



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



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

Agenda

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

Abstract

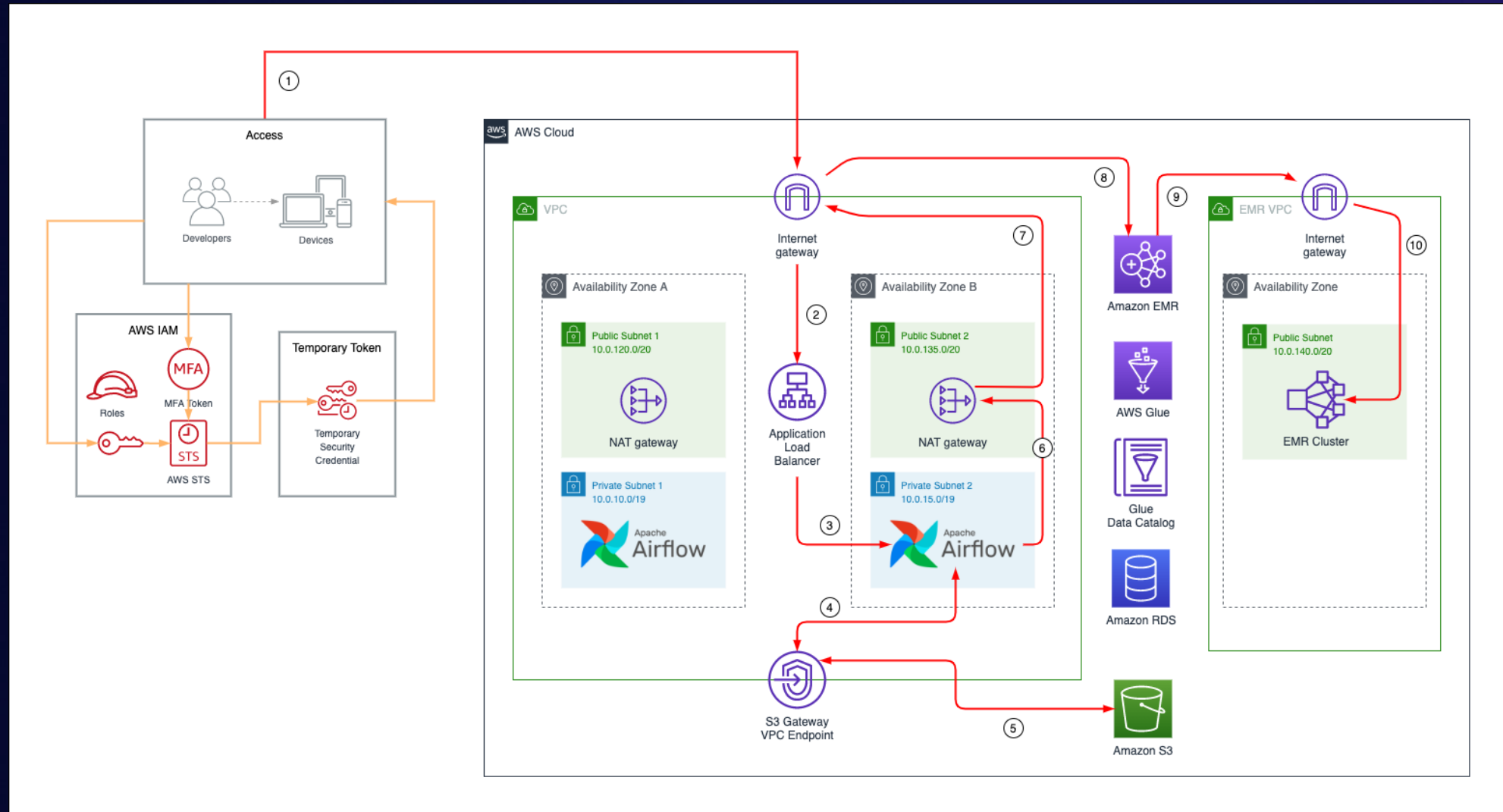
