Home / AWS / Guided Lab / Create an Application Load Balancer to distribute the incoming traffic between two EC2 Instances using Terraform

Create an Application Load Balancer to distribute the incoming traffic between two EC2 Instances using Terraform

Level: **Fundamental**

Amazon EC2 Amazon Web Services Elastic Load Balancing Terraform

0h 56m 22s left

End Lab

Open Console

Validation

Lab Credentials

User Name

Whiz_User_80425.87611124

Password

42ed8571-7073-424d-8ac5-f681539292f1

Access Key

AKIA5MJNUPSBG4LVBLAJ

Secret Key

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

Lab Resources

No Lab Resources Found

Support Documents —

1. FAQs and Troubleshooting

Need help?

How to use Hands on Lab

Troubleshooting Lab

FAQs

Submit Feedback | Share

Lab Overview Lab Steps Lab Validation

Cloud Architect

Compute, Networking, Infrastructure

Lab Steps

Task 1: Sign in to AWS Management Console

1. Click on the **Open Console** button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,
    - Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
    - Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button.
3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1.**

Note : If you face any issues, please go through **FAQs and Troubleshooting for Labs**.

Task 2: Setup Visual Studio Code

In this task, we are going to set up Visual Studio Code, which is a source code editor. It is used in this lab to write and manage the Terraform configuration files.

1. Open the visual studio code.
2. If you have already installed and using Visual studio code, open a new window.
3. A new window will open a new file and release notes page (only if you have installed or updated Visual Studio Code recently). Close the Release notes tab.
4. Open Terminal by selecting View from the Menu bar and choose Terminal.
5. It may take up to 2 minutes to open the terminal window.
6. Once the terminal is ready, let us navigate to the Desktop.

```
cd Desktop
```

7. Create a new folder by running the below command.

```
mkdir task_10004_elb
```

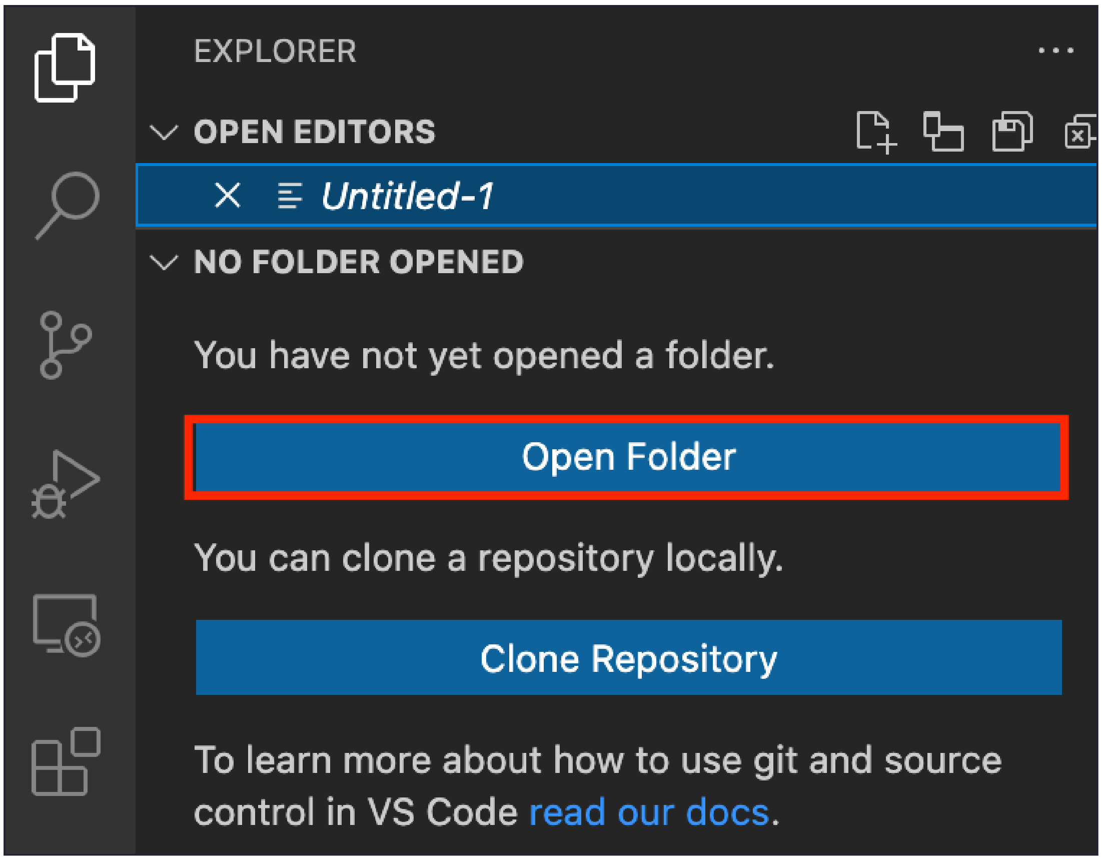
8. Change your present working directory to use the newly created folder by running the below command:

```
cd task_10004_elb
```

9. Get the location of the present working directory by running the below command:

```
pwd
```

10. Note down the location, as you will open the same in the next steps.
11. Now click on the first icon Explorer present on the left sidebar.
12. Click on the button called Open folder and navigate to the location of folder **task_10004_elb**.



