aws

SUMMIT ONLINE

ASEAN

SPONSORED BY ZEBRA CROSS TEKNOLOGI (ZEBRAX), INDONESIA

Using IaC with Terraform to Provision Big Data Platform on Amazon EMR

Dwi Fahni Denni

Lead DevOps Engineer (ZebraX)
AWS Community Builders - Indonesia

Agenda

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

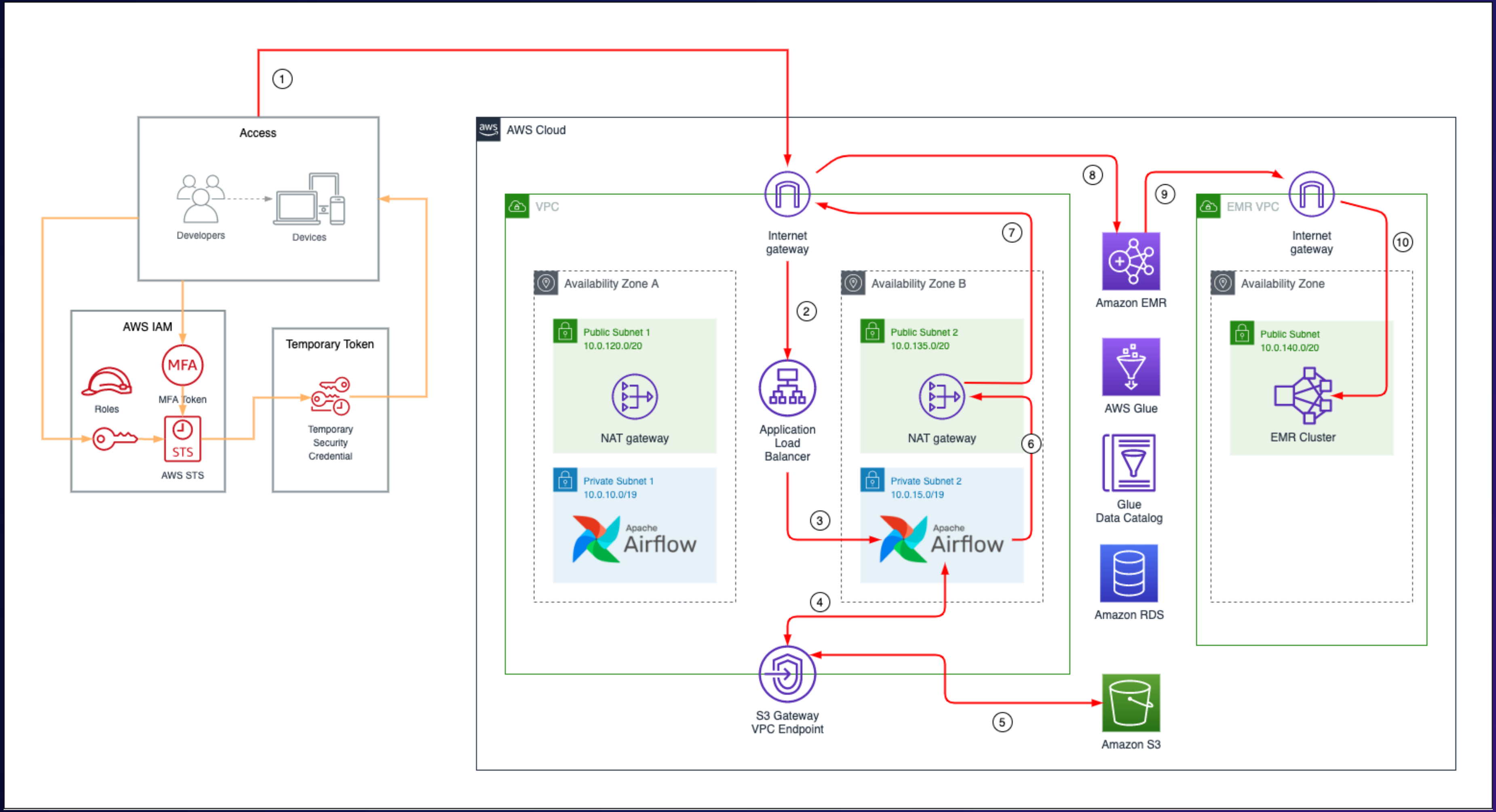
Abstract

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

- Train multiple ML Models in a serial mode it could be challenging, time consuming, and not effective. We need to create parallel modelling which provides tremendous benefit in building a variety of models by speeding up the process through parallelization so the model building process becomes more efficient.
- In this presentation, we will solving our problem in training multiple ML models using Spark Panda's UDF (a python libraries for building ML models) inside Amazon EMR cluster.
- We will also learn how to provisioning Amazon EMR cluster with Terraform as Infrastructure-as-Code tools.

