**usecase-1-GitHubActions-LocoBuzz**

**Problem Statement**

Business perspective is "Extracting information from social media platforms and delivering a comprehensive analysis report to the customer."

They have 2 environments one with UAT and one with Production and the infrastructure was launched manually over the AWS console and loosing track of the additional infrastrucure provisioning based on the requirements and wanted to move to EKS from monolithic.

**Solution statement**:

Minfy/Bespin has proposed a solution to launch the infrastructure on AWS using Terraform as IAAC for EKS using GitHubactions and migrated the traffic to EKS

**Achivements**:

EKS Infrastructure provisioning using Terraform with githubactions and proper authentication and approval mechanisims for plan and apply considering Security and Validation/approvals as the major agenda

**Outcomes**

End Customer is happy with the time consumption , Security and Validations at different levels in provisioning and maintaining the infrasture with automation.

**Stack/Flow of Implementation(two differnt repos to two differrent repos)**

Create a new repository(Differnet for UAT and PRODUCTION where state files are residing in respective main.tf files).

Add the Credentials on the environment variables in the GitHub.

Create a new directory named workflows under the .github directory, Push yaml files for CI/CD in this directory.

Create a new branch based on requirement with updated code.

Add the constructed terraform files to provision the resource.

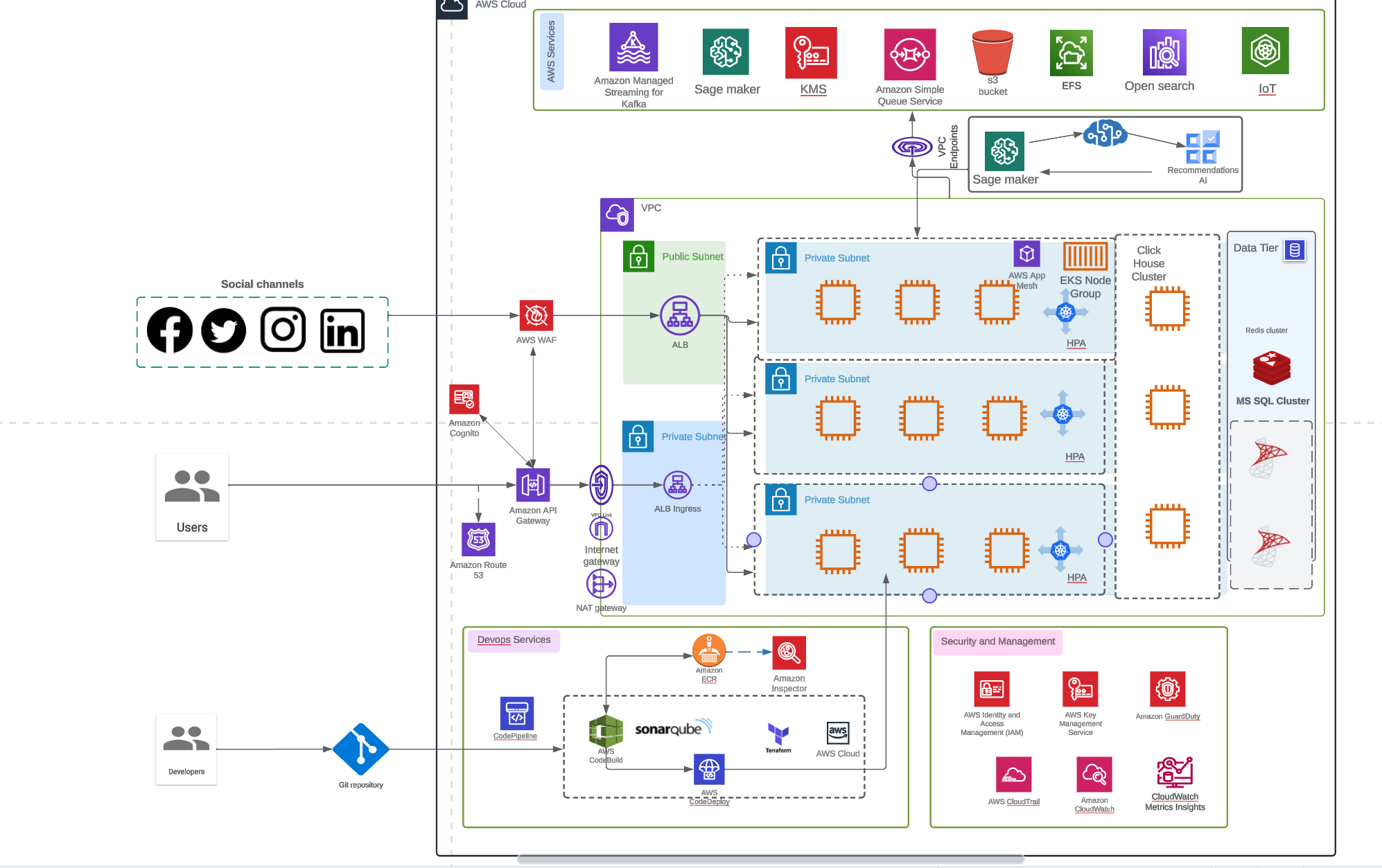
Now Push the code to the newly created branch,