You have not yet opened a folder.

You can clone a repository locally.

To learn more about how to use git and source control in VS Code read our docs.

13. (Optional) Click on Authorize button for allowing Visual Studio Code to use the task_10004_elb folder. This will only be asked when you have been using Visual Studio code for a while as you are allowing a new folder to be accessed by VSC.

14. Visual Studio Code is now ready to use.

short description and a default value.

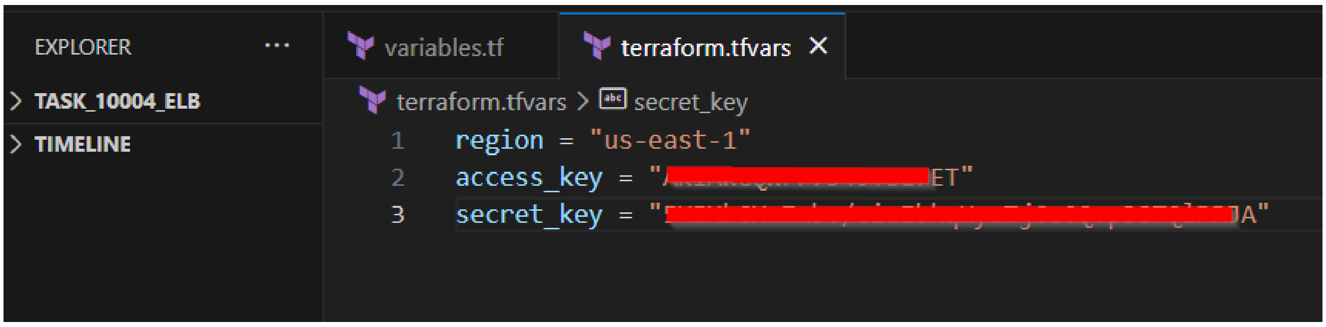
1. To create a variable file, expand the folder **task_10004_elb** and click on the **New File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note:** Don't change the location of the new file, keep it default, i.e. inside the **task_10004_elb** folder.
4. Paste the below contents in **variables.tf** file.

```
variable "access_key" {
    description = "Access key to AWS console"
}

variable "secret_key" {
    description = "Secret key to AWS console"
}
variable "region" {
    description = "Region of AWS VPC"
}
```

5. In the above content, you are declaring a variable called, access_key, secret_key, and region with a short description of all 3.
6. After pasting the above contents, save the file by pressing **ctrl + S**.
7. Now expand the folder **task_10004_elb** and click on the **New File** icon to add the file.
8. Name the file as **terraform.tfvars** and press **Enter** to save it.
9. Paste the below content into the **terraform.tfvars** file.

```
region = "us-east-1"
access_key = "<YOUR AWS CONSOLE ACCESS ID>"
secret_key = "<YOUR AWS CONSOLE SECRET KEY>"
```

10. In the above code, you are defining the dynamic values of variables declared earlier.
11. Replace the values of access_key and secret_key by copying from the lab page.
12. After replacing the values of access_key and secret_key, save the file by pressing **Ctrl + S.**



```
region = "us-east-1"
access_key = "[redacted]ET"
secret_key = "[redacted]A"
```

Task 4: Create EC2, ELB and its components in main.tf file

In this task, you will create a **main.tf** file where you will add details of the provider and resources.

1. To create a **main.tf** file, expand the folder **task_10004_elb** and click on the **New File** icon to add the file.
2. Name the file as **main.tf** and press **Enter** to save it.
3. Paste the below content into the **main.tf** file.

```
provider "aws" {
    region     = "${var.region}"
    access_key = "${var.access_key}"
    secret_key = "${var.secret_key}"
}
```

4. In the above code, you are defining the provider as aws.
5. Next, we want to tell Terraform to create a Security Group within AWS EC2, and populate it with rules to allow traffic on specific ports. In our case, we are allowing the tcp port 80 (HTTP).
6. We also want to make sure the instance can connect outbound on any port, so we're including an egress section below as well.
7. Paste the below content into the **main.tf** file after the provider.

```
############ Creating Security Group for EC2 & ELB ############

resource "aws_security_group" "web-server" {
    name        = "web-server"
    description = "Allow incoming HTTP Connections"
    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
}
```

8. Let's add another set of code after security group creation where you will create 2 EC2 instances.
9. In the below code, we have defined the Amazon Linux 2 AMI. The AMI ID mentioned above is for the us-east-1 region.
10. We have mentioned the instance type as t2.micro and instance count to be created as 2.
11. We have mentioned the resource which SSH key to use (which is already present in your AWS EC2 console). The security group ID is automatically taken by using the variable which we have set during the creation process.
12. We have added the user data to install the apache server and add a html page.