USING IAC WITH TERRAFORM TO PROVISION BIG DATA PLATFORM ON AMAZON EMR

- 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

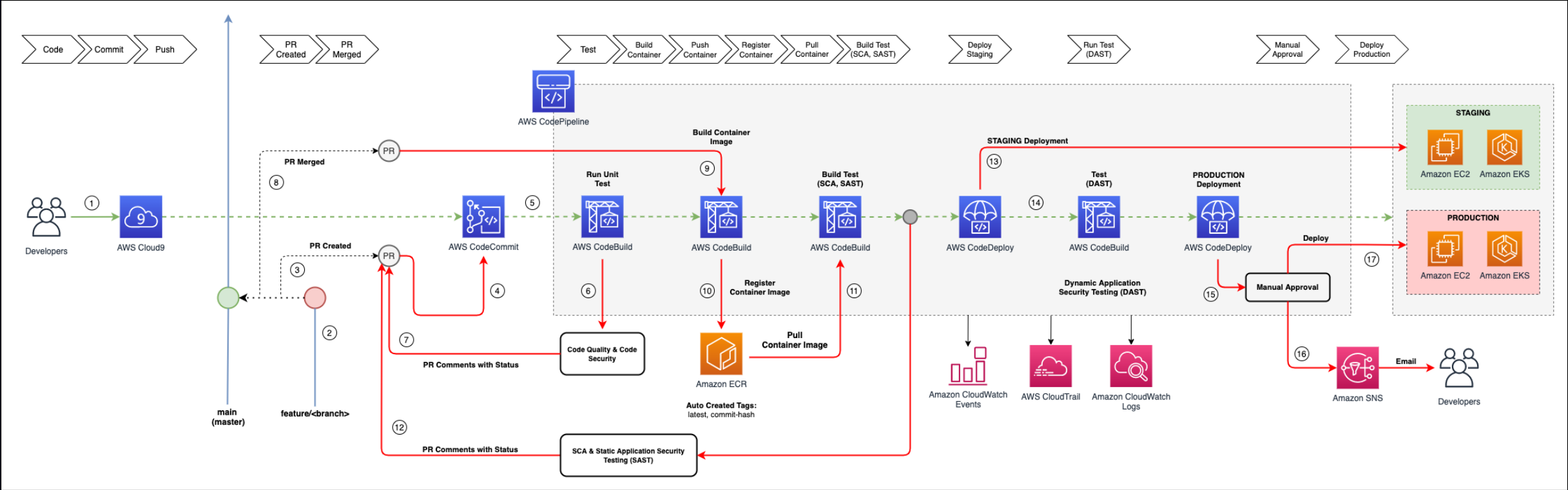
- GitOps Flow
- CI/CD Pipeline Deployment

