Creating ECS Cluster behind Application Load Balancer with Fargate using Terraform

Source code: https://github.com/khelli07/terraform-aws-alb-fargate

How I implemented it

1. I tried to search for Terraform tutorial that is available on YouTube.
2. After understanding that Terraform is basically "mapping" the AWS console to code, I tried to make the application load balancer from the web console guided by official documentation. There are some important notes on the docs, such as

   4. For **Choose a target type**, select the target type.

      > **Important**
      >
      > If your service's task definition uses the `awsvpc` network mode (which is required for the Fargate launch type), you must choose **IP addresses** as the target type This is because tasks that use the `awsvpc` network mode are associated with an elastic network interface, not an Amazon EC2 instance.

   which is luckily aligned with the task (AWS Fargate launch type).
3. Then, I would search for the tutorial for this task such as this and this. I tried to walkthrough the code and change some parameters to understand them.
4. I also saw the terraform registry documentation to see what does a parameter means and to see other possible parameters.

Difficulties encountered

Honestly, this is my first hands-on experience with cloud platform, let alone Amazon Web Services. So, wrapping my head around the assignment was not easy for me. I get the idea that Terraform helps developers, especially cloud/infrastructure engineer to deploy cloud facilities with declarative codes. But, understanding the code in the examples was not easy for me, e.g. at first I do not know why VPC's "cidr_blocks" has a value of "X". Even so, I did understand some of them along the way. Hence, I did not only learn Terraform but also the AWS itself.

How I test the cluster is working

Since I specify the output in my code,

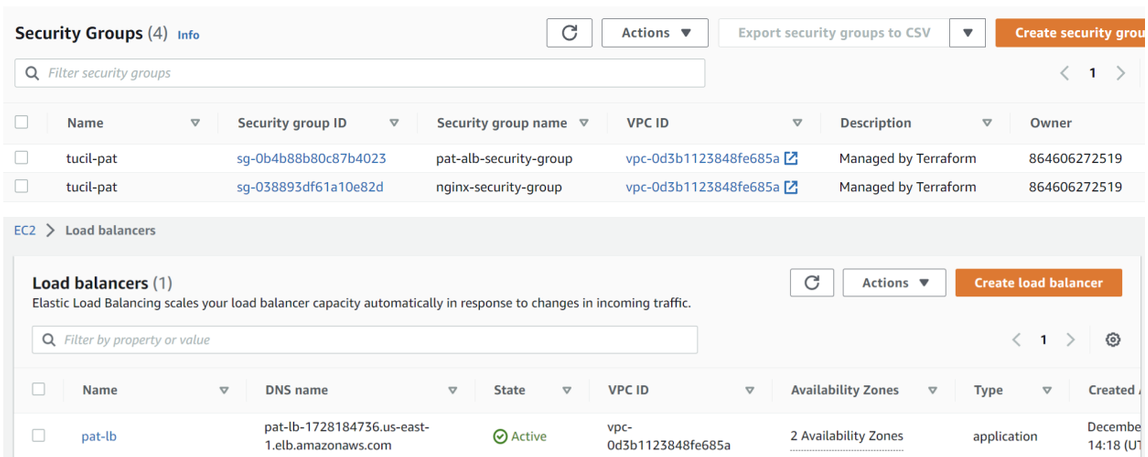
```
output "load_balancer_ip" {
  value = aws_lb.default.dns_name
}
```

Terraform will specify the output of the load balancer's ip.

```
khelli07@LAPTOP-NV03QS44:~/task/pat-IaC-terraform$ terraform apply "tfplan"
aws_ecs_service.hello_nginx: Creating...
aws_ecs_service.hello_nginx: Creation complete after 2s [id=arn:aws:ecs:us-east-1:864606272519:service/pat-cluster/hello-nginx-service]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

load_balancer_ip = "pat-lb-1728184736.us-east-1.elb.amazonaws.com"
```

When visited, nginx welcome page appear. This means that the web services is now accessible.

I also check everything that I specified in the code, such as VPC, subnets, security groups, target groups, the load balancer itself, and so on from the AWS web console. Here are some examples.



What I have learned

1. Basics of AWS instances.
2. Terraform is an infrastructure as code tool, which help us manage and provision cloud resources.
3. In Terraform, we need to specify things like VPC, subnets, routing table, or anything that are created by AWS under the hood if we use web console. This is why Terraform code would specify more resources than web console.
4. There are other types of load balancer (not just application) such as gateway and network LB which works on different OSI layer.

Bonus: Auto scaling

For auto scaling, I use this tutorial to understand and help me implement it. Here's how I did it:

1. I created an IAM role and attached the role to policy. I need to grant the program an access so it can create more instances when some conditions are met.
2. In the tutorial, the auto scaling works based on CPU and RAM, but in this task, we use request count per target. Hence, some adjustments are needed.
3. So, I search for AWS official documentation to see other predefined scaling metrics and change the code based on that.

For testing, I use ApacheBench and check the console if the services are scaled.

Side notes

I use my own AWS account since STEI account still has permission error.