13. We have provided tags for the EC2 instances.

```
 ################## Creating 2 EC2 Instances ##################

resource "aws_instance" "web-server" {

    ami                = "ami-01cc34ab2709337aa"
    instance_type      = "t2.micro"
    count              = 2
    key_name           = "whizlabs-key"
    security_groups = ["${aws_security_group.web-server.name}"]
    user_data = <<-EOF
       #!/bin/bash
       sudo su
        yum update -y
        yum install httpd -y
        systemctl start httpd
        systemctl enable httpd
        echo "<html><h1> Welcome to Whizlabs. Happy Learning from $(hostname -
f)...</p> </h1></html>" >> /var/www/html/index.html
        EOF
    tags = {
        Name = "instance-${count.index}"
    }
}
```

14. Let's add another set of code after EC2 Instances creation where you will define the data source to get the details of vpc_id and subnet_id's.
15. Since we will be using the default VPC for the creation of ELB, we need the vpc_id and subnet_id's. So, we will be using aws_vpc data source to provide details about the default VPC.
16. We will also use the data source aws_subnet to provide a set of subnet id's for a vpc_id.

```
###################### Default VPC and Subnets ######################

data "aws_vpc" "default" {
    default = true
}

data "aws_subnet" "subnet1" {
    vpc_id = data.aws_vpc.default.id
    availability_zone = "us-east-1a"
}

data "aws_subnet" "subnet2" {
    vpc_id = data.aws_vpc.default.id
    availability_zone = "us-east-1b"
}
```

17. Let's add another set of code after the data source of VPC. Here, we will be creating the Target group.
18. In the below code, we have provided the health check details.
19. We have provided the protocol as HTTP and port as 80.
20. The target type is instance and the vpc_id of the default VPC is taken from the data source variable.

```
#################### Creating Target Group ####################

resource "aws_lb_target_group" "target-group" {
    health_check {
        interval            = 10
        path                = "/"
        protocol            = "HTTP"
        timeout             = 5
        healthy_threshold   = 5
        unhealthy_threshold = 2
    }
    name          = "whiz-tg"
    port          = 80
    protocol      = "HTTP"
    target_type   = "instance"
    vpc_id = data.aws_vpc.default.id
}
```

21. Let's add another set of code after creating the Target group. We will be creating the Application Load Balancer and Listener in the below code.
22. In the code below, we have mentioned the ip_address_type as ipv4. We have specified the load balancer type as an application.
23. The security group ID is automatically taken by using the variable which we have set during the creation process.
24. The subnet_ids of the default VPC are taken from the data variable.
25. We have mentioned the tags for the load balancer.
26. For the Listener, we have provided the load balancer arn which will be taken once the load balancer is created.
27. We have configured the protocol and port as HTTP and 80 respectively and forwarded the request to the created target group.

```
############# Creating Application Load Balancer #############

resource "aws_lb" "application-lb" {
    name            = "whiz-alb"
    internal        = false
    ip_address_type     = "ipv4"
    load_balancer_type = "application"
    security_groups = [aws_security_group.web-server.id]
    subnets = [
                data.aws_subnet.subnet1.id,
                data.aws_subnet.subnet2.id
                ]
    tags = {
        Name = "whiz-alb"
    }
}

######################## Creating Listener ######################

resource "aws_lb_listener" "alb-listener" {
    load_balancer_arn          = aws_lb.application-lb.arn
    port                       = 80
    protocol                   = "HTTP"
    default_action {
        target_group_arn         = aws_lb_target_group.target-group.arn
        type                     = "forward"
    }
}
```

28. Now, we have created the required resources. We will complete the main.tf by attaching the target group to the Application load balancer.
29. In the code below, we have specified the target group arn and the target_id, i.e the id's of the created EC2 Instances.

```
################ Attaching Target group to ALB ################

resource "aws_lb_target_group_attachment" "ec2_attach" {
    count = length(aws_instance.web-server)
    target_group_arn = aws_lb_target_group.target-group.arn
    target_id        = aws_instance.web-server[count.index].id
}
```

30. Save the file by pressing **Ctrl + S**.

Task 5: Create an Output file

In this task, you will create an **output.tf** file where you add details of the output you want to display.

1. To create an **output.tf** file, expand the folder **task_10004_elb** and click on the **New File** icon to add the file.
2. Name the file as **output.tf** and press **Enter** to save it.
3. Paste the below content into the **output.tf** file.

```
output "elb-dns-name" {
  value = aws_lb.application-lb.dns_name
}
```

4. In the above code, we will extract the DNS name of the created Application load balancer and display it once the resources are created.

