



---

# TERRAFORM PROJECT

---

Yahya Abd El-Azim



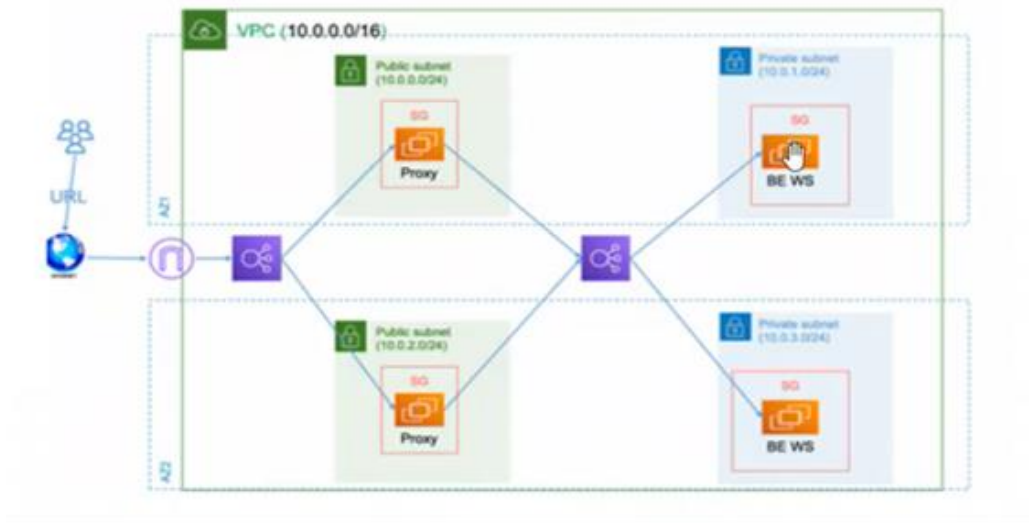
ENG: AYA NABIL

## Project Overview

This Terraform project is designed to create an infrastructure with multiple EC2 instances and two load balancers, as shown in the provided diagram. The infrastructure consists of:

1. A Virtual Private Cloud (VPC) with public and private subnets.
2. EC2 instances configured as web servers in the private subnets.
3. A proxy server in the public subnets that routes traffic to the private web servers.
4. Two load balancers:
  - The first one is a public load balancer that forwards traffic to the proxy server.
  - The second one is a private load balancer that forwards traffic to the backend EC2 web servers.