GITOPS FLOW



GitOps Pipeline

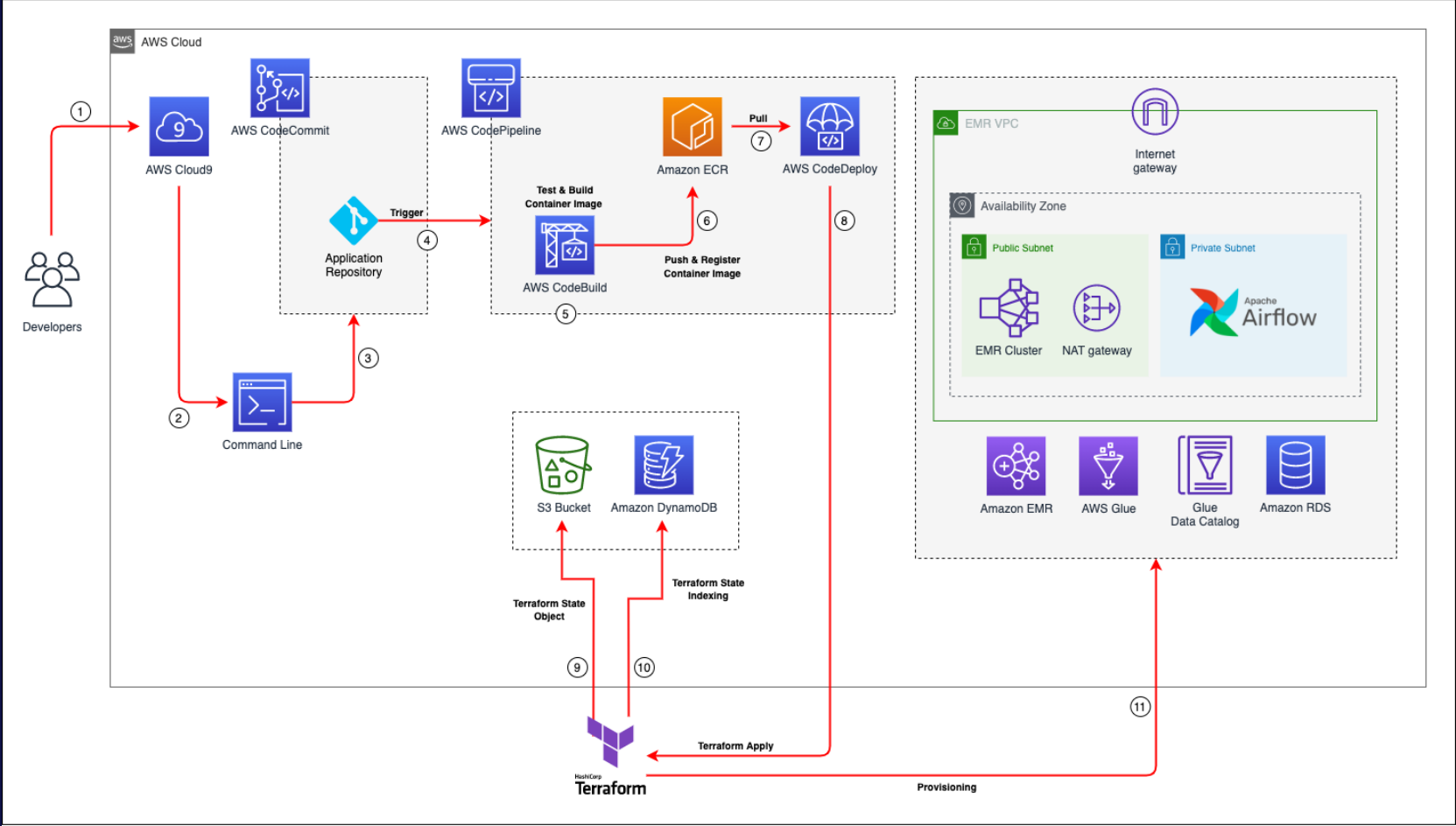
GITOPS DEVSECOPS FLOW



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)

The screenshot displays the AWS Cloud9 IDE interface, which is used for developing and running code in the cloud. The interface is divided into several panes:

- Environment Explorer (Left):** Shows the project structure. The 'infra' directory is selected, which contains files like 'buildspec-emr-plan.yml' and 'buildspec.yml'.
- Editor (Center):** Displays the 'Terraform EMR' README file. It provides instructions on how to clone the repository, get Terraform modules (officials and community), and provision the infrastructure (non-existing infrastructure and Terraform State).
- Buildspec Editor (Right):** Shows the 'buildspec.yml' file, which defines the build process. It includes environment variables for AWS credentials and repository information, and phases for installing Terraform, cloning the repository, and provisioning the infrastructure.

The 'Terraform EMR' README file contains the following instructions:

- Clone this repository**

```
git clone https://github.com/devopscorner/iac-terraform-emr.git
```
- Get Terraform Modules**
 - Officials**

```
./get-officials.sh
-- or --
make sub-officials
```
 - Community**

```
./get-community.sh
-- or --
make sub-community
```
 - Get All Modules (Officials & Community)**

```
make sub-all
```
- Provisioning your Infra (non existing infrastructure)**
 - Goto terraform/environment/providers/aws/infra**

```
cd core
terraform init
terraform workspace select lab
terraform plan
terraform apply
```
- Provisioning your Terraform State (Remote State)**

The 'buildspec.yml' file defines the build process, including environment variables, phases, and commands.