Task 6: Confirm the installation of Terraform by checking the version

1. In the Visual Studio Code, open Terminal by selecting **View** from the Menu bar and choose **Terminal**.
2. If you are not in the newly created folder change your present working directory by running the below command.

```
cd task_10004_elb
```

3. To confirm the installation of Terraform, run the below command to check the version:

```
terraform version
```

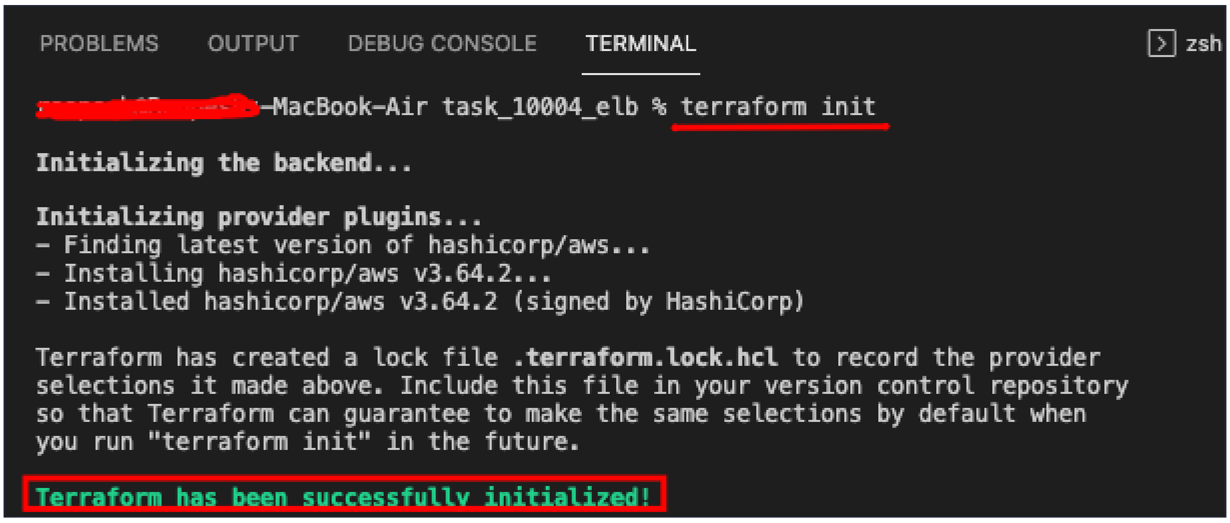
4. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, To install terraform follow the official guide link provided in the Prerequisite section above.

Task 7: Apply terraform configurations

1. Initialize Terraform by running the below command,

```
terraform init
```

2. **Note:** terraform init will check for all the plugin dependencies and download them if required, this will be used for creating a deployment plan.



```
[redacted]-MacBook-Air task_10004_elb % terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.64.2...
- Installed hashicorp/aws v3.64.2 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

3. To generate the action plans run the below command,

```
terraform plan
```

4. Review the whole generated plan.

5. To create all the resources declared in main.tf configuration file, run the below command,

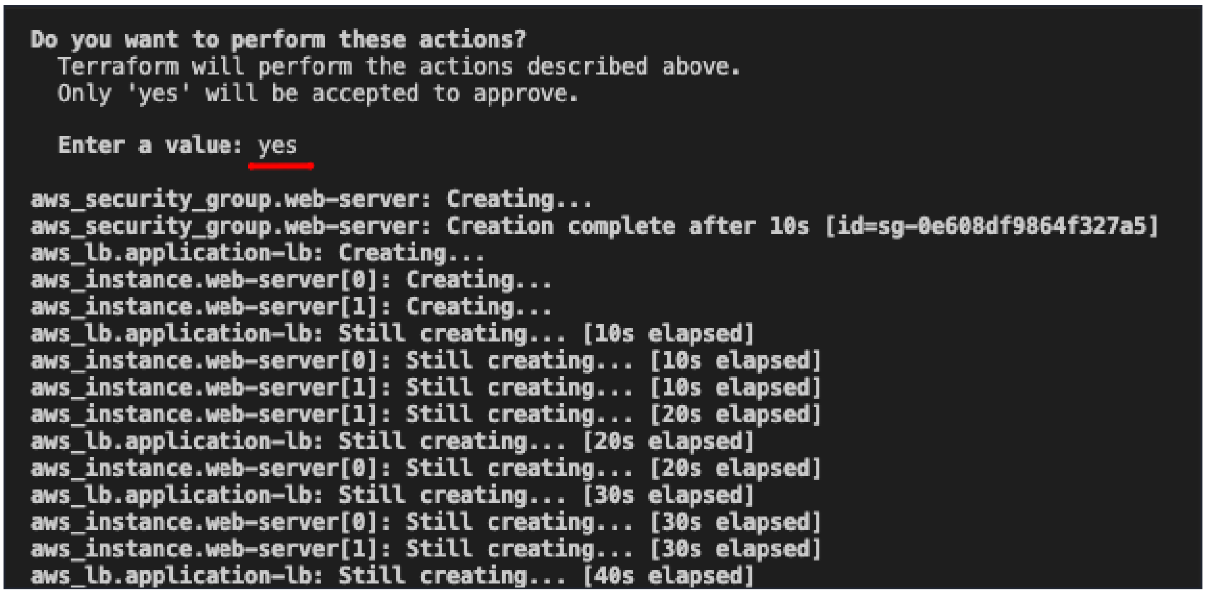
```
terraform apply
```

6. You will be able to see the resources which will be created, approve the creation of all the resources by entering **yes**.

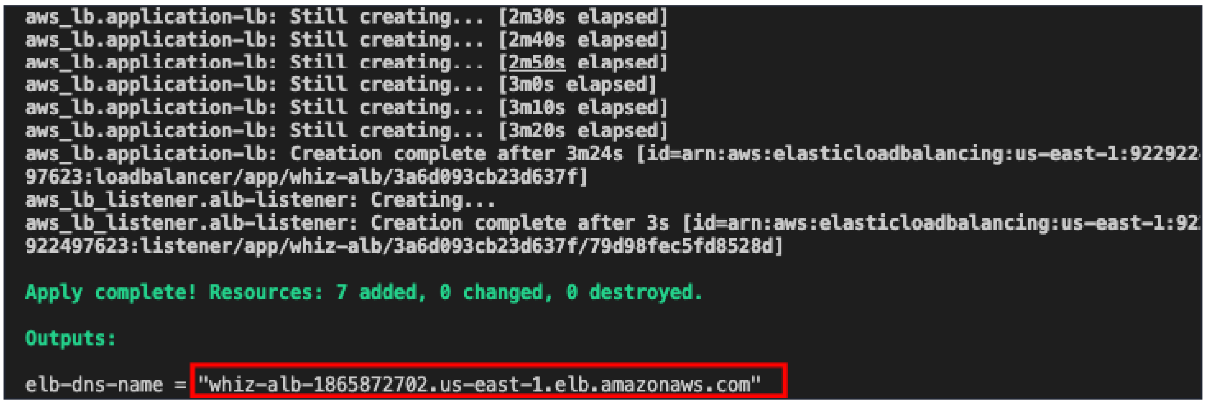
7. It may take up to 3-5 minutes for the terraform apply command to create the resources.

8. Id's of all the resources created by terraform will be visible there.

9. The output i.e DNS name of the application load balancer is extracted and displayed. Copy the DNS name.



