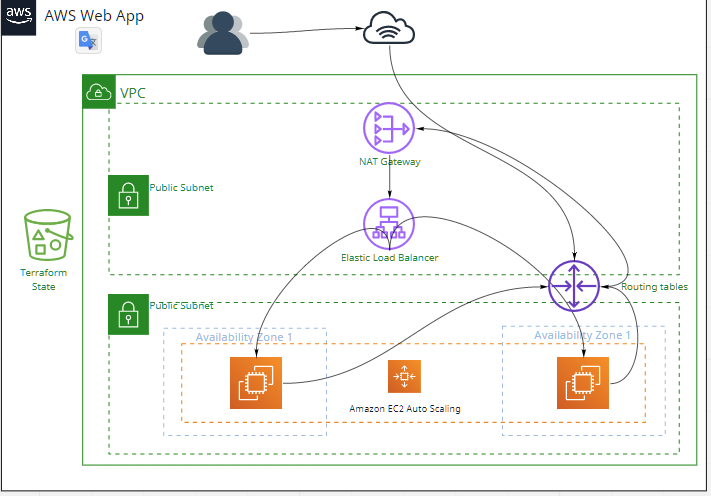
**Architecture (work in progress ) :**

****

**Submission Style:** Please create a public GitHub repository containing all relevant code and send the URL to your hiring contact

* <https://github.com/AfifSoussi/aws_web_app>
* send to hiring manager

# The assignment:

Using Terraform v12, build a module meant to deploy a web application that supports the following design:

-Terraform v12

-Module only

(should I write a wrapper + github actions ?) => later, why not

* It must include a VPC which enables future growth / scale
* It must include both a public and private subnet - where the private subnet is used for compute and the public is used for the load balancers

-Create VPC

Public subnet

Private subnet

-Create Load balancers -> inside public subnet

-Create VMs -> inside private

* Assuming that end-users only contact the load balancers and the underlying instances

are accessed for management purposes, design a security group scheme which supports the minimal set of ports required for communication

-create vms security group

allow http/s ports only through load balancers

allow ssh to vm

allow all ingress traffic

* The AWS generated load balancer hostname will be used for requests to the public facing web application

No need for cloudfront ?

* An autoscaling group should be created which utilizes the latest AWS AMI

-create autoscaling group, use linux AMI to scaling

Create autoscale group

* The instances in the ASG
  + must contain both a root volume to store the application / services
  + must contain a secondary volume meant to store any log data bound for /var/log
  + must include a web server of your choice

-create second volume \* vms and mount /var/log

-install a web server

-create a simple webpage

* Your completed module should include a README which explains the module inputs and any important design decisions you made which may assist in evaluation

-write module architecture / input / output

-draw architecture (miro ?)

-write important decision

*All requirements in this task for configuring the operating system should be defined in the launch configuration and/or the user data script (no external config tools like chef, puppet, etc)*

*-create a launch config or template on aws*

*Create a user data script for linux*

*Your module should not be tightly coupled to your AWS account - it should be designed so that it can be deployed to any arbitrary AWS account*

*-use variables for the account etc*

# Additional Areas to Focus On (Extra credit):

* You must ensure that all data is encrypted at rest
* Ideally, you should design these web servers so they can be managed without logging in with the root key
* We should have some sort of alarm mechanism that indicates when the application is

experiencing any issues

* Configure the autoscaling group to automatically add and remove nodes based on load
* You should assume that this web server may receive high volumes of web traffic, thus you should appropriately manage the storage / growth of logs

-encrypt data at rest ->create encrypted EFS

-management withtout a root key ??

-add alarm -> create cloudwatch integration + alarm

-Storage / growth logs -> EFS dynamic elasticity / collect logs in cloudwatch and add deletion script.

Usefui links :

<https://www.bogotobogo.com/DevOps/Terraform/Terraform-VPC-Subnet-ELB-RouteTable-SecurityGroup-Apache-Server-2.php>

<https://docs.aws.amazon.com/efs/latest/ug/wt2-apache-web-server.html>

encryption at rest

<https://docs.aws.amazon.com/efs/latest/ug/efs-enforce-encryption.html>

<https://aws.amazon.com/premiumsupport/knowledge-center/public-load-balancer-private-ec2/>

<https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html>

<https://aws.amazon.com/blogs/storage/automate-mounting-amazon-efs-file-systems-from-the-ec2-launch-instance-wizard/>

nat gateway + ELB

<https://aws.plainenglish.io/provisioning-aws-infrastructure-using-terraform-vpc-private-subnet-alb-asg-118b82c585f2>