Provisioning Amazon EMR

DEPLOY AMAZON EMR – AWS DEVELOPER TOOLS

Developer Tools

CodePipeline

► Source • CodeCommit

► Artifacts • CodeArtifact

► Build • CodeBuild

► Deploy • CodeDeploy

▼ Pipeline • CodePipeline

Getting started

Pipelines

Pipeline

History

Settings

► Settings

🔍 Go to resource

📄 Feedback

devopscorner-iac-terraform-emr-cicd

Source Succeeded

Pipeline execution ID: fa2287d9-de05-43a2-97f3-d58d0f9506ef

Source

AWS CodeCommit

Succeeded - 23 minutes ago

d00a9861

Source: Added sample test

Disable transition

Build Succeeded

Pipeline execution ID: fa2287d9-de05-43a2-97f3-d58d0f9506ef

Build

AWS CodeBuild

Succeeded - 21 minutes ago

Details

d00a9861 Source: Added sample test

Disable transition

Plan Succeeded

Pipeline execution ID: fa2287d9-de05-43a2-97f3-d58d0f9506ef

Terraform-Plan

AWS CodeBuild

Succeeded - 18 minutes ago

Details

d00a9861 Source: Added sample test

```
1766 +.[*m].[*m].[*mprotocol].[*m].[*m] = "tcp"
1767 +.[*m].[*m].[*msecurity_group_id].[*m].[*m] = (known after apply)
1768 +.[*m].[*m].[*mself].[*m].[*m] = false
1769 +.[*m].[*m].[*msource_security_group_id].[*m].[*m] = (known after apply)
1770 +.[*m].[*m].[*mto_port].[*m].[*m] = 9443
1771 +.[*m].[*m].[*mtype].[*m].[*m] = "ingress"
1772 }
1773
1774 # random_pet.this.[*m] will be created.[*m].[*m]
1775 .[*m] +.[*m].[*m] resource "random_pet" "this" {
1776 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1777 +.[*m].[*m].[*m].[*m].[*m] = 2
1778 +.[*m].[*m].[*mseparator].[*m].[*m] = "-"
1779 }
1780
1781 # module.s3_bucket.data.aws_iam_policy_document.combined[*].[*m] will be read during apply
1782 # (config refers to values not yet known).[*m].[*m]
1783 .[*m] <+.[*m].[*m] data "aws_iam_policy_document" "combined" {
1784 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1785 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1786 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1787 }
1788
1789 # module.s3_bucket.aws_s3_bucket_ownership_controls.this[*].[*m] will be created.[*m].[*m]
1790 .[*m] +.[*m].[*m] resource "aws_s3_bucket_ownership_controls" "this" {
1791 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1792 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1793 }
1794 +.[*m].[*m]rule {
1795 +.[*m].[*m].[*m].[*m].[*m] = "BucketOwnerPreferred"
1796 }
1797 }
1798
1799 # module.s3_bucket.aws_s3_bucket_policy.this[*].[*m] will be created.[*m].[*m]
1800 .[*m] +.[*m].[*m] resource "aws_s3_bucket_policy" "this" {
1801 +.[*m].[*m].[*m].[*m].[*m] = "devopscorner-emr"
1802 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1803 +.[*m].[*m].[*m].[*m].[*m] = (known after apply)
1804 }
1805
1806 # module.s3_bucket.aws_s3_bucket_public_access_block.this[*].[*m] will be created.[*m].[*m]
1807 .[*m] +.[*m].[*m] resource "aws_s3_bucket_public_access_block" "this" {
1808 +.[*m].[*m].[*m].[*m].[*m].[*m].[*m] = true
1809 +.[*m].[*m].[*m].[*m].[*m].[*m].[*m] = true
1810 +.[*m].[*m].[*m].[*m].[*m].[*m].[*m] = (known after apply)
1811 +.[*m].[*m].[*m].[*m].[*m].[*m].[*m] = (known after apply)
1812 +.[*m].[*m].[*m].[*m].[*m].[*m].[*m] = true
1813 +.[*m].[*m].[*m].[*m].[*m].[*m].[*m] = true
1814 }
1815
1816 .[*m]Plan:.[*m] 2* to add, * to change, * to destroy.
1817 .[*m].[*m]
1818
1819 Note: You didn't use the -out option to save this plan, so Terraform can't
1820 guarantee to take exactly these actions if you run "terraform apply" now.
1821
1822 [Container] 2*22/*3/*9 23:29:24 Phase complete: BUILD State: SUCCEEDED
1823 [Container] 2*22/*3/*9 23:29:24 Phase context status code: Message:
1824 [Container] 2*22/*3/*9 23:29:24 Entering phase POST_BUILD
1825 [Container] 2*22/*3/*9 23:29:24 Phase complete: POST_BUILD State: SUCCEEDED
1826 [Container] 2*22/*3/*9 23:29:24 Phase context status code: Message:
1827 [Container] 2*22/*3/*9 23:29:24 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
1828 [Container] 2*22/*3/*9 23:29:24 Phase context status code: Message:
1829 [Container] 2*22/*3/*9 23:29:24 Phase context status code: Message:
```