Chapter 1:
Architecture Big Data Platform

Architecture Big Data Platform



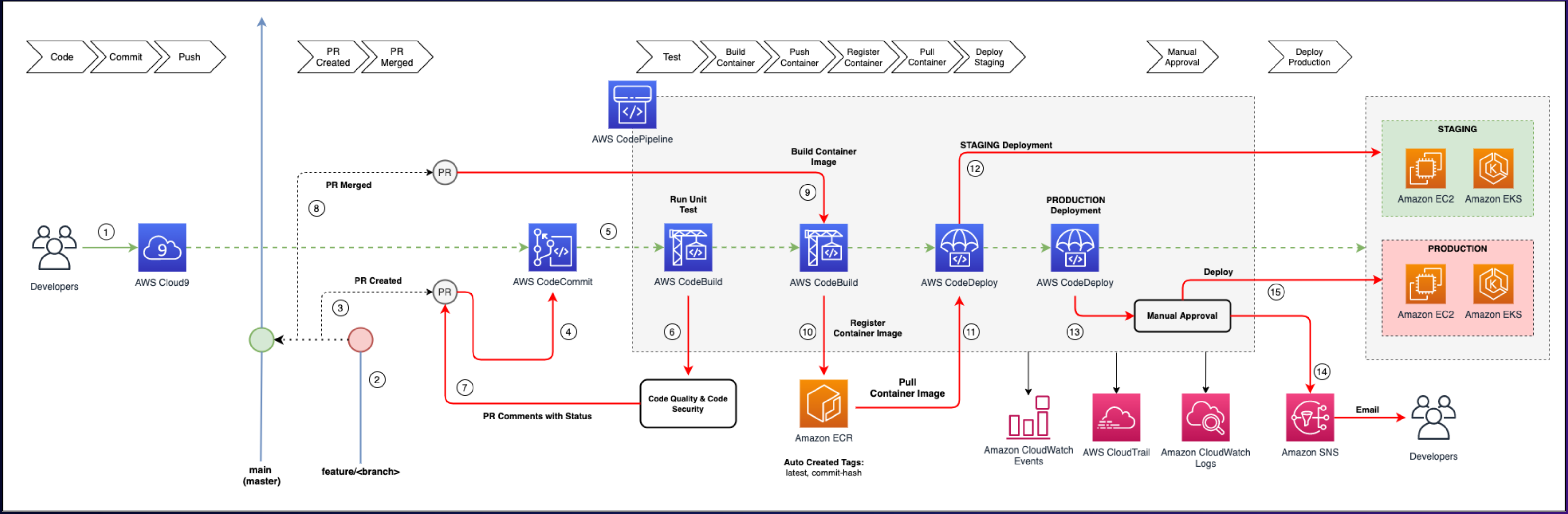
Chapter 2: GitOps Pipeline

GitOps Pipeline

- GitOps Flow
- CI/CD Pipeline Deployment

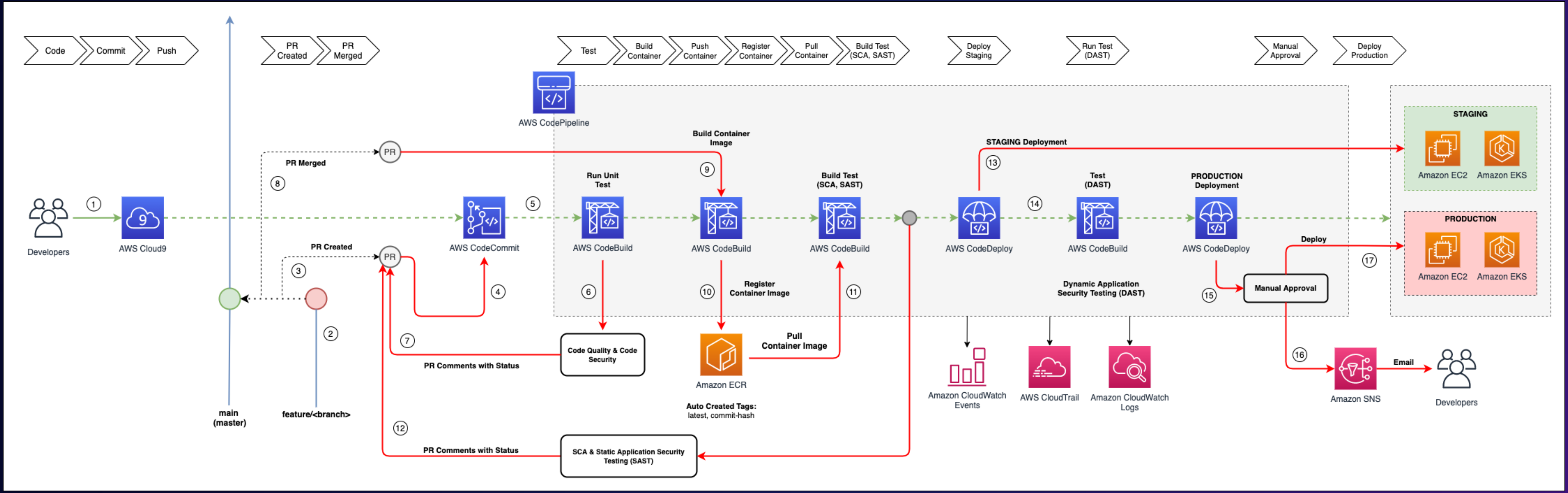
