Six ECS services using single load balancer

# Basic understanding –

In this assignment we are going to create-6 microservices in our which are attached to a load balancer, and other services that we are using are code commit as a version control, Code pipeline which will build the docker image using the buildspec.yml and deploy on the ECS cluster, other thing that we are using the

Route53 to configure domain and sub-domain to each services

AWS SERVICES USED IN ASSINGMENT–

1. VPC
2. ECS
3. Code Commit
4. Code build
5. Code pipeline
6. Route-53
7. Application load balancer
8. ECR
9. Cloud watch
10. Autoscaling

* VPC – Is a private cloud computing environment contained within a public cloud. Essentially, a VPC provide logically isolated environment in the cloud inside the VPC we have created different services such as,
* Internet gateway – the basic purpose of gateway is to allow the communication between the VPC and internet
* Subnet -it is use to define the IP address of the cider block
* 02 var-vpc.tf – is the file where we define the variables we have allocated to the VPC
* security groups – is act as a virtual firewall for controlling the incoming and outgoing traffic which allows or block the Ports
* 03-IAM.tf – contains the basic IAM Roles and Policies that are required the purpose is to allow access to user or services to have compliance ,have basic polices and Roles for
* **ALB.tf –** the file contains the Resource of application load balancer which distribute the load among the different servicesapart form that it consist of the target groups which each service have its own target group
* CICD-pipeline.tf - consist of the basic code for the

Code build – the basic purpose of the code build is to build the image for the cluster using

single Buildspec.yml for each services

Each service have its own pipeline

A)- pre\_build use to login the repository,

B)- build use to build the docker image ,

3)-final stage is to push the image

ECS – In ECS we are creating a single cluster which have 6 farget micro services each service have its own task catering the need of the cluster or we can increase the count of the container in each service according to our need

Autoscaling – we have applied auto scaling for each service when ever a health count goes down it will replace the instance