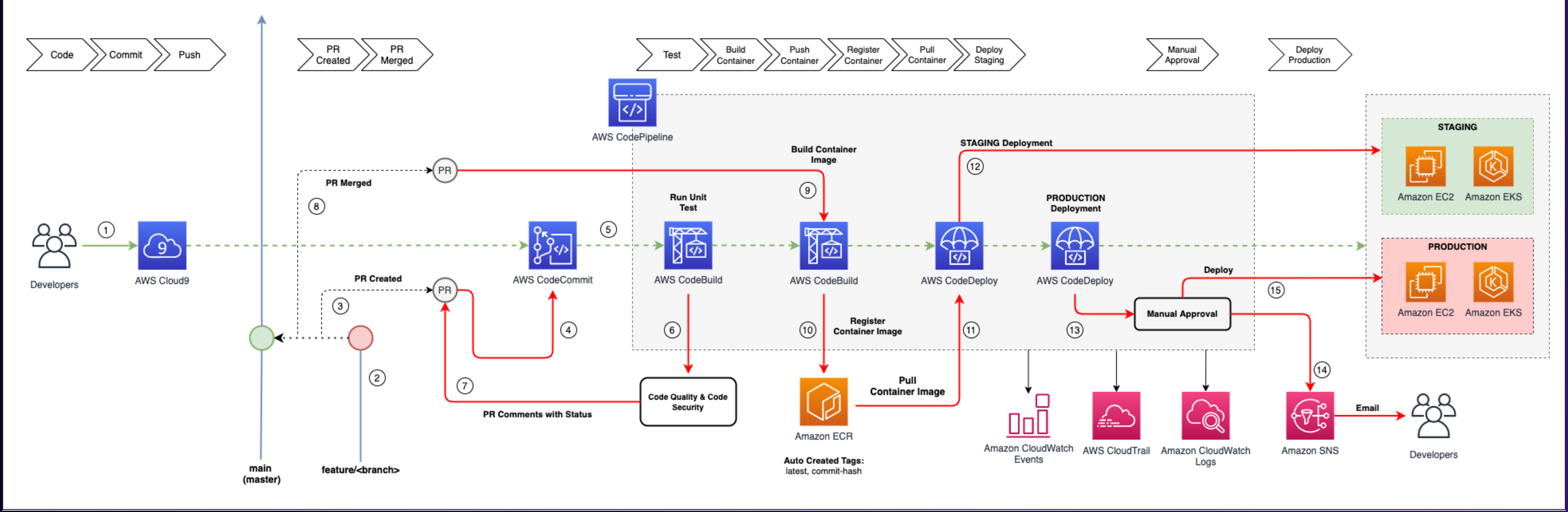
GitOps Pipeline

GitOps Pipeline

- GitOps Flow
- CI/CD Pipeline Deployment

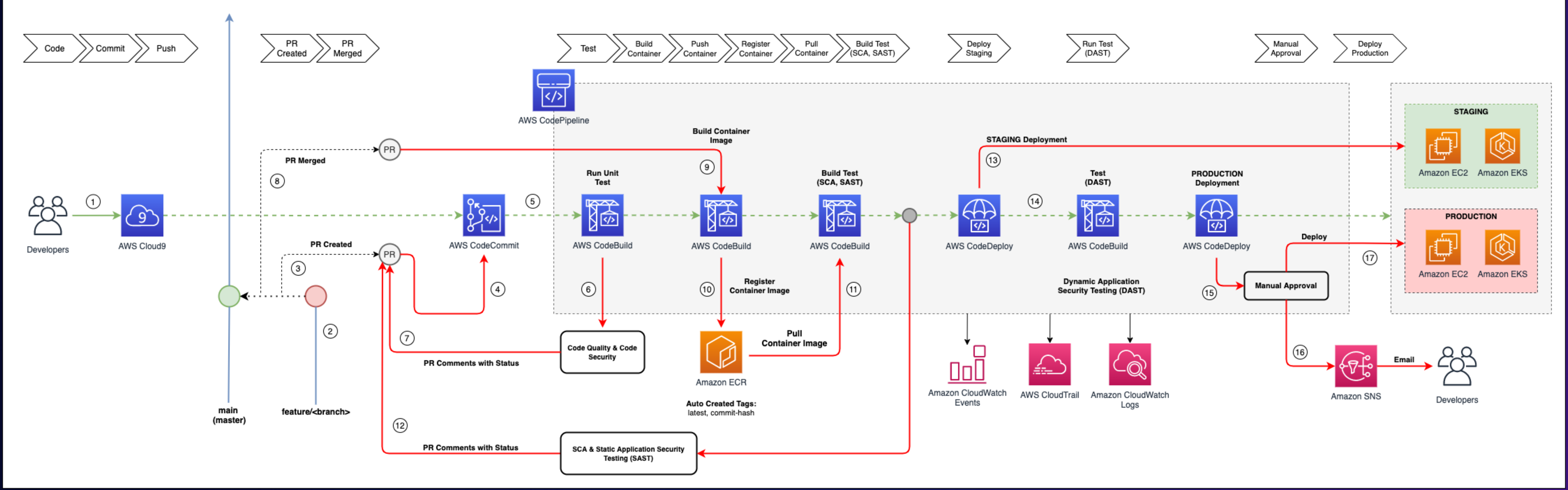
GitOps Pipeline

GITOPS FLOW



GitOps Pipeline

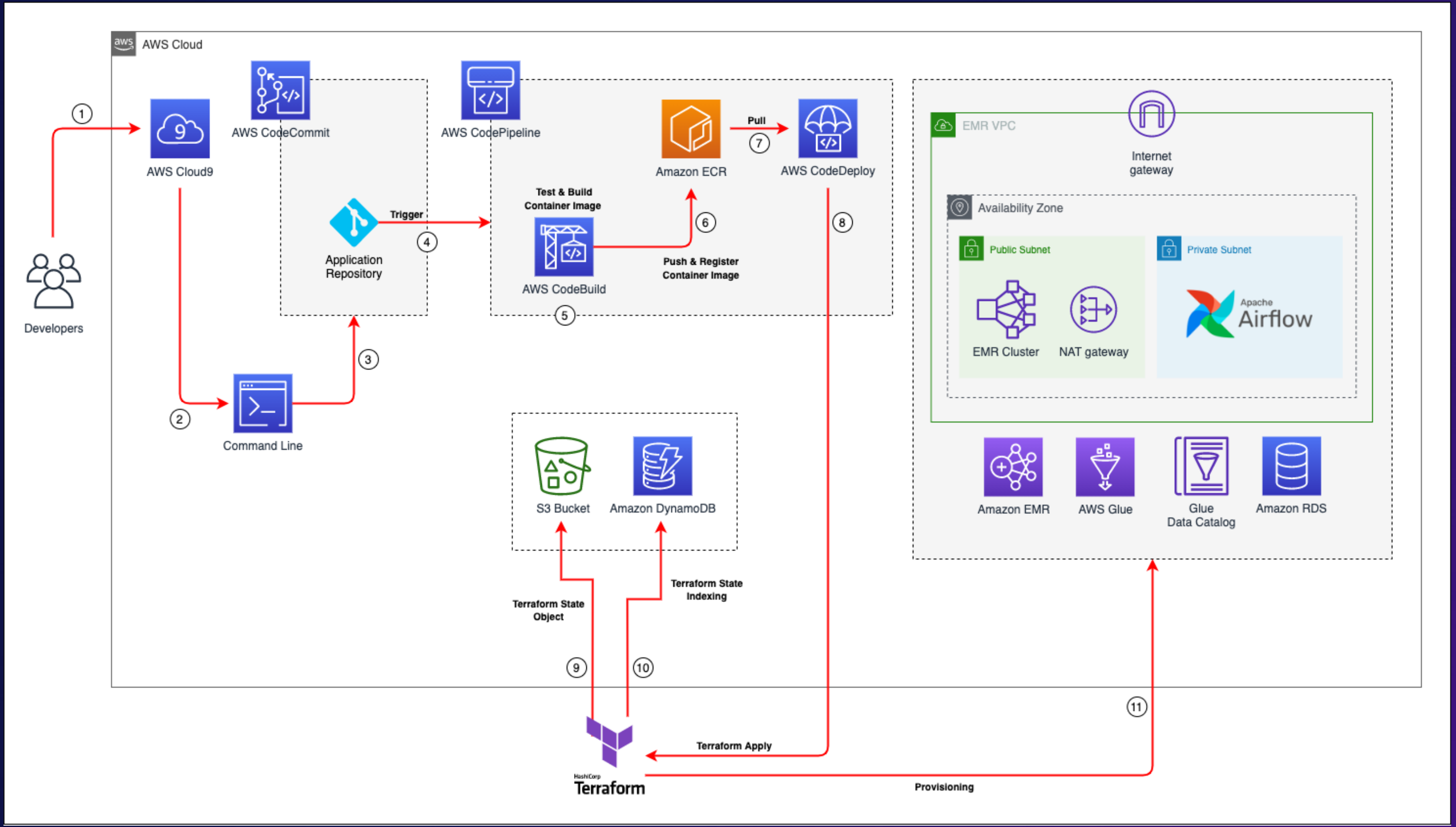
GITOPS DEVSECOPS FLOW



GitOps Pipeline

CI/CD PIPELINE DEPLOYMENT

Workflow of CI/CD pipeline in provisioning Amazon EMR Cluster Terraform



Demo

Provisioning Amazon EMR

- Preparations
- Deploy Amazon EMR
- Bootstrap Script & Terraform Stack
- Running Spark Job