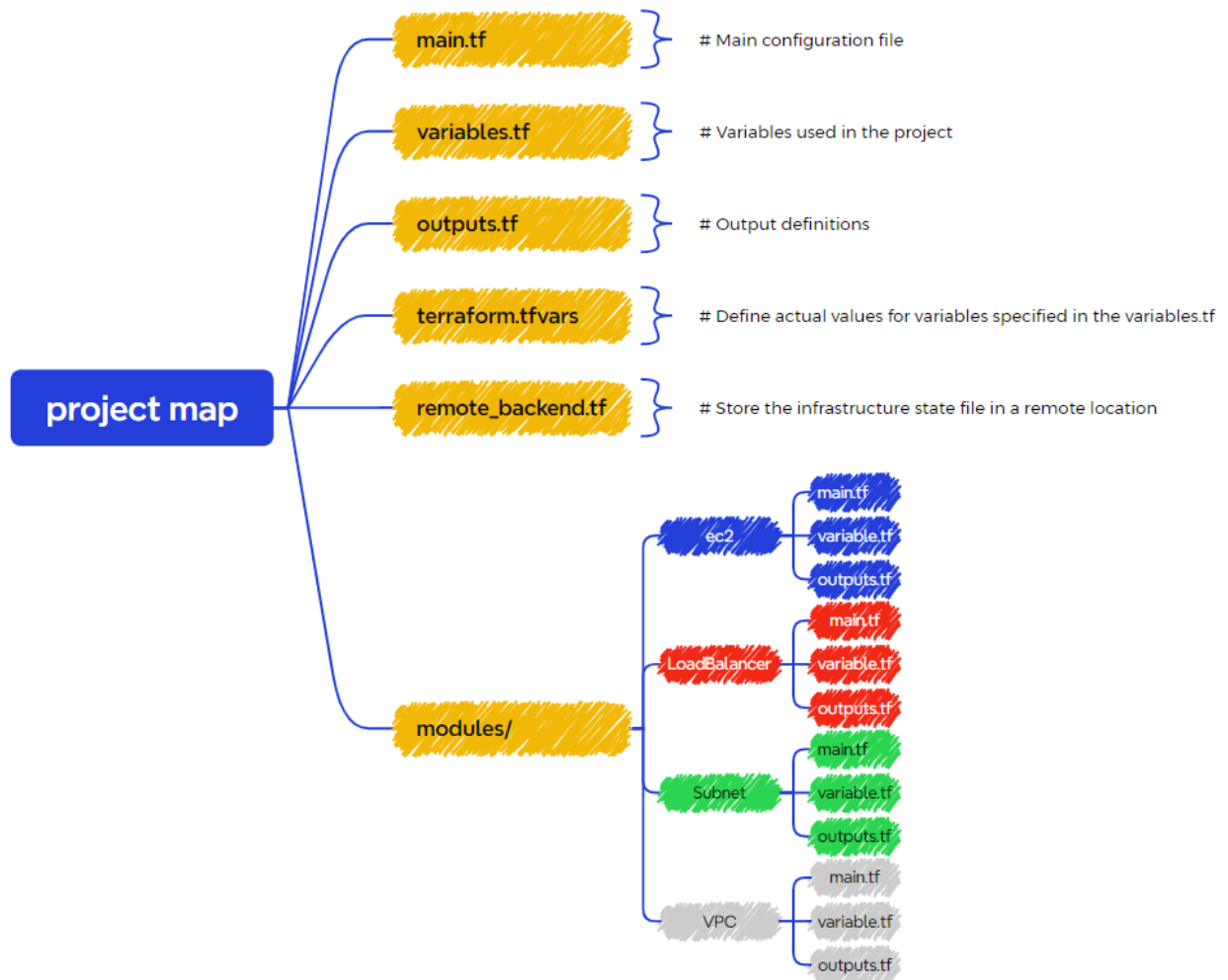
The project uses AWS services, including EC2, Elastic Load Balancers (ELB), and S3 for storing Terraform state files.



## Prerequisites

1. **Terraform** installed on your local machine.
2. **AWS CLI** configured with appropriate credentials.
3. An existing **S3 bucket** and **DynamoDB table** for remote state management.
4. **SSH key pair** for EC2 access.

# Project Structure



## Implementation Details

1. **Create a New Workspace:** Start by creating a new Terraform workspace called `dev`:

```
[yahya@localhost projectTerra]$ terraform workspace new dev
Created and switched to workspace "dev"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
[yahya@localhost projectTerra]$ terraform workspace select dev
[yahya@localhost projectTerra]$
```

## 2. Remote State Configuration: Configure the backend to store the state file remotely using S3:

```
1  remote_backend.tf > resource "aws_dynamodb_table" "terraform_locks" > tags
2  data "aws_s3_bucket" "existing_bucket" {
3  | bucket = "mm-remote-statefile"
4  | }
5
6  resource "aws_s3_bucket" "terraform_state" {
7  | count = length(data.aws_s3_bucket.existing_bucket.id) == 0 ? 1 : 0
8  | bucket = "mm-remote-statefile"
9  | force_destroy = true
10 |
11 | lifecycle {
12 | | prevent_destroy = true
13 | | }
14 |
15 | tags = {
16 | | Name = "Terraform State Bucket"
17 | | }
18 | }
19
20 resource "aws_s3_bucket_versioning" "enable" {
21 | count = length(data.aws_s3_bucket.existing_bucket.id) == 0 ? 1 : 0
22 | bucket = aws_s3_bucket.terraform_state[0].id
23 | versioning_configuration {
24 | | status = "Enabled"
25 | | }
26 | }
27
28 resource "aws_dynamodb_table" "terraform_locks" {
29 | name = "NM-locks"
30 | billing_mode = "PAY_PER_REQUEST"
31 | hash_key = "LockID"
32 |
33 | attribute {
34 | | name = "LockID"
35 | | type = "S"
36 | | }
37 |
38 | lifecycle {
39 | | ignore_changes = [name]
40 | | }
41 |
42 | tags = {}
43 | | Name = "Terraform Lock Table"
44 | | }
45 | }
46
47 terraform {
48 | backend "s3" {
49 | | bucket = "mm-remote-statefile"
50 | | key = "terraform.tfstate"
51 | | region = "us-east-1"
52 | | dynamodb_table = "NM-locks"
53 | | encrypt = true
54 | | }
55 | }
56
```