Deploy Amazon EMR

(ETA: 30 minutes)

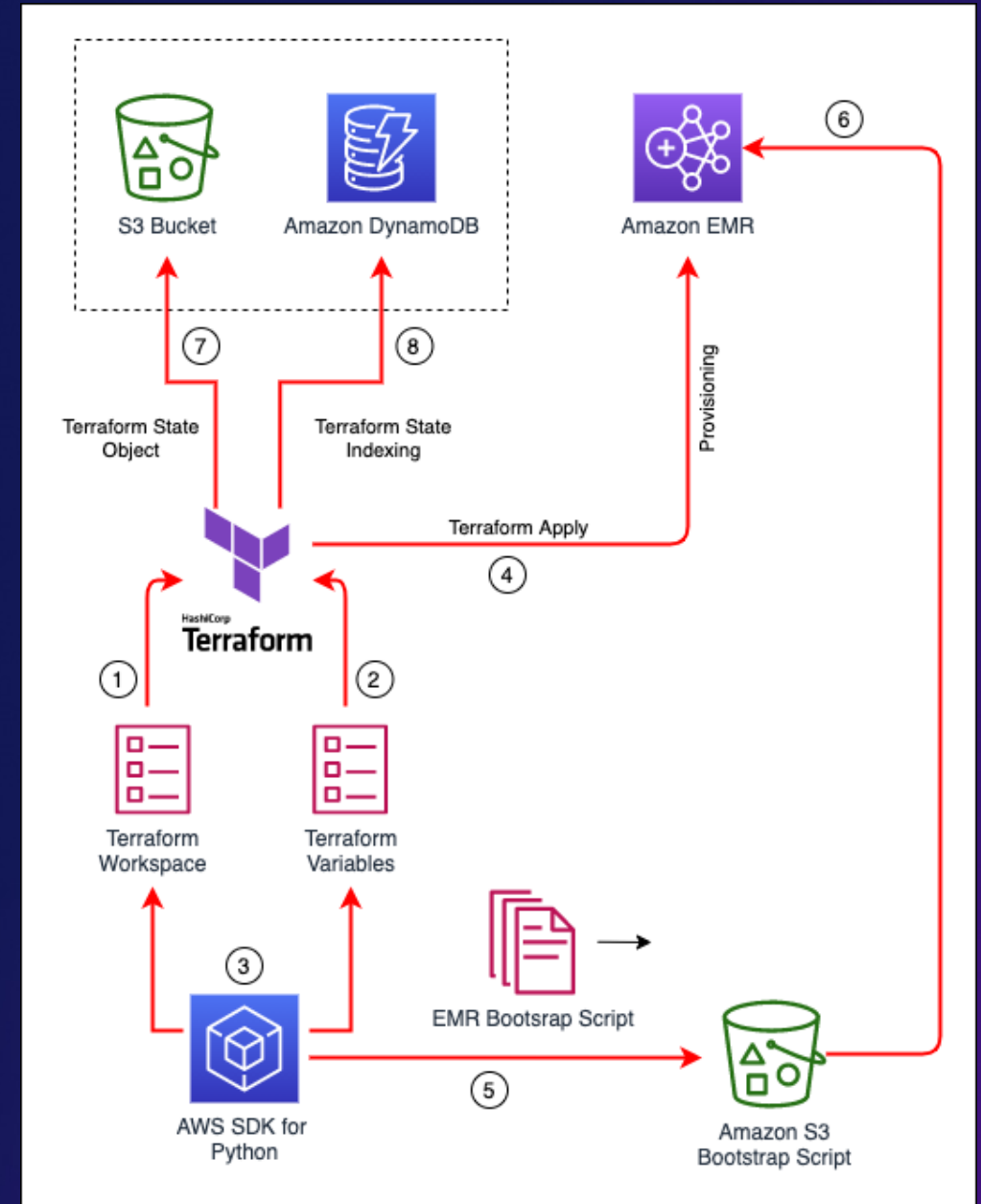


Provisioning Amazon EMR

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

Provisioning Amazon EMR

RUNNING SPARK JOB – PARALLEL MODEL

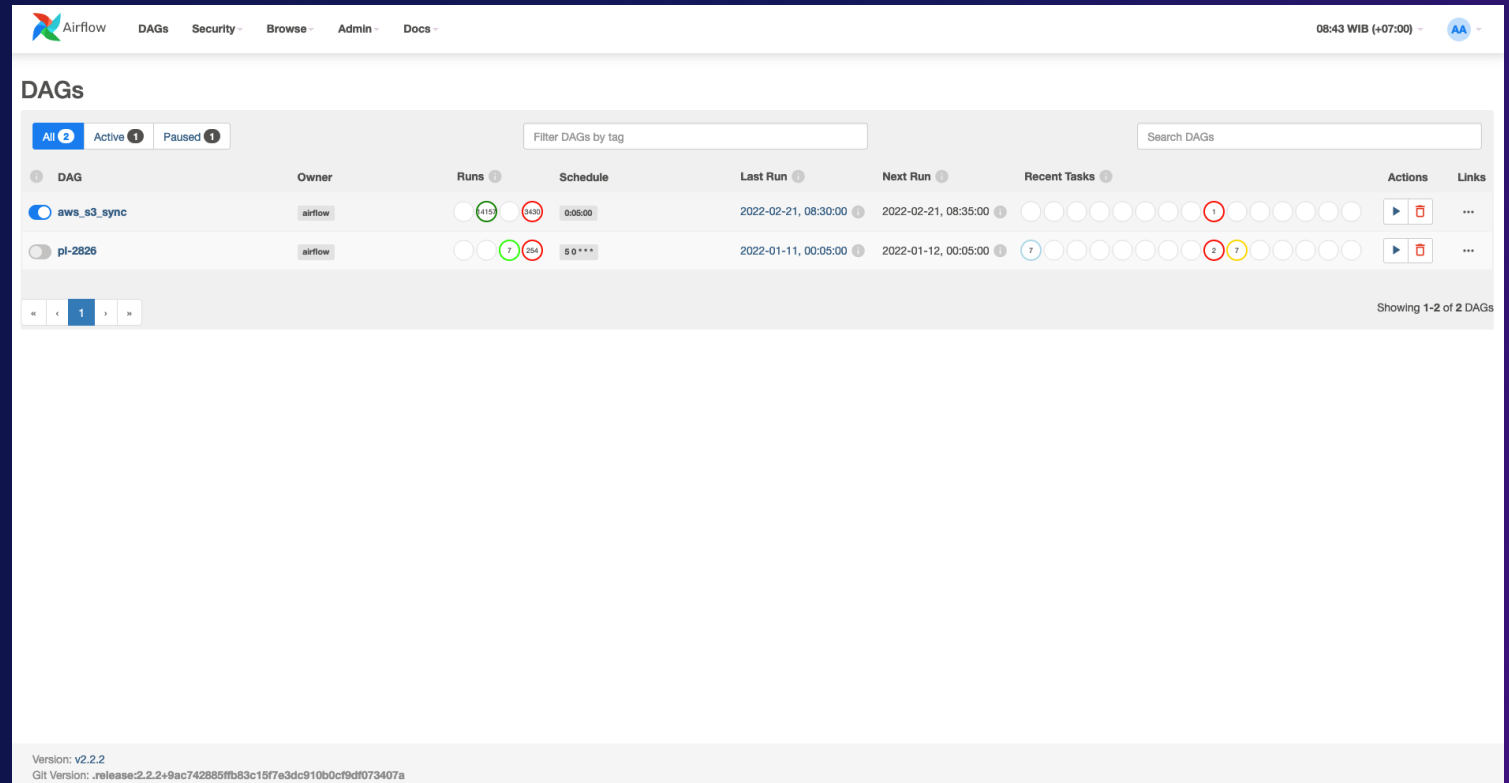
```
parallel_model.py 9+ x
Users > dwidenni > Desktop > parallel_model.py > train_model > f1_test
1 base_model = tree.DecisionTreeClassifier()
2 schema = StructType([
3     StructField("id", StringType()),
4     StructField("accuracy", FloatType()),
5     StructField("weighted_f1", FloatType()),
6     StructField("weightedPrecision", FloatType()),
7     StructField("weightedRecall", FloatType())
8 ])
9
10 # Wrap the python udf using @pandas_udf
11 # this wrapped function will be used for parallel training
12 @pandas_udf(schema, PandasUDFType.GROUPED_MAP)
13 def train_model(train_ds):
14     idx = str(train_ds[partition_column].unique()[0]).lower()
15     X = train_ds[feature_column]
16     y = train_ds[target_column]
17     X_train, X_test, y_train, y_test = train_test_split(
18         X,
19         y,
20         train_size=train_ratio,
21         random_state=42
22     )
23
24     base_model.fit(X_train, y_train)
25     y_pred_test = base_model.predict(X_test)
26     accuracy_test = metrics.accuracy_score(y_test, y_pred_test).tolist()
27
28     f1_test = metrics.f1_score(y_test, y_pred_test, average='weighted').
29     tolist()
30