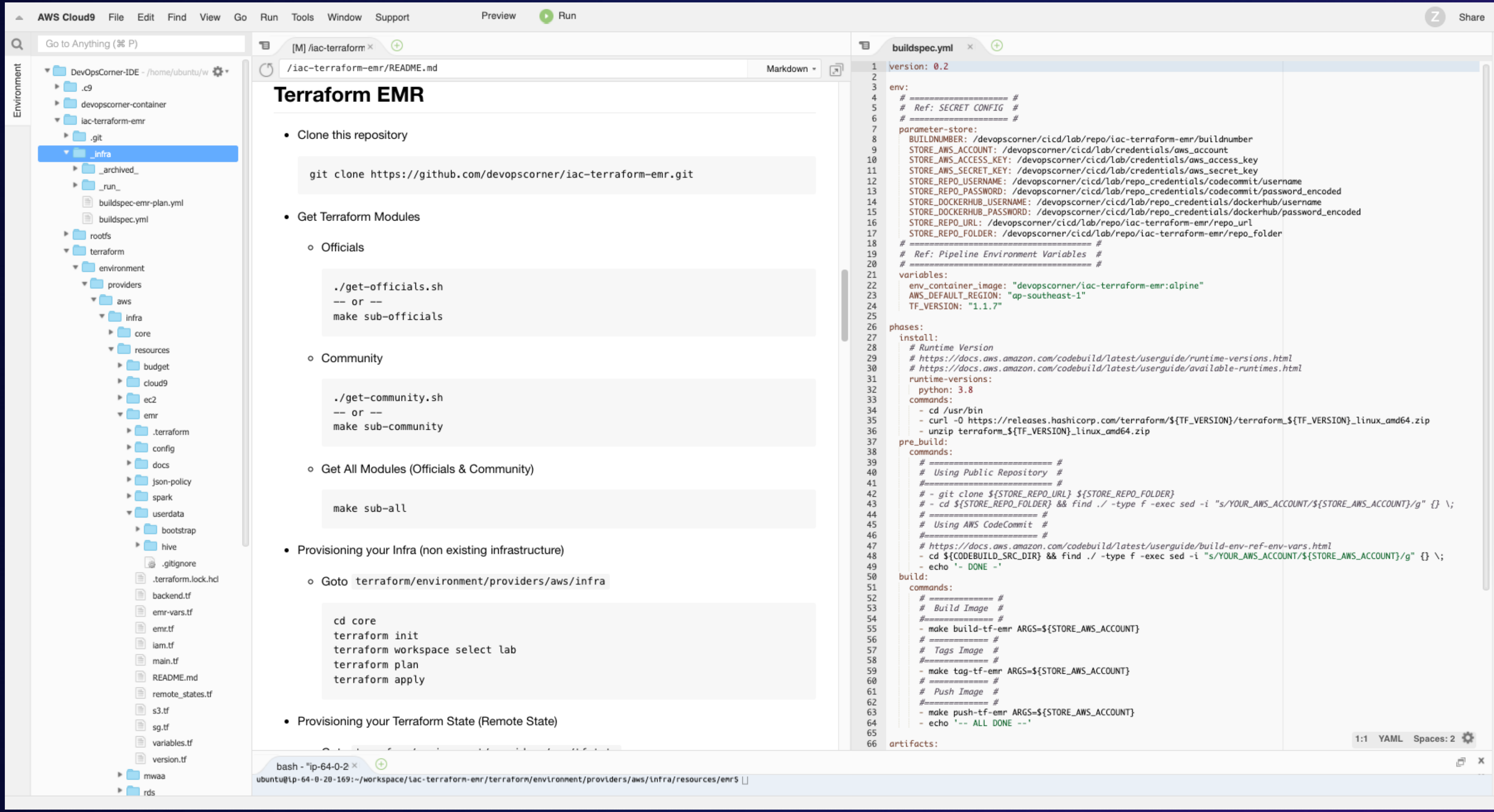
Provisioning Amazon EMR

PREPARATIONS

- IAM Role, Policy (AssumeRole)
- Infrastructure
  - VPC, Subnet, NAT, Internet-Gateway, DNS
- S3 Bucket
  - Bucket for Terraform State
  - Bucket for EMR Bootstrap & EMR Configuration
- DynamoDB
  - Indexing Terraform State
- Database (RDS) – *optional*