General purpose buckets		Directory buckets		
General purpose buckets (1) Info All AWS Regions		Copy ARN	Empty	Delete Create bucket
Buckets are containers for data stored in S3.				
Find buckets by name				
Name	AWS Region	IAM Access Analyzer		Creation date
<a href="#">mm-remote-statefile</a>	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>		September 29, 2024, 23:51:43 (UTC+03:00)

3. **VPC:** Create a VPC using a custom VPC module.
  - o creating a VPC module in `modules/VPC/main.tf`:

```

1 # CREATE A VPC
2 resource "aws_vpc" "vpc" {
3   cidr_block = var.vpc_cidr
4   tags = {
5     Name = "N-VPC"
6   }
7 }
8
9 # CREATE AN INTERNET GATEWAY
10 resource "aws_internet_gateway" "igw" {
11   vpc_id = aws_vpc.vpc.id
12   tags = {
13     Name = "N-IGW"
14   }
15 }
16
17 # CREATE ELASTIC IP
18 resource "aws_eip" "eip" {
19   domain = "vpc"
20 }
21
22 # CREATE NAT IN THE FIRST PUBLIC SUBNET
23 resource "aws_nat_gateway" "nat" {
24   allocation_id = aws_eip.eip.id
25   subnet_id = var.nat_subnet_id
26   tags = {
27     Name = "N-NAT"
28   }
29   # To ensure proper ordering, i will add an explicit dependency on the Internet Gateway.
30   depends_on = [aws_internet_gateway.igw]
31 }

```

main.tf | outputs.tf

```

modules > VPC > outputs.tf > output "igw_id" > value
1 # OUTPUT TO RETRIVE VPC ID
2 output "vpc_id" {
3   value = aws_vpc.vpc.id
4 }
5
6 # OUTPUT TO RETRIVE INTERNET GATEWAY ID
7 output "igw_id" {
8   value = aws_internet_gateway.igw.id
9 }
10
11 #OUTPUT TO RETRIVE NAT ID
12 output "nat_id" {
13   value = aws_nat_gateway.nat.id
14 }

```

main.tf | outputs.tf | variables.tf

```

modules > VPC > variables.tf > ...
1 # VPC VARIALES
2 variable "vpc_cidr" {
3   description = "VPC CIDR"
4   type = string
5 }
6 # NAT VARIABLES
7 variable "nat_subnet_id" {
8   description = "The subnet ID of the public subnet in which to place the gateway"
9   type = string
10 }

```

Your VPCs (1/2) Info

Last updated 11 minutes ago

Search

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP op
-	vpc-04b593bdf95824032	Available	172.31.0.0/16	-	dopt-05e
N-VPC	vpc-03267210d0c64f3fe	Available	10.0.0.0/16	-	dopt-05e

vpc-03267210d0c64f3fe / N-VPC

Details | Resource map | CIDRs | Flow logs | Tags | Integrations

**Details**

VPC ID vpc-03267210d0c64f3fe	State Available	DNS hostnames Disabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-05ebf2bb089936ff9	Main route table rtb-03a53e8c389aca399	Main network ACL acl-016b38bda3e893905
Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -	IPv6 CIDR (Network border group) -
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups	Owner ID 042440520971	