```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.web-server: Creating...
aws_security_group.web-server: Creation complete after 10s [id=sg-0e608df9864f327a5]
aws_lb.application-lb: Creating...
aws_instance.web-server[0]: Creating...
aws_instance.web-server[1]: Creating...
aws_lb.application-lb: Still creating... [10s elapsed]
aws_instance.web-server[0]: Still creating... [10s elapsed]
aws_instance.web-server[1]: Still creating... [10s elapsed]
aws_instance.web-server[1]: Still creating... [20s elapsed]
aws_lb.application-lb: Still creating... [20s elapsed]
aws_instance.web-server[0]: Still creating... [20s elapsed]
aws_lb.application-lb: Still creating... [30s elapsed]
aws_instance.web-server[0]: Still creating... [30s elapsed]
aws_instance.web-server[1]: Still creating... [30s elapsed]
aws_lb.application-lb: Still creating... [40s elapsed]
```

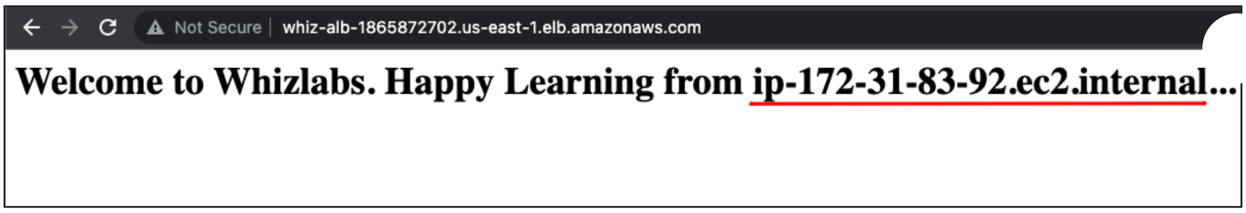


```
aws_lb.application-lb: Still creating... [2m30s elapsed]
aws_lb.application-lb: Still creating... [2m40s elapsed]
aws_lb.application-lb: Still creating... [2m50s elapsed]
aws_lb.application-lb: Still creating... [3m0s elapsed]
aws_lb.application-lb: Still creating... [3m10s elapsed]
aws_lb.application-lb: Still creating... [3m20s elapsed]
aws_lb.application-lb: Creation complete after 3m24s [id=arn:aws:elasticloadbalancing:us-east-1:922922
97623:loadbalancer/app/whiz-alb/3a6d093cb23d637f]
aws_lb_listener.alb-listener: Creating...
aws_lb_listener.alb-listener: Creation complete after 3s [id=arn:aws:elasticloadbalancing:us-east-1:92
922497623:listener/app/whiz-alb/3a6d093cb23d637f/79d98fec5fd8528d]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