31     precision_test = metrics.precision_score(y_test, y_pred_test,
32     average='weighted').tolist()
33
34     recall_test = metrics.recall_score(y_test, y_pred_test,
35     average='weighted').tolist()
36
37     model_results_test = pd.DataFrame(
38         [[idx, accuracy_test, f1_test, precision_test, recall_test]],
39         columns=["id", "accuracy", "weighted_f1", "weightedPrecision",
40         "weightedRecall"])
41
42     filename = 'finalized_model_{}.sav'.format(idx)
43     joblib.dump(base_model, filename)
44     return model_results_test
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
parallel_model.py 9+ x
Users > dwidenni > Desktop > parallel_model.py > ...
43
44 # Create spark session and load the data into a spark Dataframe
45 spark = SparkSession.builder \
46     .appName("parallel_model_training") \
47     .getOrCreate()
48
49 df = spark.read.options(header='True', inferSchema='True')\
50     .csv('activity_analytics.csv')
51
52 # Set all necessary information for the training process
53 partition_column = ["asset_type"]
54 target_column = ['status']
55 feature_column = ["avg_", "max_", "min_", "stddev_", "rng_"]
56 train_ratio = 0.7
57
58 # Call our pandas_udf function using groupby-apply method
59 # The training proces will be done in parallel fashion instead of serial
60 partition_result = df.groupby(partition_column).apply(
61     train_model
62 ).toPandas()
63
```