#### 4. Subnets: create public and private subnets using a custom **Subnet** module

- Define subnets, route tables, and gateways as needed in `modules/Subnet/main.tf`

```
1 resource "aws_subnet" "public_subnets" {
2   count           = length(var.pub_subnets)
3   vpc_id          = var.vpc_id
4   cidr_block      = var.pub_subnets[count.index].subnets_cidr
5   availability_zone = var.pub_subnets[count.index].availability_zone
6   tags = {
7     Name = "sprints_public_subnet_${count.index}"
8   }
9 }
10
11 resource "aws_route_table" "public-rt" {
12   vpc_id = var.vpc_id
13   route {
14     cidr_block = "0.0.0.0/0"
15     gateway_id = var.igw_id
16   }
17 }
18
19 resource "aws_route_table_association" "public-rta" {
20   count           = length(aws_subnet.public_subnets)
21   subnet_id       = aws_subnet.public_subnets[count.index].id
22   route_table_id = aws_route_table.public-rt.id
23 }
24
25 resource "aws_subnet" "private_subnets" {
26   count           = length(var.priv_subnets)
27   vpc_id          = var.vpc_id
28   cidr_block      = var.priv_subnets[count.index].subnets_cidr
29   availability_zone = var.priv_subnets[count.index].availability_zone
30   tags = {
31     Name = "sprints_private_subnet_${count.index}"
32   }
33 }
34
35 resource "aws_route_table" "private-rt" {
36   vpc_id = var.vpc_id
37   route {
38     cidr_block      = "0.0.0.0/0"
39     nat_gateway_id = var.nat_id
40   }
41 }
42
43 resource "aws_route_table_association" "private-rta" {
44   count           = length(aws_subnet.private_subnets)
45   subnet_id       = aws_subnet.private_subnets[count.index].id
46   route_table_id = aws_route_table.private-rt.id
47 }
```

```
main.tf  outputs.tf X
modules > Subnet > outputs.tf > output "private_subnets_id"
1 output "public_subnets_id" {
2   value = aws_subnet.public_subnets[*].id
3 }
4
5 output "private_subnets_id" {
6   value = aws_subnet.private_subnets[*].id
7 }
```

```
modules > Subnet > variables.tf > variable "vpc_id"
1 variable "vpc_id" {
2   description = "ID of the vpc in where the subnets will be"
3   type       = string
4 }
5
6 variable "pub_subnets" {
7   description = "public subnets info"
8   type       = list(object({
9     subnets_cidr = string
10    availability_zone = string
11  }))
12 }
13
14 variable "igw_id" {
15   description = "ID of the IGW to be used in route table for public subnets"
16   type       = string
17 }
18
19 variable "priv_subnets" {
20   description = "private subnets info"
21   type       = list(object({
22     subnets_cidr = string
23     availability_zone = string
24   }))
25 }
26
27 variable "nat_id" {
28   description = "ID of the NAT to be used in route table for private subnets"
29   type       = string
30 }
```

Launch AWS Academy Learner | subnets | VPC Console | 100.24.120.192 | 3.86.89.112

https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#subnets:

Services Search [Alt+S] N. Virginia voclabs/user3431266-Yahya\_Abd\_El-Azim\_Shaker\_ @ 0424-4052-0971

**VPC dashboard** x

EC2 Global View

Filter by VPC

Virtual private cloud

Your VPCs

Subnets

Route tables

Internet gateways

Egress-only internet gateways

Carrier gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

NAT gateways

Peering connections

**Subnets (1/14) info**

Find resources by attribute or tag

Last updated less than a minute ago

Actions Create subnet

Name	Subnet ID	State	VPC	IPv4 CIDR
<input checked="" type="checkbox"/> sprints_public_subnet_0	subnet-0ef39188b7aed5209	Available	vpc-061ce60ed965663c4   N-VPC	10.0.0.0/24
<input type="checkbox"/> sprints_private_subnet_1	subnet-078315a12fe7b8840	Available	vpc-03267210d0c64f3fe   N-VPC	10.0.3.0/24