elb-dns-name = "whiz-alb-1865872702.us-east-1.elb.amazonaws.com"
```

10. Optionally, you can note down the IDs of all the resources.

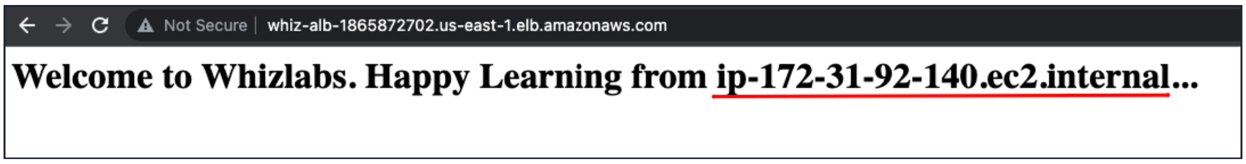
Task 8: Check the HTML page and traffic distribution

1. In the terraform file, we have used user data to create an apache server and publish a HTML page.
2. Open a new tab in the browser and paste the DNS name of the created load balancer.
3. The HTML content created in the user data is displayed in the page.
4. Keep refreshing the page to see the incoming traffic distribution between the two instances.



Not Secure | whiz-alb-1865872702.us-east-1.elb.amazonaws.com

Welcome to Whizlabs. Happy Learning from ip-172-31-83-92.ec2.internal...

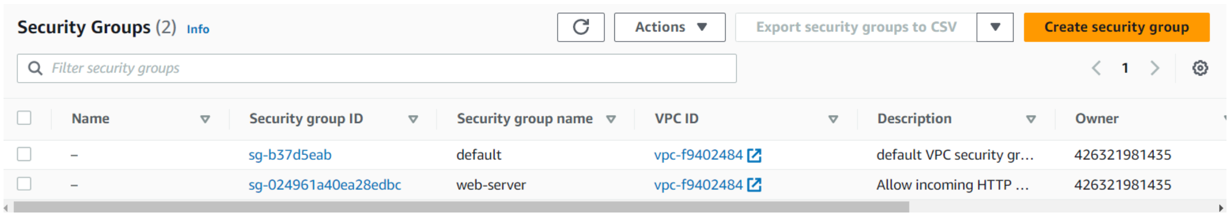

Not Secure | whiz-alb-1865872702.us-east-1.elb.amazonaws.com

Welcome to Whizlabs. Happy Learning from ip-172-31-92-140.ec2.internal...

5. We can now say that the 2 EC2 Instances, an Application load balancer has been created. The incoming traffic is being distributed to two EC2 instances with the help of the load balancer.
6. We can also confirm that the security group is allowing HTTP incoming requests.

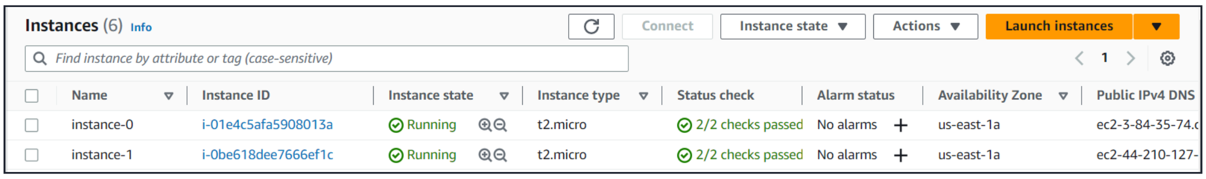
Task 9: Check the resources in AWS Console

1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **EC2** by clicking on **Services** on the top, then click on **EC2** in the **Compute** section.
3. Navigate to **Security Groups** under **Network & Security** on the left panel.
4. You will be able to see the security group with the name **web-server** which we have created in the terraform.


Security Groups (2) Info

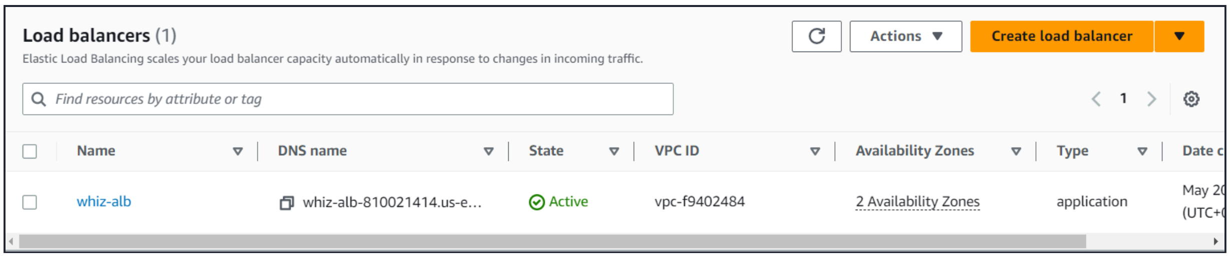
Name	Security group ID	Security group name	VPC ID	Description	Owner
–	sg-b37d5eab	default	vpc-f9402484	default VPC security gr...	426321981435
–	sg-024961a40ea28edbc	web-server	vpc-f9402484	Allow incoming HTTP ...	426321981435

5. Navigate to **Instances** under **Instances** on the left panel**.**
6. You can see the instances created. You can check the configurations that we applied in the terraform file like key pair, security group, instance type, etc.


Instances (6) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
