**Design Document:**

**Purpose:** Aim of this document is to provide detailed information about architecture and deployment strategy for a web application on the AWS Container service.

**Requirement:** As per the input provided, the requirement is to deploy the python hello-wrold application to suitable AWS container service with considering all aspects like Security , High Availability, Scalability and Cost .

**Assumptions**:

* Assuming that the current solution is to deploy a simple web application on AWS container service.

**Design Overview:**

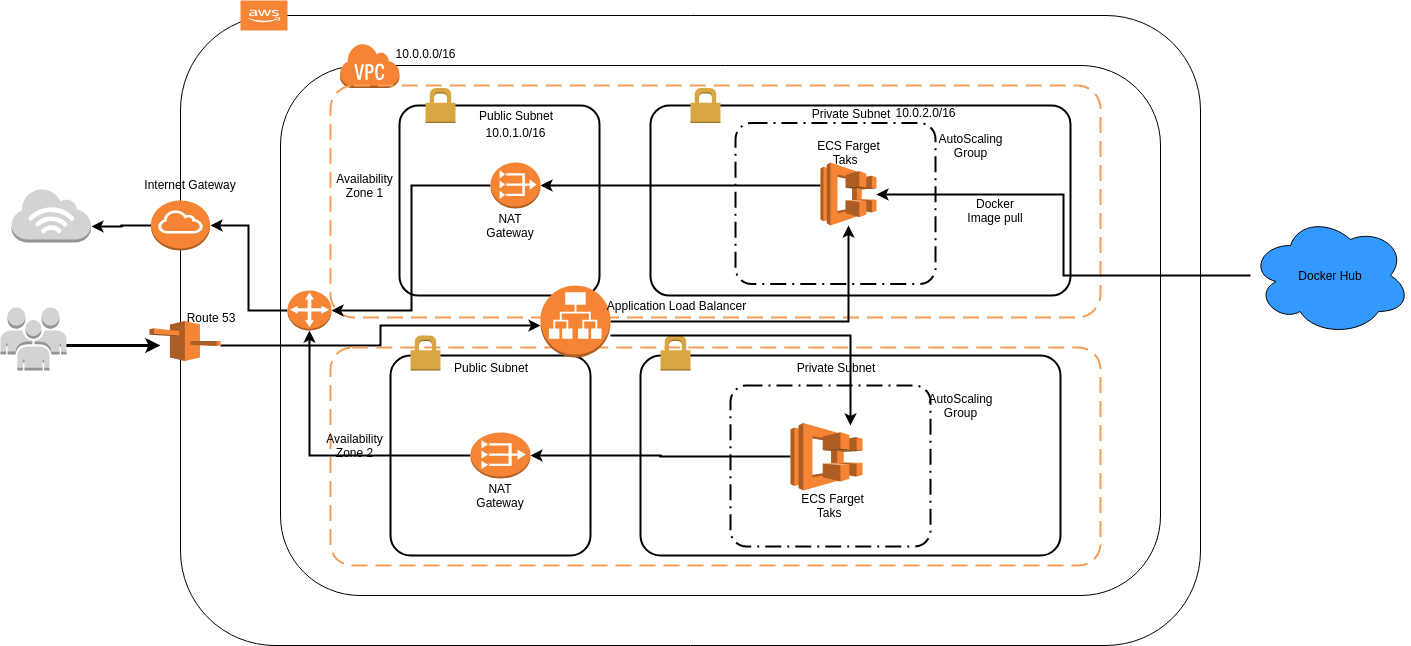
AWS cloud provides multiple compute and Container services to run our applications. If we are planning to run application with major control on internal servers then it is always preferred to use EC2 servers or if we are going with containers, then we should use ECS with EC2 launch type where we have full control over internal servers. On the other hand, if we are not much concerned about internal servers and infra and majorly focusing on our application development then AWS is providing services where we can get rid off managing infra and servers. These services are like AWS lambda, ECS with Farget deployment.

For this current requirement I have considered ECS Farget which is scalable and reliable serverless container service.

When a user requests for a specific URL that request reaches to AWS Route 53 service. Amazon Route 53 is a highly available and scalable cloud Domain Name System(DNS) web service. It is designed to give developers and businesses an extremely reliable and cost effective way to route end users to Internet applications by translating names like www.example.com into the numeric IP addresses like 10.0.2.1 that computers use to connect to each other.

Route 53 then redirects the request to application load balancer, which is configured in Route 53 hosted zone. Once Application load balancer receives request it sends request to any one of the ECS service and once receives the response it sends back the response to Route 53 and then reaches to end user.

**Architecture Diagram:**

****

In this design we have considered AWS ECS service with Farget deployment type to run the container as it is cost efficient and easy to manage serverless container service.

To consider the security aspect we have created a VPC which is a virtual private cloud with two subnets public and private. All the services or front end applications which direclty need to be communicated to internet can be deployed in public subnet and all the services/application which should not be directly exposed to internet should be deployed in private subnet.

Public subnet is attached to internet Gateway which enables the two way internet communication and private subnet is attached to a NAT gateway which allows services inside private subnet to access internet when required.

Docker image once build in local, can be pushed to Docker hub or AWS ECR. When we create task and container definiton in ECS, we provide the image url so that ECS service can pull the image from repository. It is always necessary to keep your docker image safe in repository by implementing required authentication.

To achieve scalability we have used auto scaling with ECS so that when AWS Farget will launch servers in backgroud it will always keep the minimum no. Of servers up and running all the time and based on the load it will scale-out or scale-in to the defined maximum or minimum no. In scaling policy.

For Balacing the load and accessing the service we have used Application Load balancer which works as an entry point. Load Balancer keeps track of the servers configured in LB target group and distribute the traffic to all of these servers equally.

Route 53 is used to manage the DNS and maps the ELB DNS to our domain.

This architecture contains multi AZ deployment to support High Availability. In case of one availability zone failure, resources will be available in another availability zone.