**subnet-0ef39188b7aed5209 / sprints\_public\_subnet\_0**

Details Flow logs Route table Network ACL CIDR reservations Sharing Tags

**Details**

Subnet ID subnet-0ef39188b7aed5209	Subnet ARN arn:aws:ec2:us-east-1:042440520971:subnet/subnet-0ef39188b7aed5209	State Available	IPv4 CIDR 10.0.0.0/24
Available IPv4 addresses 248	IPv6 CIDR -	IPv6 CIDR association ID -	Availability Zone us-east-1a
Availability Zone ID use1-az1	Network border group us-east-1	VPC vpc-061ce60ed965663c4   N-VPC	Route table rtb-069df997efc31aeef
Network ACL acl-0a01c75485b23168d	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	

Launch AWS Academy Learner | subnets | VPC Console | 100.24.120.192 | 3.86.89.112

https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#subnets:

Services Search [Alt+S] N. Virginia voclabs/user3431266-Yahya\_Abd\_El-Azim\_Shaker\_ @ 0424-4052-0971

**VPC dashboard** x

EC2 Global View

Filter by VPC

Virtual private cloud

Your VPCs

Subnets

Route tables

Internet gateways

Egress-only internet gateways

Carrier gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

NAT gateways

Peering connections

**Subnets (1/14) info**

Find resources by attribute or tag

Last updated 1 minute ago

Actions Create subnet

Name	Subnet ID	State	VPC	IPv4 CIDR
<input type="checkbox"/> sprints_public_subnet_0	subnet-0ef39188b7aed5209	Available	vpc-061ce60ed965663c4   N-VPC	10.0.0.0/24
<input checked="" type="checkbox"/> sprints_private_subnet_1	subnet-078315a12fe7b8840	Available	vpc-03267210d0c64f3fe   N-VPC	10.0.3.0/24

**subnet-078315a12fe7b8840 / sprints\_private\_subnet\_1**

Details Flow logs Route table Network ACL CIDR reservations Sharing Tags

**Details**

Subnet ID subnet-078315a12fe7b8840	Subnet ARN arn:aws:ec2:us-east-1:042440520971:subnet/subnet-078315a12fe7b8840	State Available	IPv4 CIDR 10.0.3.0/24
Available IPv4 addresses 250	IPv6 CIDR -	IPv6 CIDR association ID -	Availability Zone us-east-1b
Availability Zone ID use1-az2	Network border group us-east-1	VPC vpc-03267210d0c64f3fe   N-VPC	Route table rtb-0f97c44f48fd80d3b
Network ACL acl-016b38bda3e893905	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	
Auto-assign customer-owned IPv4 address	Default subnet No	Outpost ID -	IPv4 CIDR reservations -



## 5. EC2 Instances:

- Create EC2 instances using a custom EC2 module.
- Use a data source to get the AMI ID for EC2:

```
1 data "aws_ami" "ami_id" {
2   most_recent = true
3   owners      = ["amazon"]
4   filter {
5     name     = "name"
6     values   = ["amzn2-ami-hvm-*-x86_64-gp2"]
7   }
8 }
```

- Create the security group

```
10 resource "aws_security_group" "sg" {
11   vpc_id = var.sg_vpc_id
12
13   ingress {
14     from_port = 443
15     to_port   = 443
16     protocol  = "tcp"
17     cidr_blocks = ["0.0.0.0/0"]
18   }
19
20   ingress {
21     from_port = 80
22     to_port   = 80
23     protocol  = "tcp"
24     cidr_blocks = ["0.0.0.0/0"]
25   }
26
27   ingress {
28     from_port = 22
29     to_port   = 22
30     protocol  = "tcp"
31     cidr_blocks = ["0.0.0.0/0"]
32   }
33
34   egress {
35     from_port = 0
36     to_port   = 0
37     protocol  = "-1"
38     cidr_blocks = ["0.0.0.0/0"]
39   }
40
41   tags = {
42     Name = "sg"
43   }
44 }
45
```

- Create public instance

```
46 resource "aws_instance" "pub-ec2" {
47   count           = length(var.ec2_public_subnet_id)
48   ami             = data.aws_ami.ami_id.id
49   instance_type   = "t2.micro"
50   subnet_id       = var.ec2_public_subnet_id[count.index]
51   security_groups = [aws_security_group.sg.id]
52   associate_public_ip_address = true
53   key_name         = var.key_pair_name
54
55   tags = {
56     Name = "public_ec2_${count.index}"
57   }
58 }
```