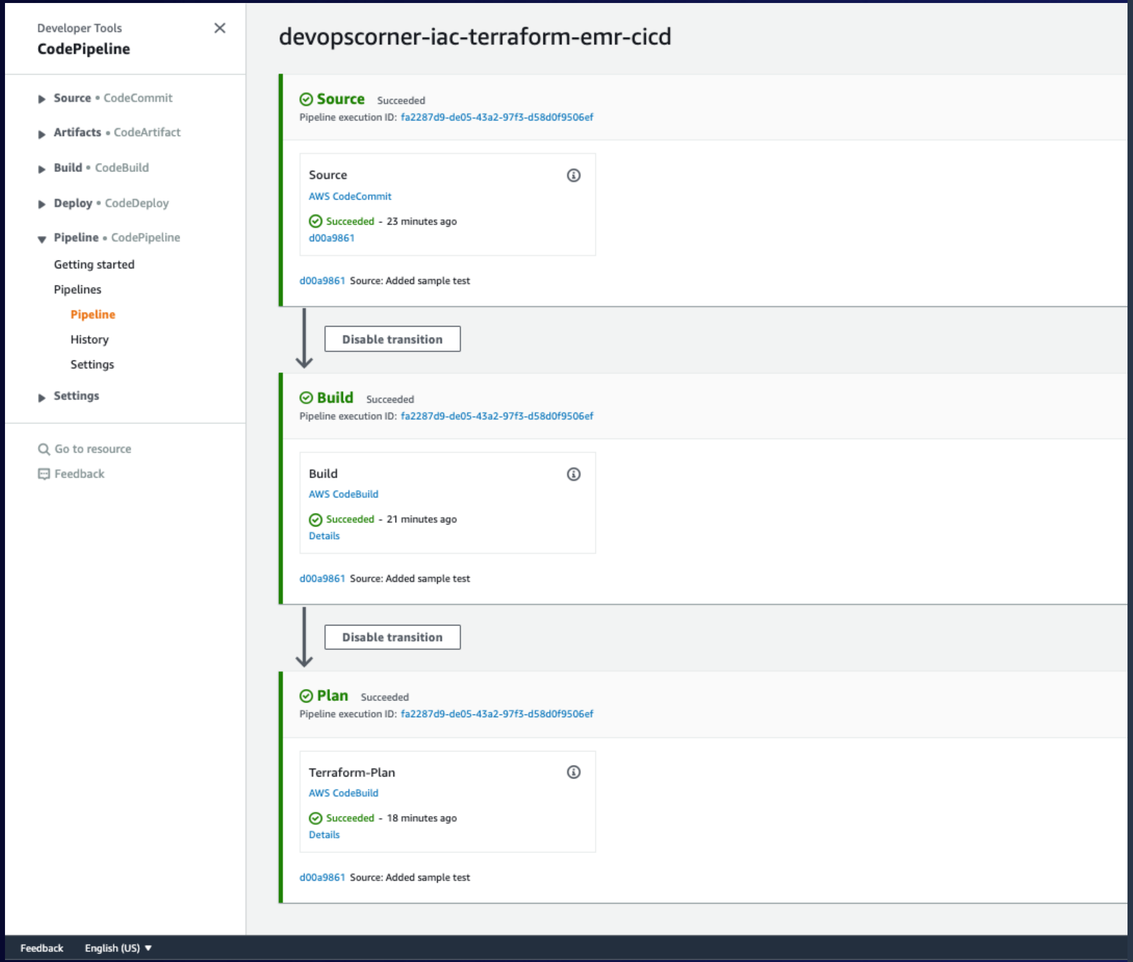
Provisioning Amazon EMR

