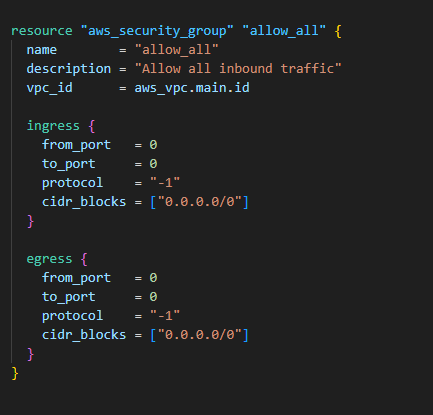
Coalfire Terraform coding challenge

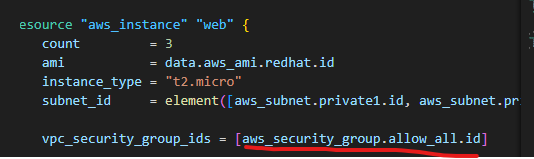
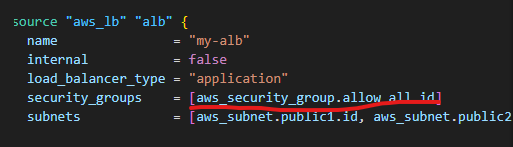
Question 1

After reviewing the entire repo regarding terraform configuration files, specifically the part of configuring security groups for EC2 instance and application load balancer’s network traffic, here is my thought on making changes to these files in order to implement a more least privileged configuration for deployment.

In the given network.tf file, we can see that the author specified a “aws\_security\_group” “allow\_all” resource configuration. But under the security group configuration, the author is allowing all type of network traffic coming in and out to the security group. (Line 44 to Line 54; the ports are configured as 0, ip ranges are configured as 0.0.0.0/0, the protocols are configured as -1)



Meanwhile, from the compute.tf file, I can see that this “aws\_security\_group” config is attached to both the aws ec2 instance and application load balancer which means the ec2 and alb are all public exposed with no network access restrictions.



To implement least privileged network access rule by using security groups, we need to modify the security group configuration in the network.tf file. Ideally, we would need to create 2 security groups with different network rules and attach them to the ec2 instance and application load balancer.

Ingress:

For ec2 instance, since it is a Red hat Linux server, we can have port 22/TCP opened in order to allow connection between the server and the engineer. I will also allow port 80 opened since it’s hosting the application (assuming web application so it goes through internet) and end user would need to connect to the application through this port in order to get the service. Additionally, load balancer’s listener is listening to this port according to the compute.tf file. As for cider blocks, typically the VPC would need to be integrated with all infrastructure network by peering or VPN, engineer would need to connect to the infrastructure network first in order to connect to the server through Private IP. I’m leaving the ingress cider blocks all opened for now because I’m not sure if user will be using office IP, or using VPN, or using aws bastion host to connect to the server(we can connect through public IP as well but it’s not secured)

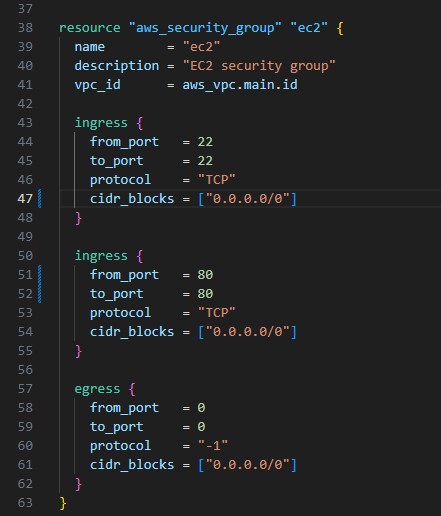
For application load balancer, since it’s an application, most of the time we can have port 80 and 443 opened so the external traffic can go through it from internet. (port 443 is more secured than port 80 since it’s encrypted). But since the listener is only listening to port 80 so we just only have port 80 opened at this time. We are not specifying the subnet because traffic could come from anywhere from internet.

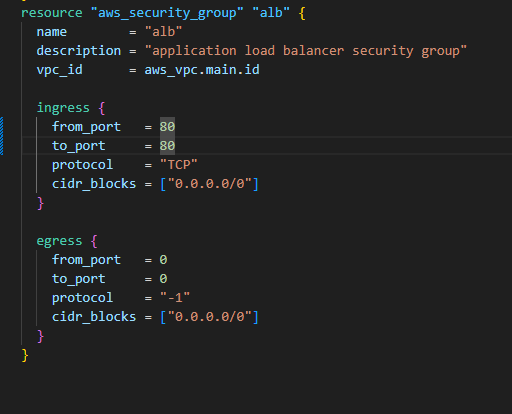
Egress:

I’m assuming that the server would need to connect to other services(end user, database, serverless function,etc…) So I will leave the egress rule as opened to all ports.

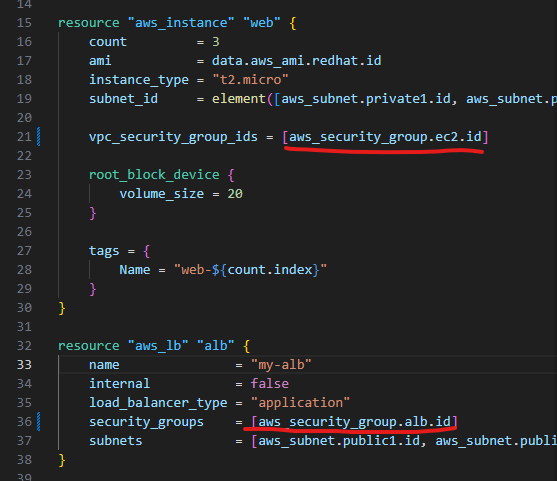
I will be modifying the existing security group configuration and use it for EC2 instance, then I will create a new security group configuration for application load balancer.

These would be the configuration file after my modification:

 Line38 to62

Line64 to 82

After I modified the security group configuration in network.tf file, I would still need to attach the new security groups to the EC2 instance and application load balancer.

Line21 and Line 36

Question 2

In order to expand the mount point of Linux data file system, we need to utilize the following commands.(using sudo to pertain highest permission on system)

sudo yum install cloud-utils-growpart

sudo growpart

sudo resize2fs

Before we run these commands in bash script, we would need to identify the partition that linux server is using normally, /dev/xvda or dev/sda. I’m using partition 1 here.

Since /dev/sda is usually used for legacy device but not included virtualized disk so I’m using /dev/xvda under this circumstance. And also most of the time, Linux is using ext4 file system because it’s widely supported so I’m using ext4 here.

I’m not including the specific volume size here because the aws growpart command will automatically detect the increased unused volume and expand it.

Please review following bash script, I have also included the script in my repo(EbsExpand.sh) so feel free to check from there as well.

#!/bin/bash

# The name of the EBS volume

VOLUME="/dev/xvda1"

# 1. Grow the partition

# install the growpart

sudo yum install -y cloud-utils-growpart

# Expand the partition

echo "Expanding the partition"

sudo growpart "$VOLUME" 1

# 2. Resize the filesystem

# For ext4 filesystem

echo "Resizing the ext4 filesystem..."

sudo resize2fs "${VOLUME}1"

#conclusion

echo "Disk resized successfully."