GitOps Pipeline

GITOPS FLOW



GitOps Pipeline

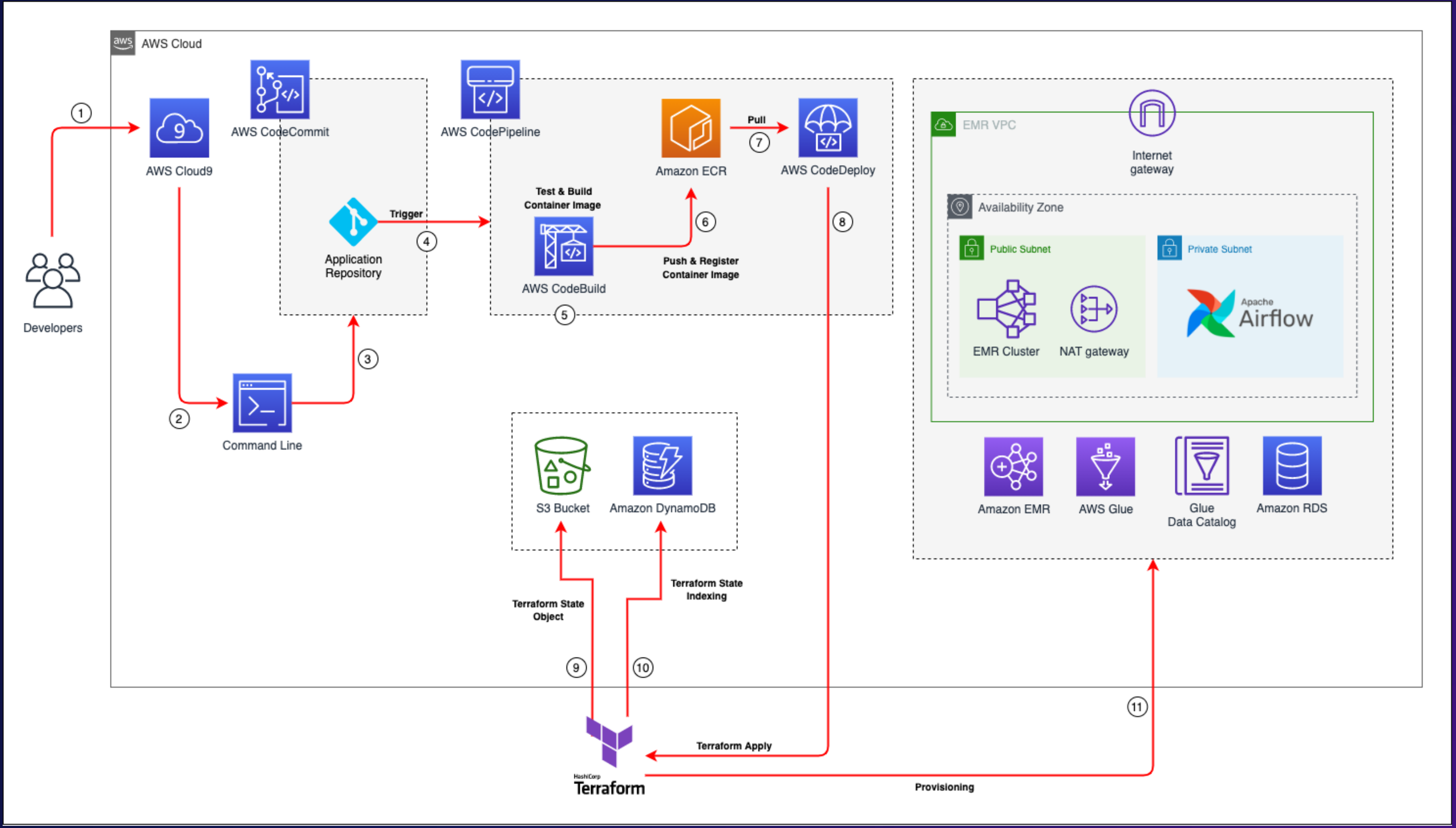
GITOPS DEVSECOPS FLOW



GitOps Pipeline

CI/CD PIPELINE DEPLOYMENT

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



Chapter 3:
Demo

Provisioning Amazon EMR