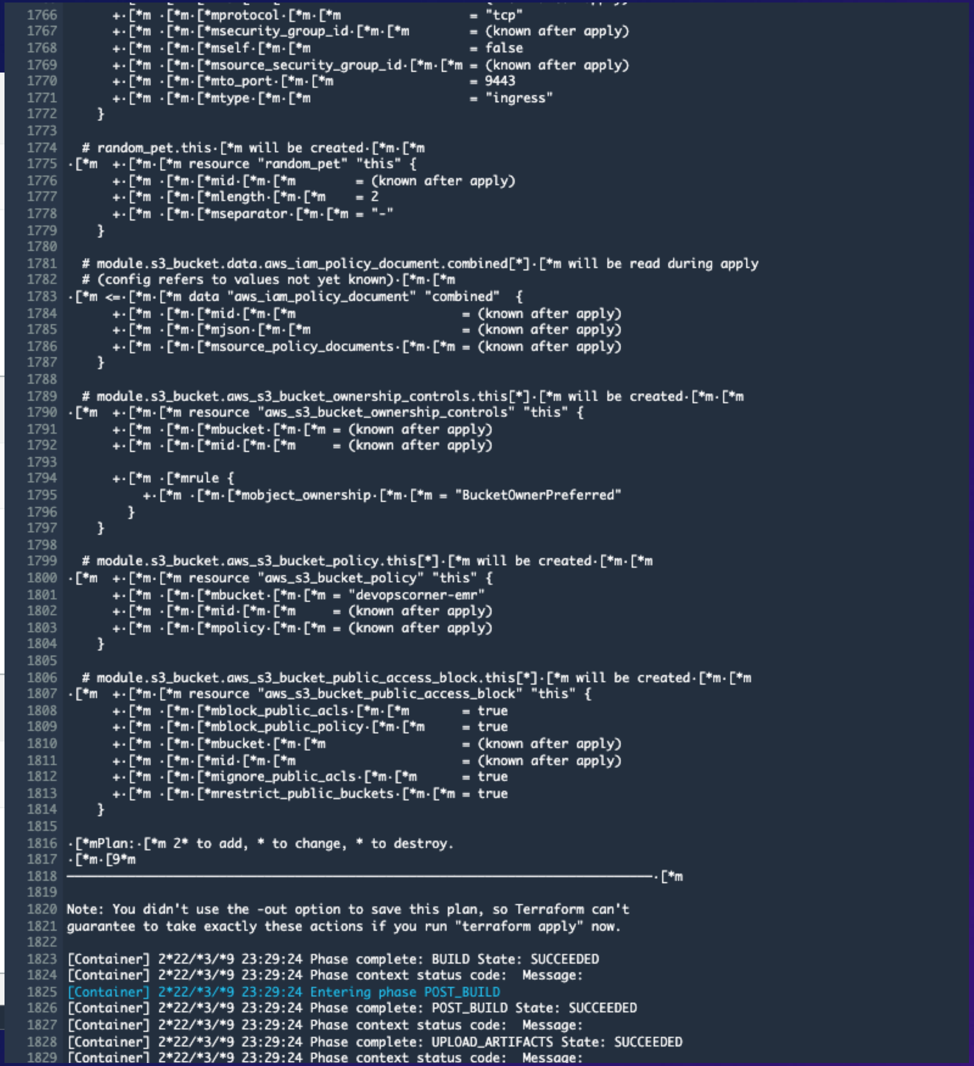
DEPLOY AMAZON EMR – BUILDSPEC (AWS CODEBUILD)