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



The screenshot displays the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Security, Browse, Admin, and Docs. The main header shows the time as 08:43 WIB (+07:00). Below the header, the 'DAGs' section is active, showing a list of DAGs. The interface includes filters for 'All' (2), 'Active' (1), and 'Paused' (1) DAGs. A search bar for DAGs is also present. The table lists two DAGs: 'aws_s3_sync' and 'pt-2826'. Each row shows the DAG name, owner (airflow), status (Active/Paused), schedule, last run, next run, and recent tasks. The 'aws_s3_sync' DAG is active, with a schedule of 0:05:00, last run on 2022-02-21 at 08:30:00, and next run on 2022-02-21 at 08:35:00. The 'pt-2826' DAG is paused, with a schedule of 5 0 ***, last run on 2022-01-11 at 00:05:00, and next run on 2022-01-12 at 00:05:00. The bottom of the interface shows the version (v2.2.2) and Git commit hash.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
aws_s3_sync	airflow	115 (115)	0:05:00	2022-02-21, 08:30:00	2022-02-21, 08:35:00	1	[Play] [Stop]	...
pt-2826	airflow	7 (7)	5 0 ***	2022-01-11, 00:05:00	2022-01-12, 00:05:00	2	[Play] [Stop]	...

Showing 1-2 of 2 DAGs

Version: v2.2.2
Git Version: .release:2.2.2+9ac742885ffb83c16f7e3dc910b0cf9df073407a

References



References

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

Thank you!

Dwi Fahni Denni

Lead DevOps Engineer, ZebraX
AWS Community Builder



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



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

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/amazonwebsiteservices



youtube.com/user/AmazonWebServices



twitch.tv/aws