- Use remote-exec provisioners to install Apache server and configure the web servers:

```

provisioner "remote-exec" {
  inline = [
    "set -e",
    "sleep 10",
    "sudo yum update -y",
    "sudo yum install -y httpd",
    "sudo systemctl start httpd",
    "sudo systemctl enable httpd",
    | <<-EOT
    echo '<html><body><h1>Welcome to Public yahya EC2 Instance ${count.index}</h1>
    </body></html>' | sudo tee /var/www/html/index.html
    EOT
    ]
  connection {
    type      = "ssh"
    host      = self.public_ip
    user      = "ec2-user"
    private_key = file("~/Downloads/kk.pem")
    timeout   = "5m"
  }
}

```

- Use local-exec

```

provisioner "local-exec" {
  when      = create
  on_failure = continue
  command   = "echo public-ip-${count.index} : ${self.public_ip} >> all-ips.txt"
}

```

- Create private instance

```

resource "aws_instance" "priv-ec2" {
  count          = length(var.ec2_private_subnet_id)
  ami            = data.aws_ami.ami_id.id
  instance_type  = "t2.micro"
  subnet_id     = var.ec2_private_subnet_id[count.index]
  security_groups = [aws_security_group.sg.id]
  associate_public_ip_address = false
  lifecycle {
    | | create_before_destroy = true
  }
  tags = {
    | Name = "private_ec2_${count.index}"
  }

  # User data to set up Apache and configure index.html
  user_data = <<EOF
  | | | | | #!/bin/bash
  | | | | | sleep 10
  | | | | | sudo yum update -y
  | | | | | sleep 10
  | | | | | sudo yum install -y httpd
  | | | | | sleep 10
  | | | | | sudo systemctl start httpd
  | | | | | sudo systemctl enable httpd
  | | | | | echo "<html><body><h1>${var.ec2_html[count.index]}</h1>
  | | | | | <p>welcome to Priv${count.index}</p>
  | | | | | </body></html>" | sudo tee /var/www/html/index.html
  | | | | | sudo systemctl restart httpd
  | | | | | EOF

  # Local provisioner to log the private IPs of created instancesyes
  provisioner "local-exec" {
    when      = create
    on_failure = continue
    command   = "echo private-ip-${count.index} : ${self.private_ip} >> all-ips.txt"
  }
}

```

```
main.tf | outputs.tf | variables.tf
modules > ec2 > outputs.tf > output "pub-ips" > value
1 output "public_ec2_id" {
2   | value = aws_instance.pub-ec2[*].id
3 }
4
5 output "private_ec2_id" {
6   | value = aws_instance.priv-ec2[*].id
7 }
8
9 output "security_group_id" {
10  | value = aws_security_group.sg.id
11 }
12
13 output "pub-ips" {
14   | value = aws_instance.pub-ec2[*].public_ip
15 }
16 }

main.tf | outputs.tf | variables.tf
modules > ec2 > variables.tf > variable "key_pair_name" > default
16 variable "ec2_html" {
17   default = [
18     | "Welcome to Private EC2 Instance 2"
19   ]
20 }
21
22 variable "key_pair_name" {
23   description = "Name of the EC2 Key Pair"
24   type        = string
25   default     = "key"
26 }
27 }
```

The final result:

[Alt+S]

N. Virginia

voclabs/user3431266=Yahya\_Abd\_El-Azim\_Shaker\_@ 0424-4052-0971

Instances (4) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

Instance state = running

Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>	public_ec2_1	i-0f32a48ec1fdf7938	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-
<input type="checkbox"/>	private_ec2_1	i-05a70fa7fa9ad9837	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-
<input type="checkbox"/>	public_ec2_0	i-0034c0d5dab0604ac	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-
<input type="checkbox"/>	private_ec2_0	i-0490cffde03fc1706	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-