Provisioning Amazon EMR

DEPLOY AMAZON EMR – AWS DEVELOPER TOOLS





Deploy Amazon EMR

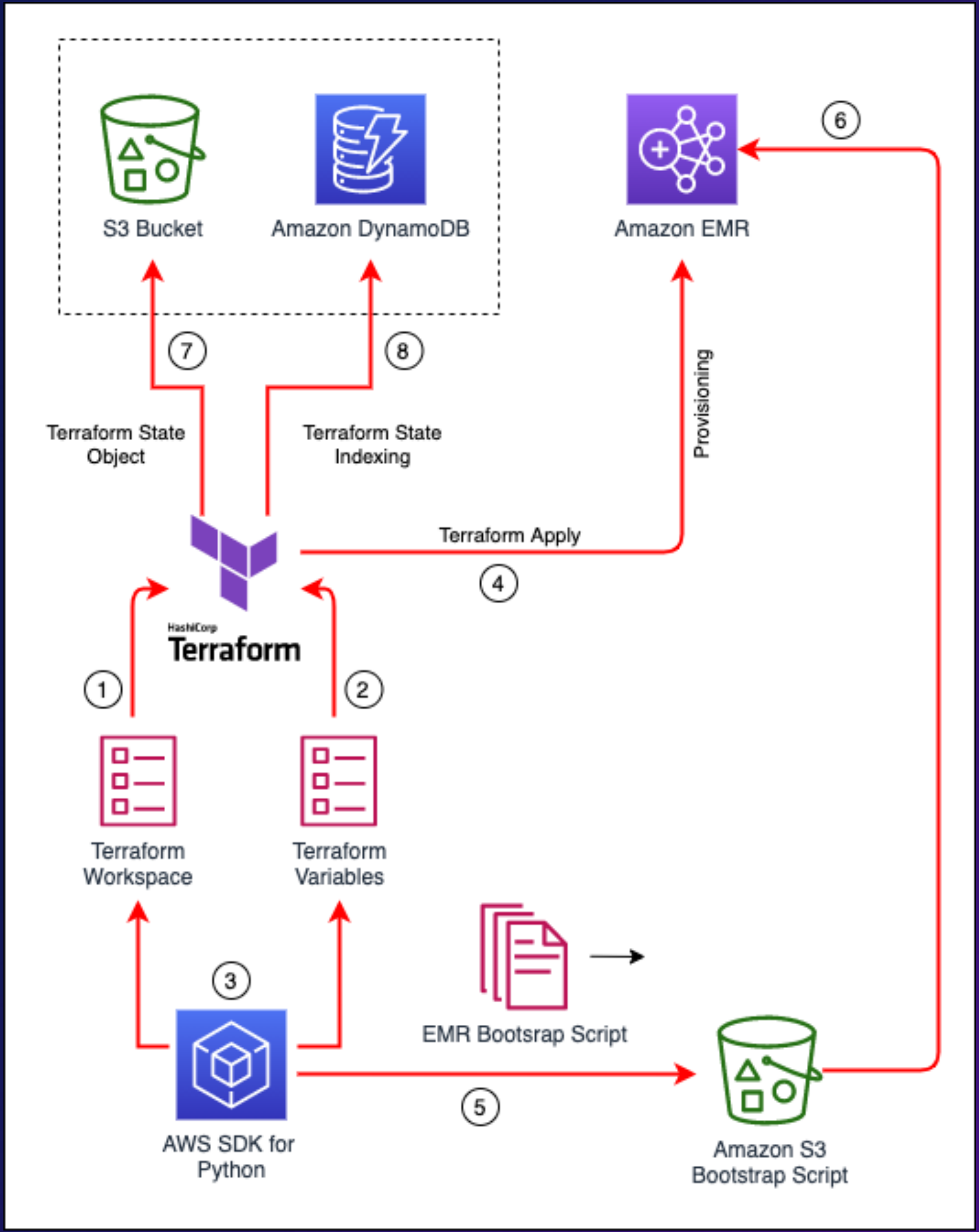
(ETA: 30 minutes)

Provisioning Amazon EMR

BOOTSTRAP SCRIPT & TERRAFORM STACK

Bootstrap script including:

- IAM (Role & Policy)
- Provisioning Master, Core & Task Instance Fleet
- Instance Type Provisioning
- Volume (EBS) Size Instance
- Hadoop Debugging
- Autoscaling
- Monitoring & Log



Run Spark Jobs

Provisioning Amazon EMR

RUNNING SPARK JOB – SERIAL MODEL

serial_model.py 9+

Users > dwidenni > Desktop > serial_model.py > train_model

```python
def train_model(train_ds):
    idx = str(train_ds[partition_column].unique()[0]).lower()
    X = train_ds[feature_column]
    y = train_ds[target_column]
    X_train, X_test, y_train, y_test = train_test_split(
        X,
        y,
        train_size=train_ratio,
        random_state=42
    )

    base_model.fit(X_train, y_train)

    y_pred_test = base_model.predict(X_test)

    accuracy_test = metrics.accuracy_score(y_test, y_pred_test).tolist()
    f1_test = metrics.f1_score(y_test, y_pred_test, average='weighted').
    tolist()

    precision_test = metrics.precision_score(y_test, y_pred_test,
                                average='weighted').tolist()

    recall_test = metrics.recall_score(y_test, y_pred_test,
                              average='weighted').tolist()

    model_results_test = pd.DataFrame(
        [[idx, accuracy_test, f1_test, precision_test, recall_test]],
        columns=["id", "accuracy", "weighted_f1", "weightedPrecision",
        "weightedRecall"])

    filename = 'finalized_model_{}.sav'.format(idx)
    joblib.dump(base_model, filename)

    return model_results_test

# load the data into pandas dataframe
df = pd.read_csv('activity_analytics.csv')

# set all necesarry information for the training process
partition_column = ["asset_type"]
target_column = ['status']
feature_column = ["avg_","max_","min_","stddev_","rng_"]
train_ratio = 0.7
partition_list = list(df[partition_column].unique())

