

Cloud Computing

Cloud Computing

Task-1 Development Phase

Professor – Georgi Dimchev

Name – Aditya Modi

Matriculation number – 92115852

Date – 28 June 2023

Webpage features

The website is highly available.

The webpage can be accessed from people all-around the globe and should not experience latency.

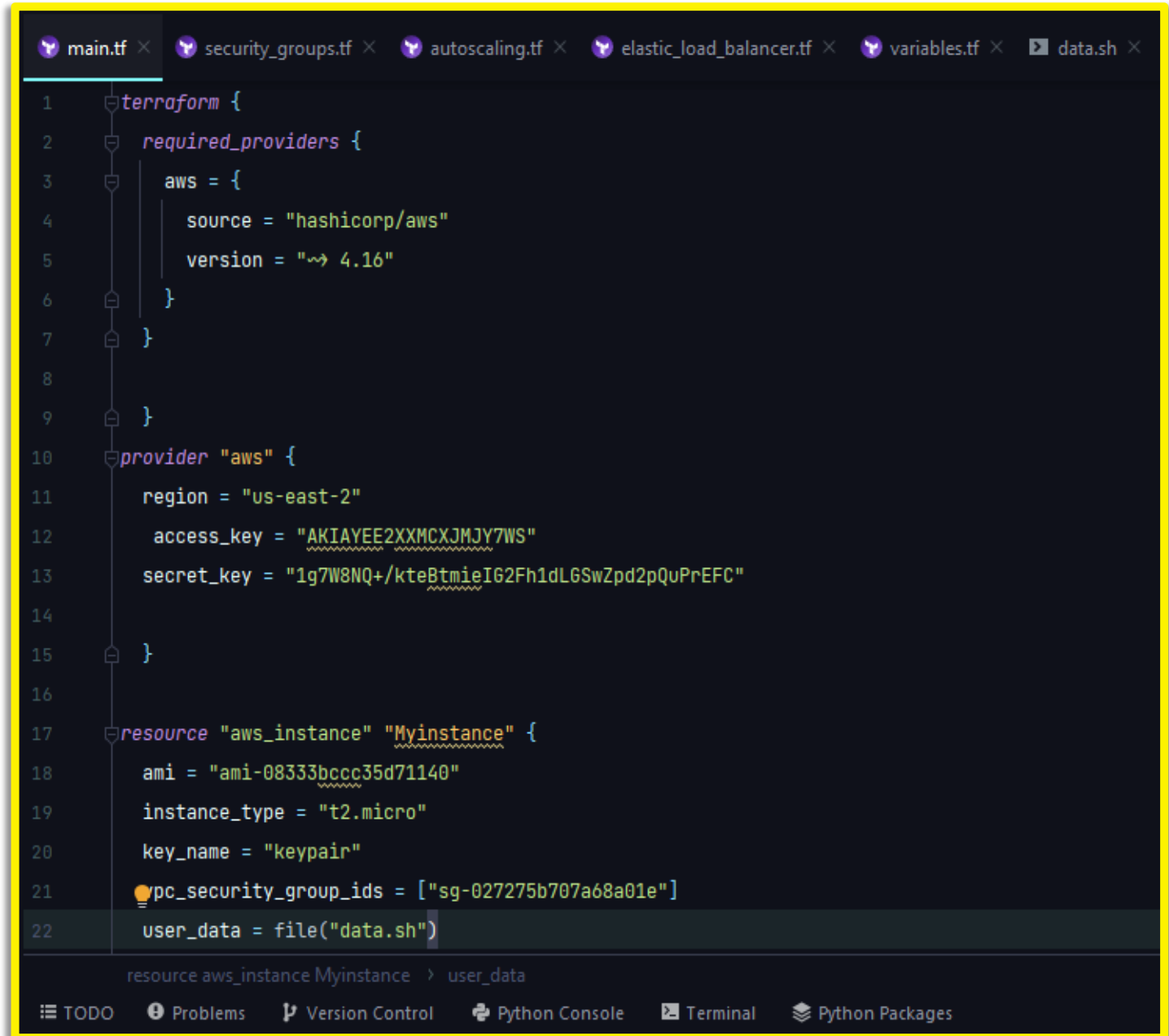
The backend automatically autoscale itself if more visitors come to the webpage.

Setting up instance in terraform

We have set up our main file in terraform. Assigned provider and access key.

we have discussed that I am going to use elastic compute cloud to launch my webpage.

For using it we need a virtual image. We have to define the image we use using Linux image, security group and a key pair.



```
main.tf x security_groups.tf x autoscaling.tf x elastic_load_balancer.tf x variables.tf x data.sh x
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "~> 4.16"
6     }
7   }
8 }
9 }
10 provider "aws" {
11   region = "us-east-2"
12   access_key = "AKIAEE2XXMCXJMZY7WS"
13   secret_key = "1g7W8NQ+/kteBtmieIG2Fh1dLGSwZpd2pQuPrEFC"
14 }
15 }
16
17 resource "aws_instance" "Myinstance" {
18   ami = "ami-08333bccc35d71140"
19   instance_type = "t2.micro"
20   key_name = "keypair"
21   vpc_security_group_ids = ["sg-027275b707a68a01e"]
22   user_data = file("data.sh")
23 }
```

resource aws_instance Myinstance > user_data

TODO Problems Version Control Python Console Terminal Python Packages

Security groups

We setup a separate file for our security groups. Security group acts as a firewall that controls to and from the resources in the virtual private cloud (VPC). We enable inbound traffic from HTTP and HTTPS sources. We have also defined the protocol and port range.

```
main.tf × security_groups.tf × autoscaling.tf × elastic_load_balancer.tf × variables.tf × data.sh ×
1
2 resource "aws_security_group" "sg" {
3     name = "sg"
4     description = "security group"
5     vpc_id = "vpc-0ef05830c36d35c9a"
6     ingress {
7         from_port = 80
8         protocol = "tcp"
9         to_port = 80
10        cidr_blocks = ["0.0.0.0/0"]
11    }
12    ingress {
13        from_port = 443
14        protocol = "tcp"
15        to_port = 443
16        cidr_blocks = ["0.0.0.0/0"]
17    }
18
19    tags = {
20        Name = "sg for subnets"
21    }
22 }
```

