**APES (AWS pipeline for elemental services)**

“Development of Infrastructure for VOD (Video on Demand) and live streaming using AWS elemental services.”

Our goal is to develop an infrastructure using AWS elemental services for VOD (Video on Demand) and live streaming, so that it can be used directly as needed by the organisation or outsource as per demand in the market

The project requirement includes creating an AWS infra/services pipeline. Pipeline should be able to automate creation of elemental services needed to run live streaming and VOD playback on AWS.

**Architecture:**

**High level architecture diagram for live streaming & VOD**

**Development:**

* Phase 1: Pre-Requisite (Development of Initial Infrastructure)
* Phase 2: Execution and Implementation of Elemental services
* Phase 3: Testing and Rolling out

**Phase 1: Pre-Requisite (Development of Initial Infrastructure for pipeline)**

The following will be developed under the pre-requisite for the initial infrastructure:

1. Repo Creation for IAC
2. AWS code pipeline creation/ GitHub actions creation
3. VPC Module Creation
4. Terraform Initial configuration

**Tools and Technology used**  

The pipeline uses the tools listed below:

* **Terraform for Infrastructure as code (IAC):**

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes both low-level components like compute instances, storage, and networking, as well as high-level components like DNS entries and SaaS features.

* **Terragrunt:**

Terragrunt is a thin wrapper for Terraform that provides extra tools for keeping your Terraform configurations DRY, working with multiple Terraform modules, and managing remote state. We chose terraform because it allowed many flexibilities like reusing the modules, maintaining a remote state so that multiple people can work together like application code in a version control system. Due to these features terraform is becoming a de facto for IAC and hence our obvious choice.

* **GitHub Action for CICD:**

It is a CICD tool offered by GitHub. GitHub as a code repository is used hers, so using GitHub actions for CICD helped us completely automate the infrastructure**.**

**Development Process and Technical Installation Plan**

1. **Repo Creation for IAC:**

**Steps to create Repo:**

* Started by creating a [repo](https://github.com/Deloitte/APES) with the name **APES**.
* The repo has a branch protection rule for main for pushing the code.
* Only the team **APES** and admins have written access to the repo.
* The directory tree structure of the repo is:

**Text

Description automatically generated**

1. **Terraform Initial Configuration:**

1. **Initialise the repo with Terraform variables file**
2. **Bucket Creation for Terraform state file**

**About S3 Bucket:**

S3 is an AWS resource which refers **Standard Storage Service**. It is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

**Steps to create:**

1. Create S3 bucket for storing the terraform state-file.
2. Provide the terraform backend configuration for S3 bucket.
3. **DynamoDB Table module creation for locking of state file**

**About DynamoDB Table module:**

* Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale.
* DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data export tools.
* Written Terraform configuration files for DynamoDB Table.

**Steps to create:**

* Create a DynamoDB table using the Terraform configuration files.

1. **VPC Module creation**
2. Development of VPC module
3. Creation of Subnet Module

**About Subnet Module:**

* A subnet is a range of IP addresses in the VPC. AWS resources can be launched into a specified subnet.
* We use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.

**Steps to create:**

* Create public subnet and private subnets through Terraform IaC.

1. **Creation of Gateway Module:**

**About Internet Gateway:**

* Internet gateway enables you to connect to an EC2 instance in AWS using your local computer. An internet gateway provides a target in your VPC route tables for internet-routable traffic. For communication using IPv4.
* The gateway converts information, data or other communications from one protocol or format to another. A router may perform some of the functions of a gateway. An Internet gateway can transfer communications between an enterprise network and the Internet.

**About NAT Gateway:**

* NAT Gateway is a highly available AWS managed service that makes it easy to connect to the Internet from instances within a private subnet in an Amazon Virtual Private Cloud (Amazon VPC.

**Steps to create:**

Create Internet gateway and the NAT gateway through the Terraform IaC.

1. **Creation of Route table and Security Group:**

**About Route Table:**

A routing table is a set of rules, often viewed in table format, that is used to determine where data packets traveling over an Internet Protocol network will be directed. All IP-enabled devices, including routers and switches, use routing tables.

**Steps to create:**

* Create a route table by using terraform IaC.
* Associate a subnet with a route table.
* Determine which subnets and or gateways are explicitly associated.
* Associate a gateway with a route table.

**About Security Groups:**

* The security group has a list of all the allowed inbound and outbound ports.
* Terraform terminology uses Ingress traffic as inbound and Egress as outbound.
* In the following Terraform configuration, create a Security Group that allows two incoming ports from everywhere.

**Steps to create:**

* Create a security-groups by using terraform IaC.
* The security group has a list of all the allowed inbound and outbound ports.

1. **AWS code pipeline creation/ GitHub actions creation**