Select an instance

## 6. Load Balancers:

- Create a public and private load balancer using a custom module.
- Configure the target groups and listeners to forward traffic to the EC2 instances.

```
modules > LB > main.tf > ...
1 resource "aws_lb_target_group" "tg" {
2   count      = 2
3   port      = 80
4   protocol  = "HTTP"
5   vpc_id    = var.lb_vpc_id
6 }
7
8 resource "aws_lb_target_group_attachment" "public-target-group-attachment" {
9   count      = length(var.pub_target_id)
10  target_group_arn = aws_lb_target_group.tg[0].arn
11  target_id      = var.pub_target_id[count.index]
12  port          = 80
13 }
14
15 resource "aws_lb_target_group_attachment" "private-target-group-attachment" {
16   count      = length(var.priv_target_id)
17  target_group_arn = aws_lb_target_group.tg[1].arn
18  target_id      = var.priv_target_id[count.index]
19  port          = 80
20 }
21
22 resource "aws_lb" "load-balancer" {
23   count      = 2
24   name      = "my-load-balancer-${count.index}"
25   internal   = var.lb_internal[count.index]
26   load_balancer_type = "application"
27   subnets  = var.lb_subnets[count.index]
28   security_groups = [var.lb_sg_id]
29 }
30
31 resource "aws_lb_listener" "lb-listner" {
32   count      = 2
33   load_balancer_arn = aws_lb.load-balancer[count.index].id
34   port        = "80"
35   protocol    = "HTTP"
36   default_action {
37     type = "forward"
38     target_group_arn = aws_lb_target_group.tg[count.index].id
39   }
40 }
41
42
43
```

```
modules > LB > outputs.tf > output "private_load_balancer_dns"
1 output "public_load_balancer_dns" {
2   value = aws_lb.load-balancer[0].dns_name
3 }
4
5 output "private_load_balancer_dns" {
6   value = aws_lb.load-balancer[1].dns_name
7 }
8
9
10
11
12
13
14
15
16
17
```

```
1 variable "lb_vpc_id" {
2   type = string
3 }
4 variable "pub_target_id" {
5   type = list(string)
6 }
7 variable "priv_target_id" {
8   type = list(string)
9 }
10 variable "lb_internal" {
11   type = list(bool)
12 }
13 variable "lb_subnets" {
14   type = list(list(string))
15 }
16 variable "lb_sg_id" {
17
```

EC2 > Load balancers

Load balancers (2) Actions Create load balancer

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

Filter load balancers

	Name	DNS name	State	VPC ID	Availability Zones	Type
<input type="checkbox"/>	<a href="#">my-load-balancer-1</a>	internal-my-load-balancer...	Active	vpc-03267210d0c64f3fe	<a href="#">2 Availability Zones</a>	application
<input type="checkbox"/>	<a href="#">my-load-balancer-0</a>	my-load-balancer-0-1574...	Active	vpc-03267210d0c64f3fe	<a href="#">2 Availability Zones</a>	application

## 7. Output Values:

- Define output values for the public IP addresses and DNS names of the load balancers.
- Examples
  - **Public Load Balancer DNS:** The DNS name of the public load balancer.
  - **Private Load Balancer DNS:** The DNS name of the private load balancer.
  - **EC2 Public IPs:** The public IP addresses of EC2 instances created in the public subnets.

```
1  output "vpc_id" {
2      | description = "The ID of the VPC: "
3      | value       = module.VPC.vpc_id
4      | }
5  output "igw_id" {
6      | description = "The ID of the Internet Gateway: "
7      | value       = module.VPC.igw_id
8      | }
9  output "nat_id" {
10     | description = "The ID of the NAT Gateway: "
11     | value       = module.VPC.nat_id
12     | }
13 output "public_load_balancer_dns" {
14     | description = "Public Load Balancer DNS name: "
15     | value       = module.LB.public_load_balancer_dns
16     | }
17 output "private_load_balancer_dns" {
18     | description = "Private Load Balancer DNS name: "
19     | value       = module.LB.private_load_balancer_dns
20     | }
21 output "public_ec2_ips" {
22     | description = "Public IPs of the EC2 instances: "
23     | value       = module.ec2.pub-ips
24     | }
25 output "private_subnet_ids" {
26     | description = "Private subnet IDs: "
27     | value       = module.Subnet.private_subnets_id
28     | }
29 output "current_workspace" {
30     | description = "The current Terraform workspace being used."
31     | value       = terraform.workspace
32     | }
33 }
```