```

serial_model.py 9+

Users > dwidenni > Desktop > serial_model.py > ...

```python
# set all necesarry information for the training process
partition_column = ["asset_type"]
target_column = ['status']
feature_column = ["avg_","max_","min_","stddev_","rng_"]
train_ratio = 0.7
partition_list = list(df[partition_column].unique())

base_model = tree.DecisionTreeClassifier()

# call the train_model function inside the for loop
# the training is done in serial fashion
training_result = pd.DataFrame()
for partition in partition_list:
    sub_df = df.loc[df[partition_column]==partition]
    model_training = train_model(sub_df)
    training_result = pd.concat([training_result, model_training])

```

Provisioning Amazon EMR

RUNNING SPARK JOB – PARALLEL MODEL

paralel_model.py

Users > dwidenni > Desktop > paralel_model.py > train_model > f1_test

```python
base_model = tree.DecisionTreeClassifier()
schema = StructType([
    StructField("id", StringType()),
    StructField("accuracy", FloatType()),
    StructField("weighted_f1", FloatType()),
    StructField("weightedPrecision", FloatType()),
    StructField("weightedRecall", FloatType())
    ])

# Wrap the python udf using @pandas_udf
# this wrapped function will be used for parallel training
@pandas_udf(schema, PandasUDFType.GROUPED_MAP)
def train_model(train_ds):
    idx = str(train_ds[partition_column].unique()[0]).lower()
    X = train_ds[feature_column]
    y = train_ds[target_column]
    X_train, X_test, y_train, y_test = train_test_split(
        X,
        y,
        train_size=train_ratio,
        random_state=42
    )

    base_model.fit(X_train, y_train)
    y_pred_test = base_model.predict(X_test)
    accuracy_test = metrics.accuracy_score(y_test, y_pred_test).tolist()

    f1_test = metrics.f1_score(y_test, y_pred_test, average='weighted').
    tolist()

    precision_test = metrics.precision_score(y_test, y_pred_test,
                                    average='weighted').tolist()

    recall_test = metrics.recall_score(y_test, y_pred_test,
                                average='weighted').tolist()

    model_results_test = pd.DataFrame(
        [[idx, accuracy_test, f1_test, precision_test, recall_test]],
        columns=["id", "accuracy", "weighted_f1", "weightedPrecision",
        "weightedRecall"])

    filename = 'finalized_model_{}.sav'.format(idx)
    joblib.dump(base_model, filename)
    return model_results_test
```