Once the Code has been pushed, Create PR to Master Branch where Terraform Init & Terraform Plan execution output will be visible.

the the approver can see the plan and will take an action by approving and will merge the code to Master .

Terraform apply will trigger & provision the required resource once the data is in Master

**Architecture:**



**Usecase-2-GitHub Actions Propelor(3 different repos to 1 repo)**

**Problem Statement**

Business perspective is "

"Managing the website to facilitate the collection of goods from customers and ensuring the timely and efficient dispatch of these goods to various locations.""

Infrastructre launched with terraform and has 3 different enviornments with 3different repositories and the infrastructure keeps changing they wanted to merge all environments into single repository to ECS Fargate from Monolithic approach

**Solution statement:**

Minfy/Bespin has proposed a solution to launch the infrastructure on AWS using Terraform as IAAC for ECS fargate using GitHubactions for CI/CD and migrated the traffic to ECS Fargate

**Outcomes:**

ECS Farget Infrastructure provisioning using Terraform with githubactions and proper authentication and approval mechanisims for plan and apply considering Security and Validation/approvals as the major agenda with high availability and Automation

End Customer is happy with automation and time-consuming with migration

**Stack/Flow of Implementation (3 different repos/environment to 1 repo 3 environments)**

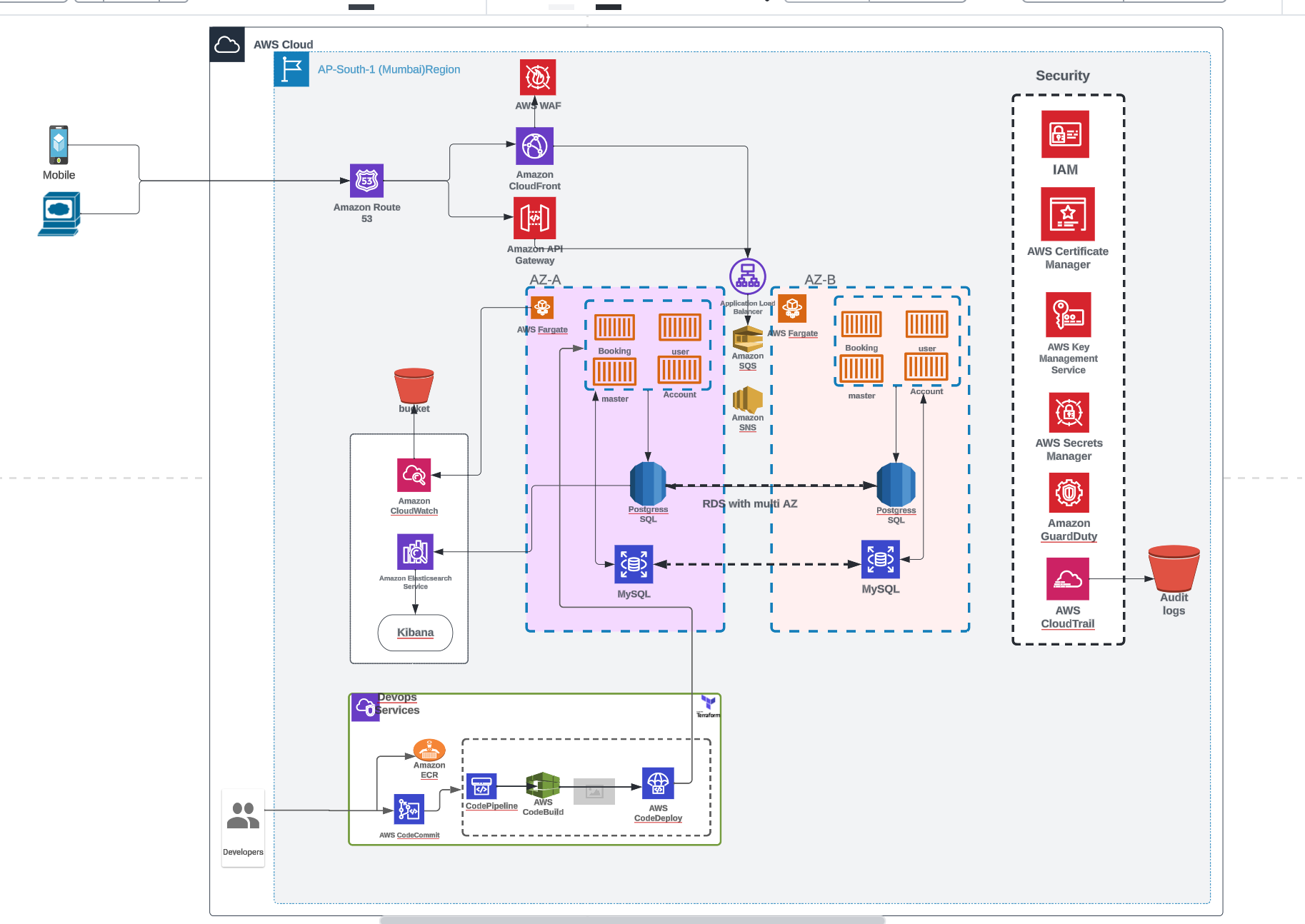
* Create a 1 new repository.
* Add the Credentials on the environment variables in the GitHub.
* Create a new directory named workflows under the .github directory, Push yaml files for CI/CD in this directory.
* Create a new branch based on requirement with updated code.
* Based on environment the variable files will vary acomodating statefile and data specific to enviornment in the variable format(.tfvar)
* State file resides in S3 with dynamodb lock

ex: Production - prod.tfvar

UAT - uat.tfvar

DEV - dev.tfvar

* Add the constructed terraform files to provision the resource.
* Now Push the code to the newly created branch