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

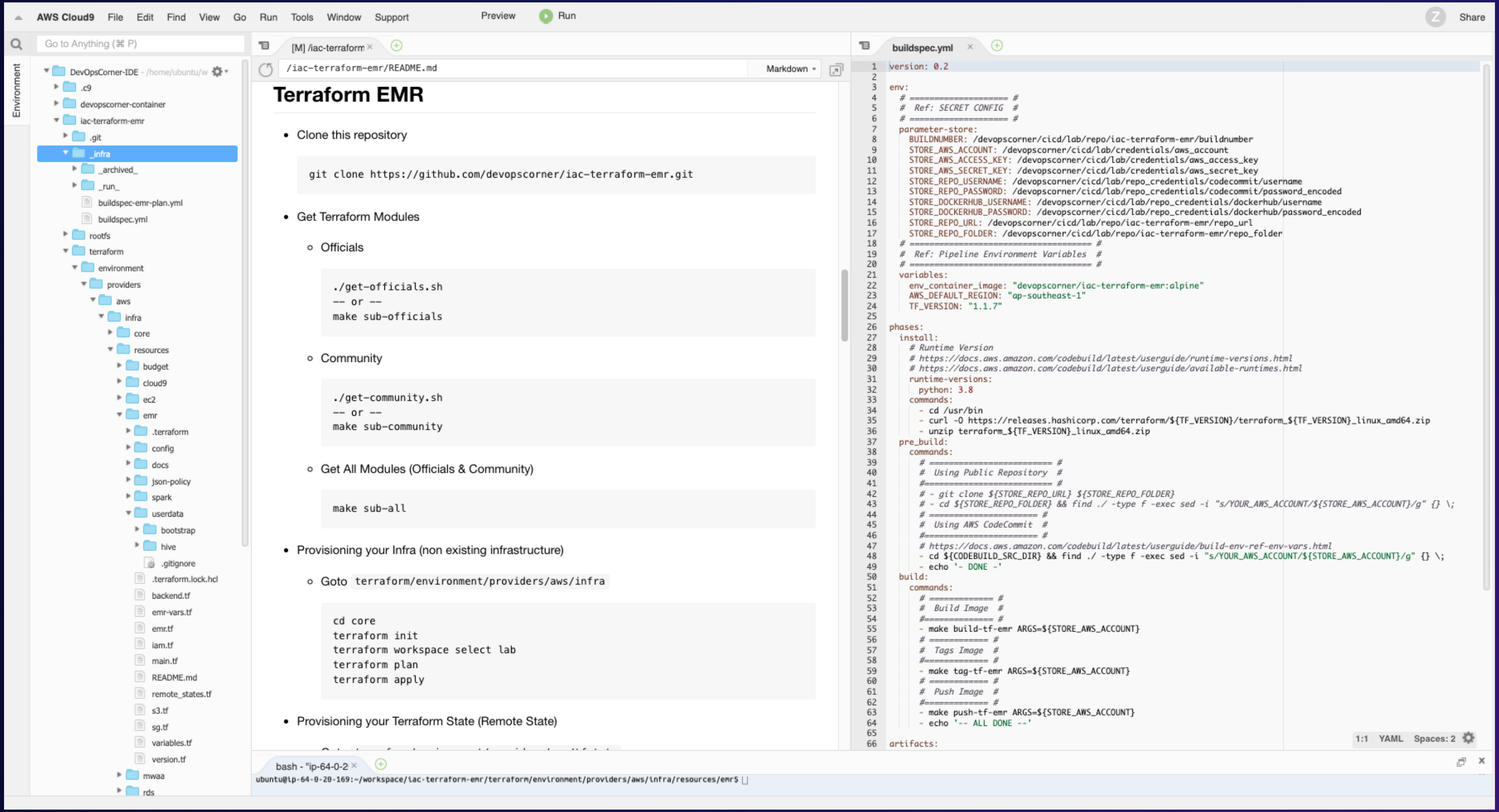
Provisioning Amazon EMR

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)



Terraform EMR

- 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)