instance-0	i-01e4c5afa5908013a	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-3-84-35-74.c
instance-1	i-0be618dee7666ef1c	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-44-210-127-

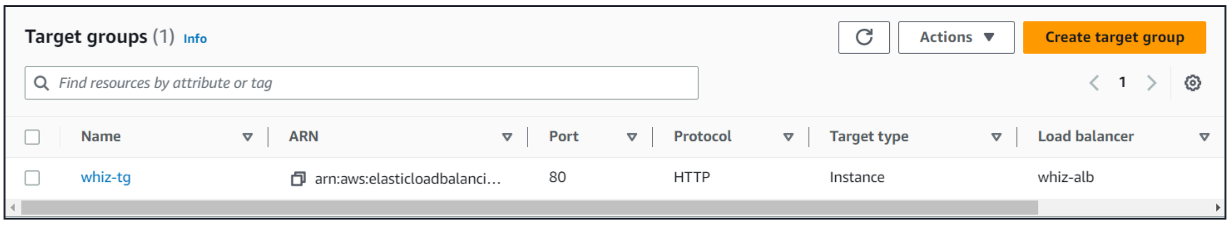
7. Navigate to **Load Balancers** under **Load Balancing** on the left panel**.**
8. You will be able to see the created load balancer with the name **whiz-alb** which we have created in the terraform.


Load balancers (1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date c
whiz-alb	whiz-alb-810021414.us-e...	Active	vpc-f9402484	2 Availability Zones	application	May 20 (UTC+(

9. Navigate to **Target Groups** under **Load Balancing** on the left panel.

10. You will be able to see the created target group with the name **whiz-tg** which we have created in the terraform.


Target groups (1) Info

Name	ARN	Port	Protocol	Target type	Load balancer
whiz-tg	arn:aws:elasticloadbalanci...	80	HTTP	Instance	whiz-alb

Do You Know?

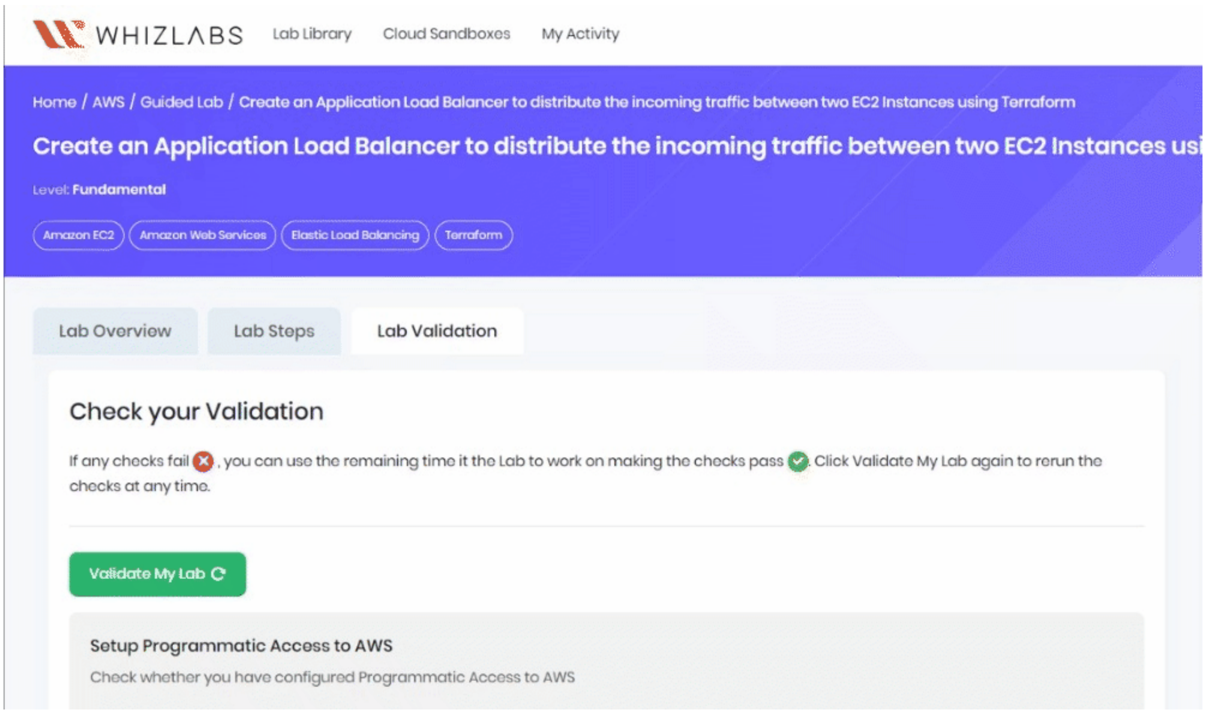
ALBs operate at the application layer (Layer 7) of the OSI model, allowing them to make intelligent routing decisions based on content, such as HTTP headers, URLs, or cookies. This enables more sophisticated routing scenarios and enhances the flexibility of your application deployments.

Task 10: Validation of the lab

1. Once the lab steps are completed, please click on the **Validation** button on the left side panel.

2. This will validate the resources in the AWS account and displays whether you have completed this lab successfully or not.

3. Sample output :



Create an Application Load Balancer to distribute the incoming traffic between two EC2 Instances usi

Level: Fundamental

Check your Validation

If any checks fail ❌, you can use the remaining time it the Lab to work on making the checks pass ✅. Click Validate My Lab again to rerun the checks at any time.

Setup Programmatic Access to AWS

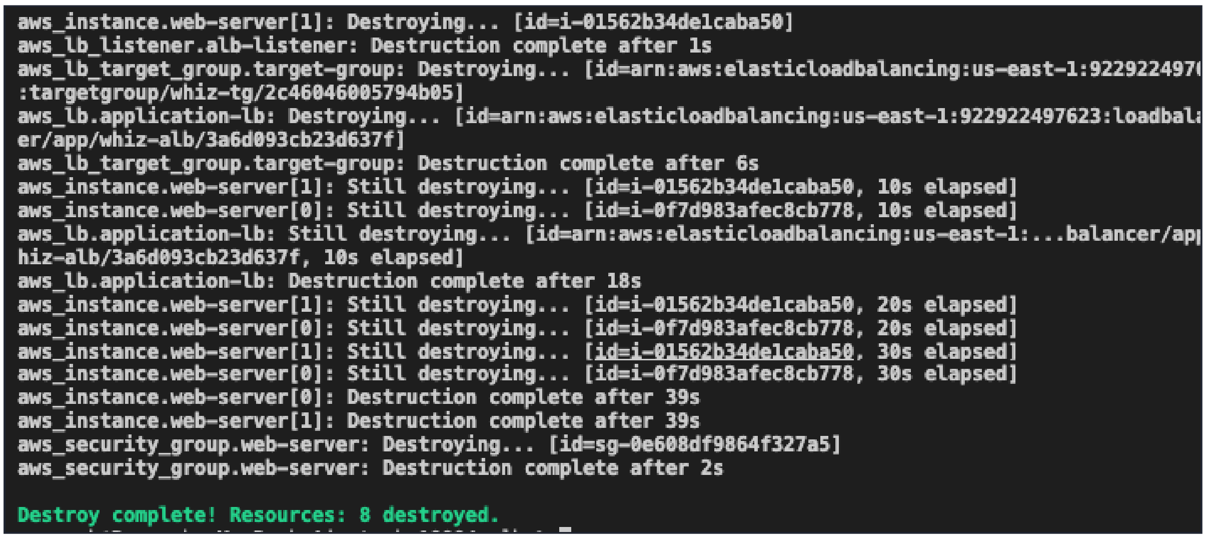
Check whether you have configured Programmatic Access to AWS

Task 11: Delete AWS Resources

1. To delete the resources, open Terminal again.
2. Run the below command to delete all the resources.