paralel_model.py

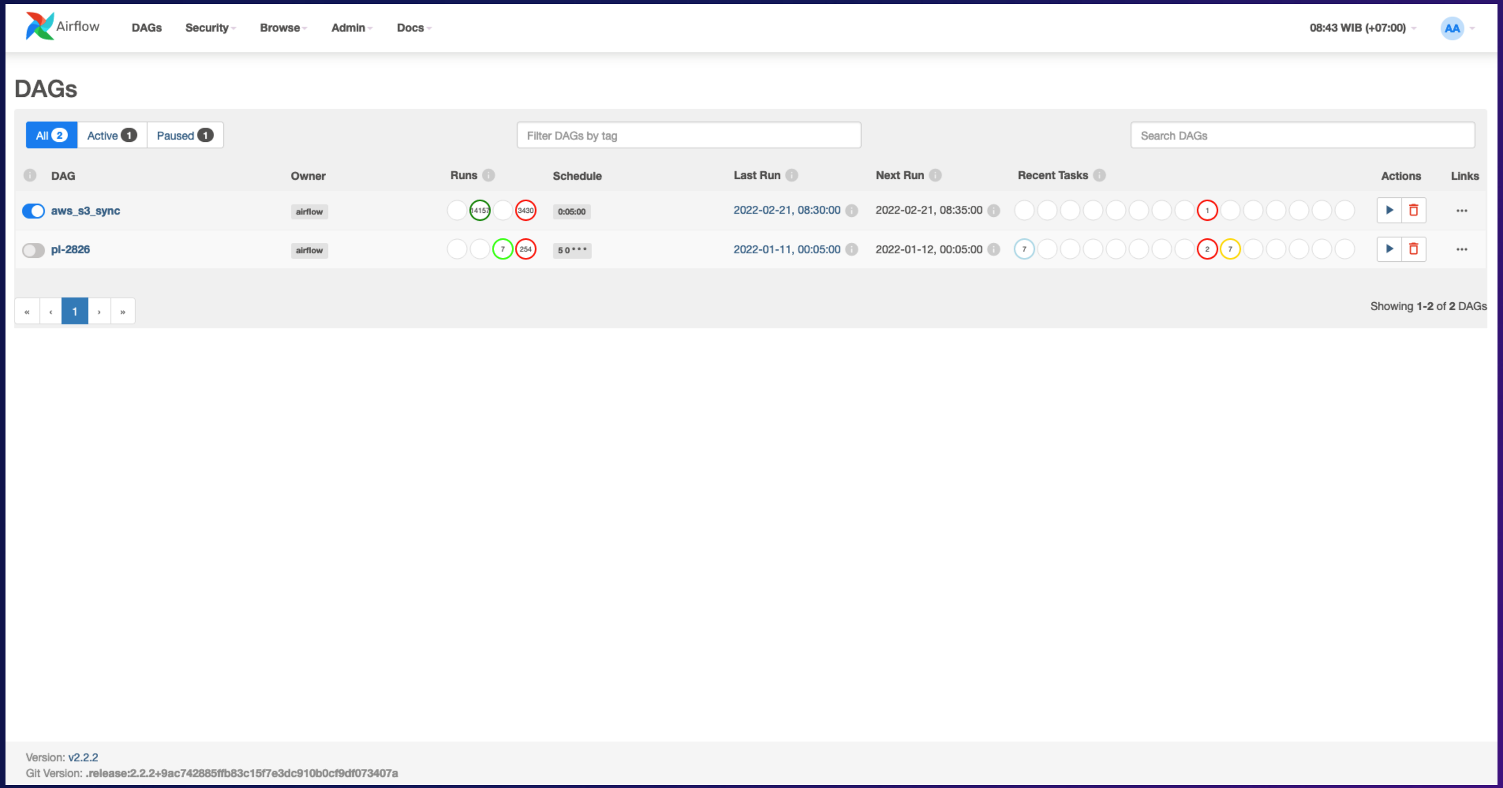
Users > dwidenni > Desktop > paralel_model.py > ...

```python

# Create spark session and load the data into a spark Dataframe
spark = SparkSession.builder \
    .appName("parallel_model_training") \
    .getOrCreate()

df = spark.read.options(header='True', inferSchema='True')\
    .csv('activity_analytics.csv')

# Set all necessary information for the training process
partition_column = ["asset_type"]
target_column = ['status']
feature_column = ["avg_","max_","min_","stddev_","rng_"]
train_ratio = 0.7

# Call our pandas_udf function using groupby-apply method
# The training proces will be done in parallel fashion instead of serial
partition_result = df.groupBy(partition_column).apply(
    train_model
    ).toPandas()
```

Provisioning Amazon EMR

RUNNING SPARK JOB – AIRFLOW MONITORING WORKFLOW

- Airflow is a useful tool to monitor workflow as Directed Acyclic Graphs (DAGs) of tasks
- The scheduler for automation pipeline and monitoring Spark job (python script) will be running under Airflow



References

References

Resources	Links
Docker Container CI/CD	https://github.com/devopscorner/devopscorner-container
User Data Installer Scripts	https://github.com/devopscorner/scripts
IaC Terraform EMR	https://github.com/devopscorner/iac-terraform-emr
Big Data and Machine Learning	https://devopscorner.id/category/machine-learning/
How to Efficiently Train Multiple ML Models on a Spark Cluster	https://medium.com/zebrax/how-to-efficiently-train-multiple-ml-models-on-a-spark-cluster-7d84512d36f0

Thank you!

Dwi Fahni Denni

Lead DevOps Engineer, ZebraX
AWS Community Builder

https://www.linkedin.com/in/dfdenni

Learn in-demand AWS Cloud skills

AWS Skill Builder

Access **500+ free** digital courses and Learning Plans

Explore resources with a variety of skill levels and **16+** languages to meet your learning needs

Deepen your skills with digital learning on demand

Train now

AWS Certifications

Earn an industry-recognized credential

Receive Foundational, Associate, Professional, and Specialty certifications

Join the **AWS Certified community** and get exclusive benefits

Access **new** exam guides

Thank you for attending **AWS Summit Online ASEAN 2022**.

Please **complete the session survey** to help us improve your Summit experience in the future.

aws-asean-marketing@amazon.com

twitter.com/AWSCloudSEAsia

linkedin.com/company/amazon-web-services

facebook.com/AmazonWebServices

instagram.com/amazonwebservices

youtube.com/user/AmazonWebServices

twitch.tv/aws