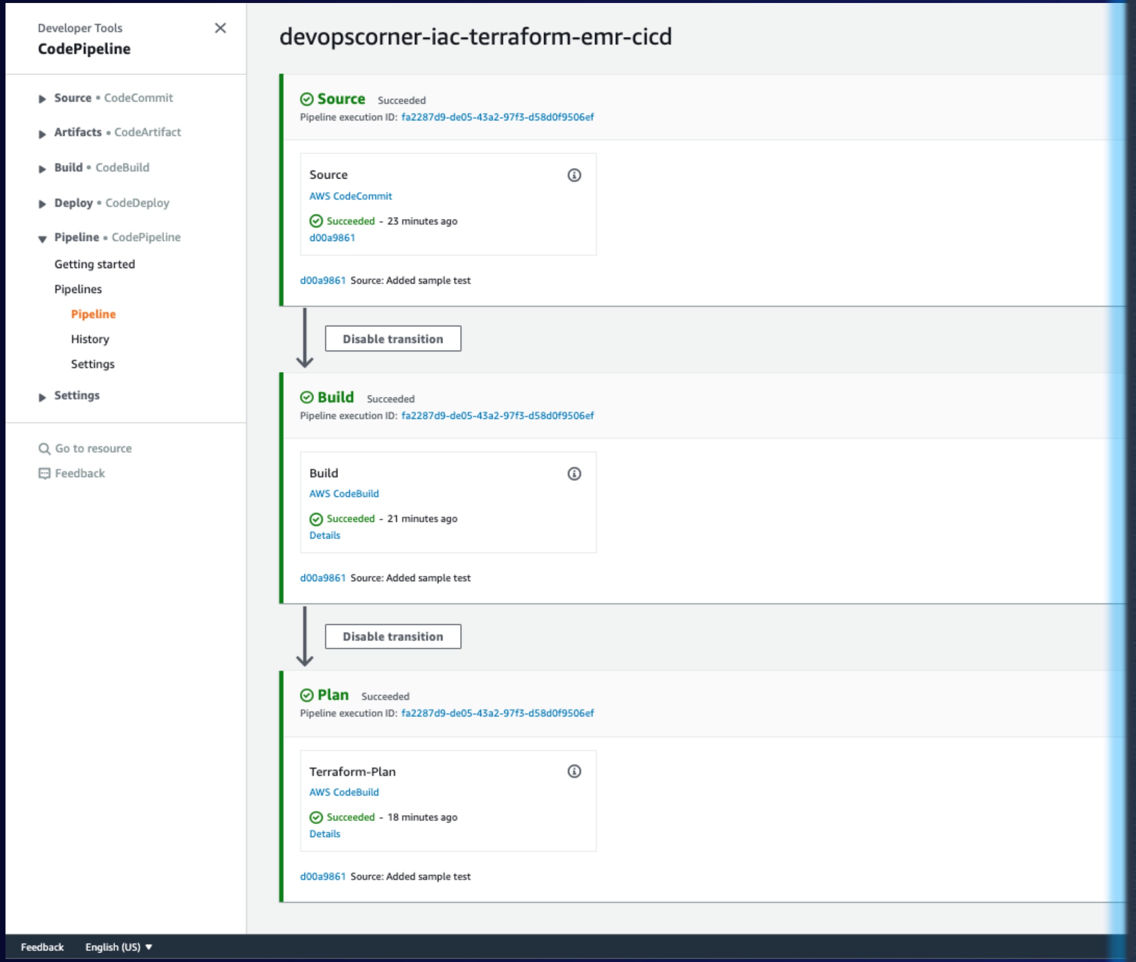
buildspec.yml

```yaml
version: 0.2

env:
  # ==================== #
  #  Ref: SECRET CONFIG  #
  # ==================== #
  parameter-store:
    BUILDNUMBER: /devopscorner/cicd/lab/repo/iac-terraform-emr/buildnumber
    STORE_AWS_ACCOUNT: /devopscorner/cicd/lab/credentials/aws_account
    STORE_AWS_ACCESS_KEY: /devopscorner/cicd/lab/credentials/aws_access_key
    STORE_AWS_SECRET_KEY: /devopscorner/cicd/lab/credentials/aws_secret_key
    STORE_REPO_USERNAME: /devopscorner/cicd/lab/repo_credentials/codecommit/username
    STORE_REPO_PASSWORD: /devopscorner/cicd/lab/repo_credentials/codecommit/password_encoded
    STORE_DOCKERHUB_USERNAME: /devopscorner/cicd/lab/repo_credentials/dockerhub/username
    STORE_DOCKERHUB_PASSWORD: /devopscorner/cicd/lab/repo_credentials/dockerhub/password_encoded
    STORE_REPO_URL: /devopscorner/cicd/lab/repo/iac-terraform-emr/repo_url
    STORE_REPO_FOLDER: /devopscorner/cicd/lab/repo/iac-terraform-emr/repo_folder
  # ==================================== #
  #  Ref: Pipeline Environment Variables  #
  # ==================================== #
  variables:
    env_container_image: "devopscorner/iac-terraform-emr:alpine"
    AWS_DEFAULT_REGION: "ap-southeast-1"
    TF_VERSION: "1.1.7"

phases:
  install:
    # Runtime Version
    # https://docs.aws.amazon.com/codebuild/latest/userguide/runtime-versions.html
    # https://docs.aws.amazon.com/codebuild/latest/userguide/available-runtimes.html
    runtime-versions:
      python: 3.8
    commands:
      - cd /usr/bin
      - curl -O https://releases.hashicorp.com/terraform/${TF_VERSION}/terraform_${TF_VERSION}_linux_amd64.zip
      - unzip terraform_${TF_VERSION}_linux_amd64.zip
  pre_build:
    commands:
      # ========================= #
      #  Using Public Repository  #
      #========================== #
      # - git clone ${STORE_REPO_URL} ${STORE_REPO_FOLDER}
      # - cd ${STORE_REPO_FOLDER} && find ./ -type f -exec sed -i "s/YOUR_AWS_ACCOUNT/${STORE_AWS_ACCOUNT}/g" {} \;
      # ====================== #
      #  Using AWS CodeCommit  #
      #======================= #
      # https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-env-vars.html
      - cd ${CODEBUILD_SRC_DIR} && find ./ -type f -exec sed -i "s/YOUR_AWS_ACCOUNT/${STORE_AWS_ACCOUNT}/g" {} \;
      - echo '- DONE -'
  build:
    commands:
      # ============= #
      #  Build Image  #
      #============== #
      - make build-tf-emr ARGS=${STORE_AWS_ACCOUNT}
      # ============ #
      #  Tags Image  #
      #============= #
      - make tag-tf-emr ARGS=${STORE_AWS_ACCOUNT}
      # ============ #
      #  Push Image  #
      #============= #
      - make push-tf-emr ARGS=${STORE_AWS_ACCOUNT}
      - echo '-- ALL DONE --'

artifacts:
```

```
ubuntu@ip-64-0-20-169:~/workspace/iac-terraform-emr/terraform/environment/providers/aws/infra/resources/emr$
```