```
terraform destroy
```

3. Enter yes to confirm the deletion.



```
aws_instance.web-server[1]: Destroying... [id=i-01562b34de1caba50]
aws_lb_listener.alb-listener: Destruction complete after 1s
aws_lb_target_group.target-group: Destroying... [id=arn:aws:elasticloadbalancing:us-east-1:9229224976
:targetgroup/whiz-tg/2c46046005794b05]
aws_lb.application-lb: Destroying... [id=arn:aws:elasticloadbalancing:us-east-1:922922497623:loadbala
er/app/whiz-alb/3a6d093cb23d637f]
aws_lb_target_group.target-group: Destruction complete after 6s
aws_instance.web-server[1]: Still destroying... [id=i-01562b34de1caba50, 10s elapsed]
aws_instance.web-server[0]: Still destroying... [id=i-0f7d983afec8cb778, 10s elapsed]
aws_lb.application-lb: Still destroying... [id=arn:aws:elasticloadbalancing:us-east-1:...balancer/ap
hiz-alb/3a6d093cb23d637f, 10s elapsed]
aws_lb.application-lb: Destruction complete after 18s
aws_instance.web-server[1]: Still destroying... [id=i-01562b34de1caba50, 20s elapsed]
aws_instance.web-server[0]: Still destroying... [id=i-0f7d983afec8cb778, 20s elapsed]
aws_instance.web-server[1]: Still destroying... [id=i-01562b34de1caba50, 30s elapsed]
aws_instance.web-server[0]: Still destroying... [id=i-0f7d983afec8cb778, 30s elapsed]
aws_instance.web-server[0]: Destruction complete after 39s
aws_instance.web-server[1]: Destruction complete after 39s
aws_security_group.web-server: Destroying... [id=sg-0e608df9864f327a5]
aws_security_group.web-server: Destruction complete after 2s

Destroy complete! Resources: 8 destroyed.
```

Completion and Conclusion

- You have set up the Visual Studio Code editor.

- You have created variables.tf and terraform.tfvars files.
- You have created a main.tf file.
- You have executed the terraform configuration commands to create the resources.
- You have checked all the resources created by opening the Console.
- You have deleted all the resources.

End Lab

1. Sign out of AWS Account.
2. You have successfully completed the lab.
3. Once you have completed the steps, click on **End Lab** from your whizlabs dashboard.