## The final Result:

The screenshot shows the Terraform CLI interface with the following components:

- Explorer:** A sidebar on the left showing the project structure, including files like `main.tf`, `outputs.tf`, `remote_backend.tf`, `terraform.tfvars`, and `variables.tf`.
- Terminal:** The main area displays the output of a Terraform plan. It shows the creation of various resources, including VPC, Internet Gateway, NAT Gateway, Load Balancers, EC2 instances, and Subnets. The final output section shows the values for the outputs defined in the configuration file.
- Output:** The final output section shows the values for the outputs defined in the configuration file, including VPC ID, Internet Gateway ID, NAT ID, Public Load Balancer DNS, Private Load Balancer DNS, Public EC2 IPs, Private Subnet IDs, and the current workspace.

The final output section shows the following values:

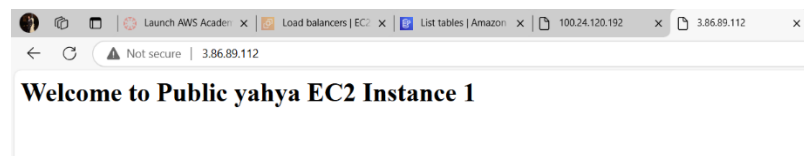
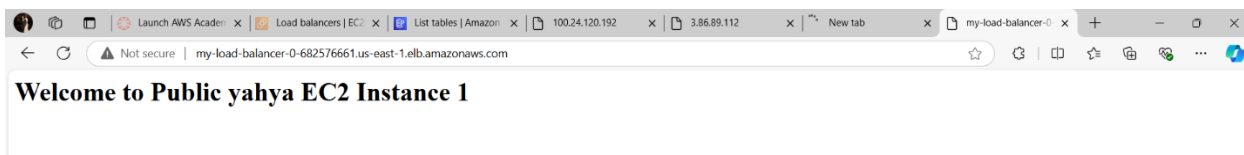
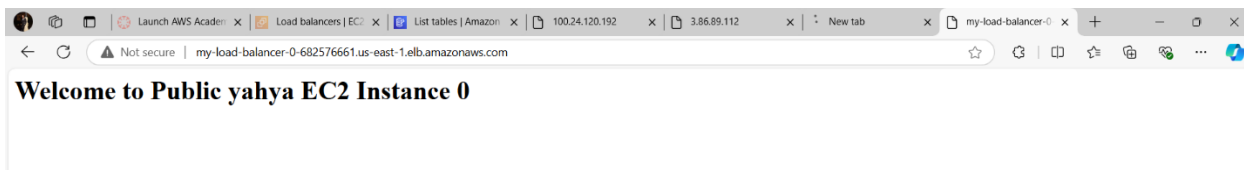
```
igw_id = "igw-02d9254f11345700d"
nat_id = "nat-03b14143e06e1f051"
private_load_balancer_dns = "internal-my-load-balancer-1-1454934893.us-east-1.elb.amazonaws.com"
private_subnet_ids = [
  "subnet-dc350fd6b3100a9fc",
  "subnet-078315a12fe7b8840",
]
public_ec2_ips = [
  "54.173.219.117",
  "54.173.152.54",
]
public_lb_ip = "my-load-balancer-0-1574641021.us-east-1.elb.amazonaws.com"
public_load_balancer_dns = "my-load-balancer-0-1574641021.us-east-1.elb.amazonaws.com"
vpc_id = "vpc-02e72710dc643fe"
```

The all-ips.txt file

```
./ outputs.tf ./ all-ips.txt x main.tf .../LB
all-ips.txt
1 private-ip-1 : 10.0.3.170
2 private-ip-0 : 10.0.1.73
3 public-ip-0 : 100.24.120.192
4 public-ip-1 : 3.86.89.112
5
```

## 8. Testing:

- Access the public DNS of the load balancer and verify it forwards traffic to the proxy.



## Conclusion

This documentation outlines the steps to implement a complex AWS infrastructure using Terraform. The project is structured to separate concerns into modules, making it easier to manage and scale