Provisioning Amazon EMR

DEPLOY AMAZON EMR – AWS DEVELOPER TOOLS



devopscorner-iac-terraform-emr-cicd

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

Source
AWS CodeCommit
Succeeded - 23 minutes ago
d00a9861

d00a9861 Source: Added sample test

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

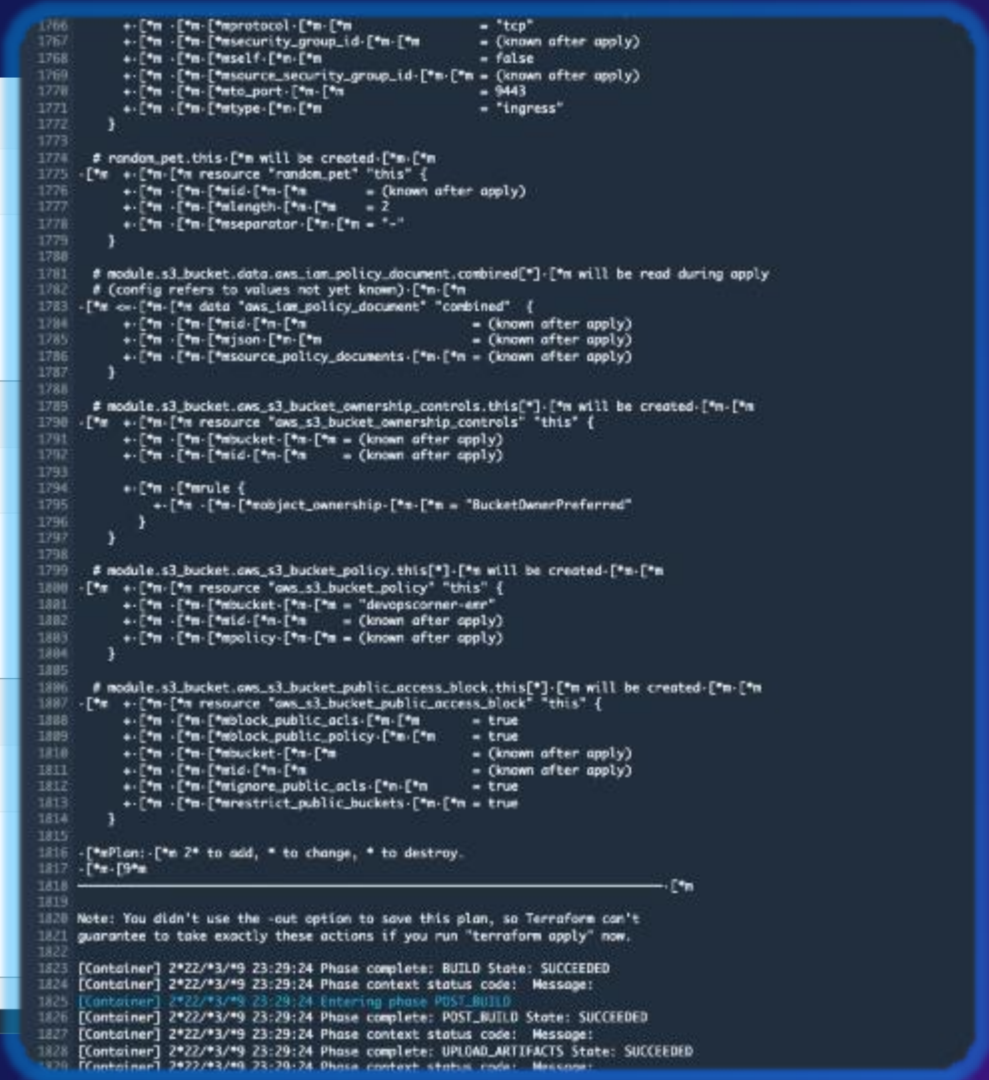
Build
AWS CodeBuild
Succeeded - 21 minutes ago
Details