Auto scaling groups

EC2 auto scaling group helps us to increase the number of instance if more visitors come to our webpage. It adds or remove the instance as per needs. We have specified the maximum number of instance to 4 and minimum to 1. The launch template will add the html file to the instance.

```
main.tf × security_groups.tf × autoscaling.tf × elastic_load_balancer.tf × variables.tf × data.sh ×

1 resource "aws_launch_template" "launch_template" {
2     name = "launch_template"
3     image_id = "ami-08333bccc35d71140"
4     instance_type = "t2.micro"
5     key_name = "keypair"
6     vpc_security_group_ids = ["sg-027275b707a68a01e"]
7     user_data = base64encode(file("data.sh"))
8
9 }
10
11 resource "aws_autoscaling_group" "auto_scale" {
12     name = "auto_scale"
13     availability_zones = ["us-east-2a", "us-east-2b"]
14     desired_capacity = 1
15     max_size = 4
16     min_size = 1
17     health_check_grace_period = 300
18     health_check_type = "ELB"
19     depends_on = [aws_lb.aws_load_balancer]
20     target_group_arns = [aws_lb_target_group.alb_target_group.arn]
21
22 }
```

Elastic load balancer

```
main.tf x security_groups.tf x autoscaling.tf x elastic_load_balancer.tf x variables.tf x data.sh x
1 resource "aws_lb" "aws_load_balancer" {
2   name = "elastic-load-balancer"
3   load_balancer_type = "application"
4   security_groups = ["sg-027275b707a68a01e"]
5   subnets = ["subnet-06cbb291201d2be26", "subnet-063fcd5d531b9d972"]
6
7   tags = {
8     name = "aws elastic load balancer"
9   }
10 }
11
12 resource "aws_lb_target_group" "alb_target_group" {
13   name = "aws-target-group"
14   target_type = "instance"
15   port = 80
16   protocol = "HTTP"
17   vpc_id = "vpc-0ef05830c36d35c9a"
18
19   tags = {
20     name = "target group"
21   }
22   health_check {
```

Elastic load balancer will distribute all the traffic to healthy EC2 instance. Load balancer increases the availability and the fault tolerance of the application. We are using application load balancer, it maintains the performance and availability of the application. It conducts different health checks. Each of the target group is used to route the traffic through one or more registered target.

Bash Script

The bash script is used to run the html page. I have already uploaded the html page on my GitHub so every time the instance run. It will download the zip file from GitHub to the main storage directory. Than it will run the html file every time our instance starts.

A screenshot of a terminal window with a dark background and yellow border. The terminal shows a bash script with 11 lines of code. The script starts with a shebang, then uses sudo to run system updates and install httpd. It then changes to the /var/www/html directory, downloads a zip file from a GitHub repository using wget, unzips it, copies the contents to the current directory, and finally enables and starts the httpd service. The terminal has several tabs at the top, with 'data.sh' being the active one.

```
1  #!/bin/bash
2  sudo su
3  yum update -y
4  yum install -y httpd
5  cd /var/www/html
6  wget https://github.com/Aditya-modi20/iu/archive/refs/heads/main.zip
7  unzip main.zip
8  cp -r iu-main/* /var/www/html/
9  rm -rf iu-main main.zip
10 systemctl enable httpd
11 systemctl start httpd
```

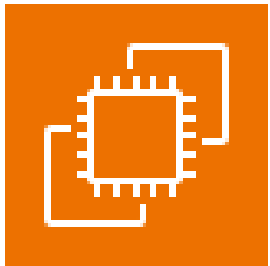
A view of our webpage



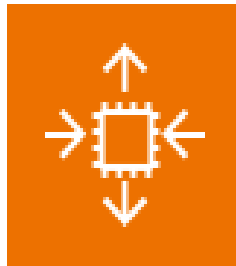
Tools used



[Amazon web service](#)



[EC2 instance](#)



[EC2 auto scaling](#)



[Terraform](#)

Progress and Opportunities for improvement

I am still testing my webpage for the given requirements and also finding ways for improvement. I have tested the webpage in terms of availability and autoscaling. The terraform code is very simple and understandable. It can be used on any pc with terraform installed on it. I have created a different file for different AWS tools, which will help to user to understand the code better and find potential problems. At this point we have spend less than a \$1 in cost of setting up all the tools and resources.

Well at this point I can see there are ton of improvement that can be made to make this webpage more user friendly and powerful to handle more task and we can add more features too. But for this project we have a specific requirement. So, we are focused on that. We can use tools like s3 bucket with a specific domain name and route 53 to make website accessible by everyone. Aws also provide free SSL certificate to securely deliver website over HTTPS using it will make our webpage more secure against different malicious attack.