* Once the Code has been pushed, Create PR to Master Branch where Terraform Init & Terraform Plan execution output will be visible.
* the the approver can see the plan and will take an action by approving and will merge the code to Master .
* Terraform apply will trigger & provision the required resource once the data is in Master.



UseCase3-AWS CodePipeline

**Problem Statement:**

As client infra grows, managing and provisioning infrastructure resources across various environment has become increasingly complex. The manual configuration of servers, databases and network components not only consumes valuable time but also introduces the risk of configuration drifts and inconsistencies. To address these challenges, we are seeking a solution to automate infrastructure provisioning, configuration, and management.

**Proposed Plan of action:**

* IaaC Terraform code for Infrastructure which includes all resources. The variable files are seggregated for each environment
* Creation of Code Commit repo which contains terraform code (main files and variable files) for building infrastructure in different environments.
* Creation of roles for CodePipeline, Codebuild service and etc…
* Creation of IAM policies for attaching to the roles created above through Terraform.
* Email notifications for approval of code changes
* Codebuild configuration (buildspec.yaml) . This includes the steps to plan and apply the terraform in three environments by switching between the workspaces.
* Code Pipeline creation
* Test Pipeline
* Infrastructure changes are made in local by creating new branch and pushed to the code commit repo.
* Creation of Pull Request in AWS console to main branch.
* The cloudwatch events and SNS topics for sending notifications before merging to main branch.
* When the code is merged to main branch pipeline gets triggered .

**Outcomes:**

* Improved deployment speed and resource consistency.
* Reduced manual intervention and human errors in the infrastructure provisioning process.
* Achieving a high level of adoption and satisfaction among the development and operations teams
* Ability to update the configuration of the infrastructure by single push to the repo.