d00a9861 Source: Added sample test

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

Terraform-Plan
AWS CodeBuild
Succeeded - 18 minutes ago
Details

d00a9861 Source: Added sample test



```
      +·[*m ·[*m·[*mprotocol·[*m·[*m                      = "tcp"
      +·[*m ·[*m·[*msecurity_group_id·[*m·[*m             = (known after apply)
      +·[*m ·[*m·[*mself·[*m·[*m                          = false
      +·[*m ·[*m·[*msource_security_group_id·[*m·[*m = (known after apply)
      +·[*m ·[*m·[*mto_port·[*m·[*m                       = 9443
      +·[*m ·[*m·[*mtype·[*m·[*m                          = "ingress"
    }

  # random_pet.this·[*m will be created·[*m·[*m
·[*m  +·[*m·[*m resource "random_pet" "this" {
      +·[*m ·[*m·[*mid·[*m·[*m        = (known after apply)
      +·[*m ·[*m·[*mlength·[*m·[*m    = 2
      +·[*m ·[*m·[*mseparator·[*m·[*m = "-"
    }

  # module.s3_bucket.data.aws_iam_policy_document.combined[*]·[*m will be read during apply
  # (config refers to values not yet known)·[*m·[*m
·[*m <=·[*m·[*m data "aws_iam_policy_document" "combined"  {
      +·[*m ·[*m·[*mid·[*m·[*m                             = (known after apply)
      +·[*m ·[*m·[*mjson·[*m·[*m                           = (known after apply)
      +·[*m ·[*m·[*msource_policy_documents·[*m·[*m = (known after apply)
    }

  # module.s3_bucket.aws_s3_bucket_ownership_controls.this[*]·[*m will be created·[*m·[*m
·[*m  +·[*m·[*m resource "aws_s3_bucket_ownership_controls" "this" {
      +·[*m ·[*m·[*mbucket·[*m·[*m = (known after apply)
      +·[*m ·[*m·[*mid·[*m·[*m     = (known after apply)

      +·[*m ·[*mrule {
          +·[*m ·[*m·[*mobject_ownership·[*m·[*m = "BucketOwnerPreferred"
        }
    }

  # module.s3_bucket.aws_s3_bucket_policy.this[*]·[*m will be created·[*m·[*m
·[*m  +·[*m·[*m resource "aws_s3_bucket_policy" "this" {
      +·[*m ·[*m·[*mbucket·[*m·[*m = "devopscorner-emr"
      +·[*m ·[*m·[*mid·[*m·[*m     = (known after apply)
      +·[*m ·[*m·[*mpolicy·[*m·[*m = (known after apply)
    }

  # module.s3_bucket.aws_s3_bucket_public_access_block.this[*]·[*m will be created·[*m·[*m
·[*m  +·[*m·[*m resource "aws_s3_bucket_public_access_block" "this" {
      +·[*m ·[*m·[*mblock_public_acls·[*m·[*m       = true
      +·[*m ·[*m·[*mblock_public_policy·[*m·[*m     = true
      +·[*m ·[*m·[*mbucket·[*m·[*m                  = (known after apply)
      +·[*m ·[*m·[*mid·[*m·[*m                      = (known after apply)
      +·[*m ·[*m·[*mignore_public_acls·[*m·[*m      = true
      +·[*m ·[*m·[*mrestrict_public_buckets·[*m·[*m = true
    }

·[*mPlan:·[*m 2* to add, * to change, * to destroy.
·[*m·[9*m
───────────────────────────────────────────────────────────────────────·[*m

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.

[Container] 2*22/*3/*9 23:29:24 Phase complete: BUILD State: SUCCEEDED
[Container] 2*22/*3/*9 23:29:24 Phase context status code:  Message:
[Container] 2*22/*3/*9 23:29:24 Entering phase POST_BUILD
[Container] 2*22/*3/*9 23:29:24 Phase complete: POST_BUILD State: SUCCEEDED
[Container] 2*22/*3/*9 23:29:24 Phase context status code:  Message:
[Container] 2*22/*3/*9 23:29:24 Phase complete: UPLOAD_ARTIFACTS State: SUCCEEDED
[Container] 2*22/*3/*9 23:29:24 Phase context status code:  Message:
```

Deploy Amazon EMR

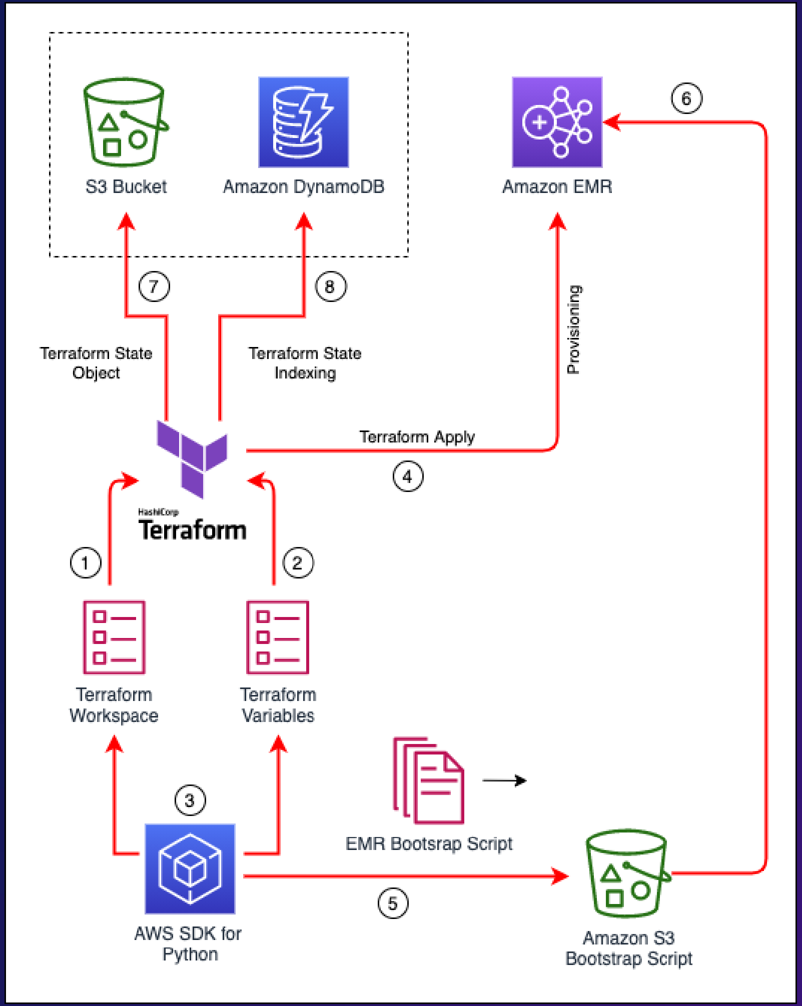
(ETA: 30 minutes)

Provisioning Amazon EMR

BOOTSTRAP SCRIPT & TERRAFORM STACK

Bootstrap script including:

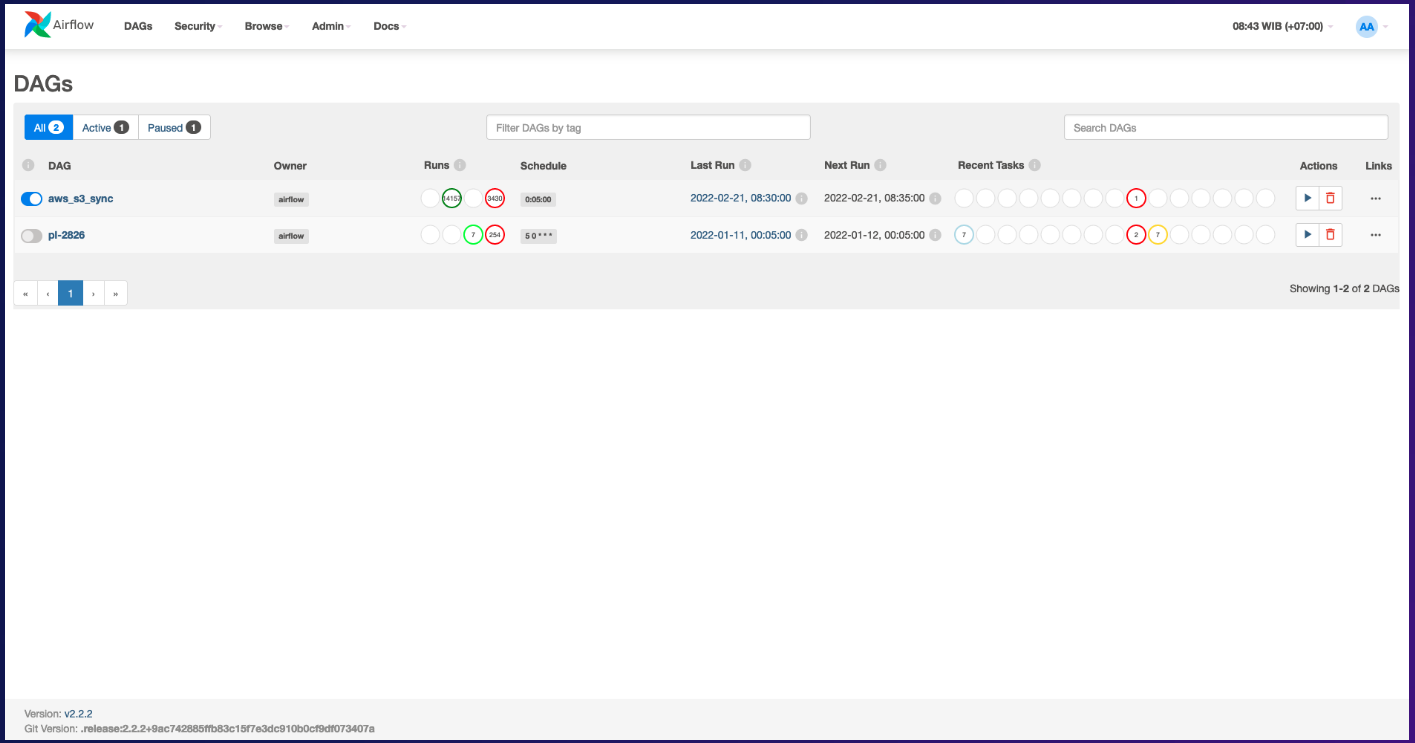
- IAM (Role & Policy)
- Provisioning Master, Core & Task Instance Fleet
- Instance Type Provisioning
- Volume (EBS) Size Instance
- Hadoop Debugging
- Autoscaling
- Monitoring & Log



Provisioning Amazon EMR

RUNNING SPARK JOB

- 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



DAGs

All 2 | Active 1 | Paused 1

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
aws_s3_sync	airflow	14157, 3430	0:05:00	2022-02-21, 08:30:00	2022-02-21, 08:35:00	1	▶ 🗑	•••
pl-2826	airflow	7, 254	5 0 * * *	2022-01-11, 00:05:00	2022-01-12, 00:05:00	7, 2, 7	▶ 🗑	•••

« ‹ 1 › »

Showing **1-2** of **2** DAGs

Version: v2.2.2
Git Version: .release:2.2.2+9ac742885ffb83c15f7e3dc910b0cf9df073407a

Appendix:

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 Builders - Indonesia

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