



Bansilal Ramnath Agarwal Charitable Trust's  
Vishwakarma Institute of Information  
Technology  
Department of  
Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar

Class: TY

Division: B

Roll No: 372028

Semester: V

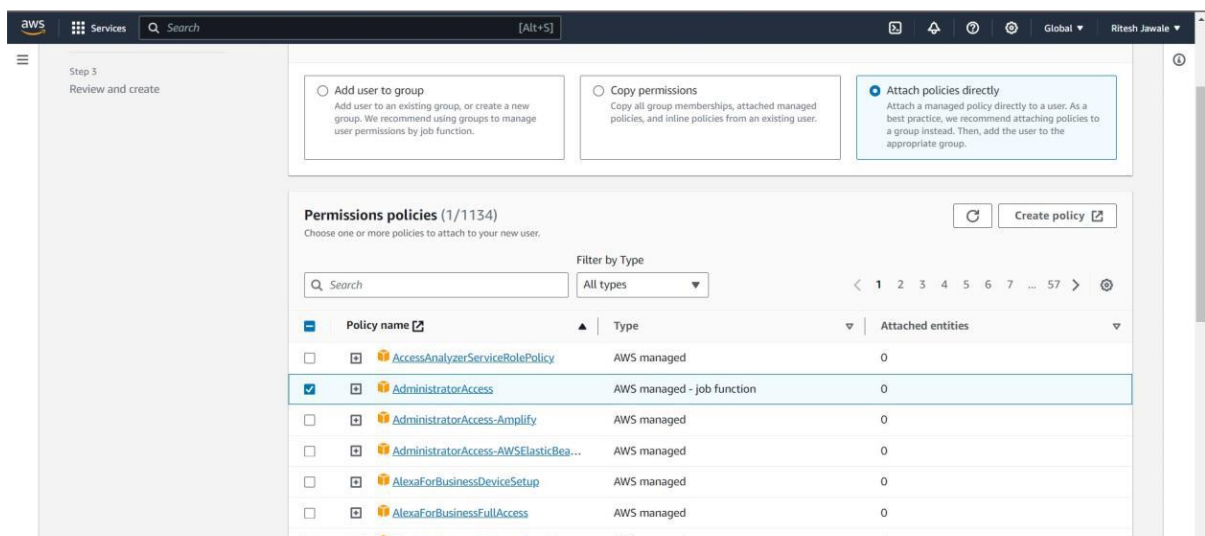
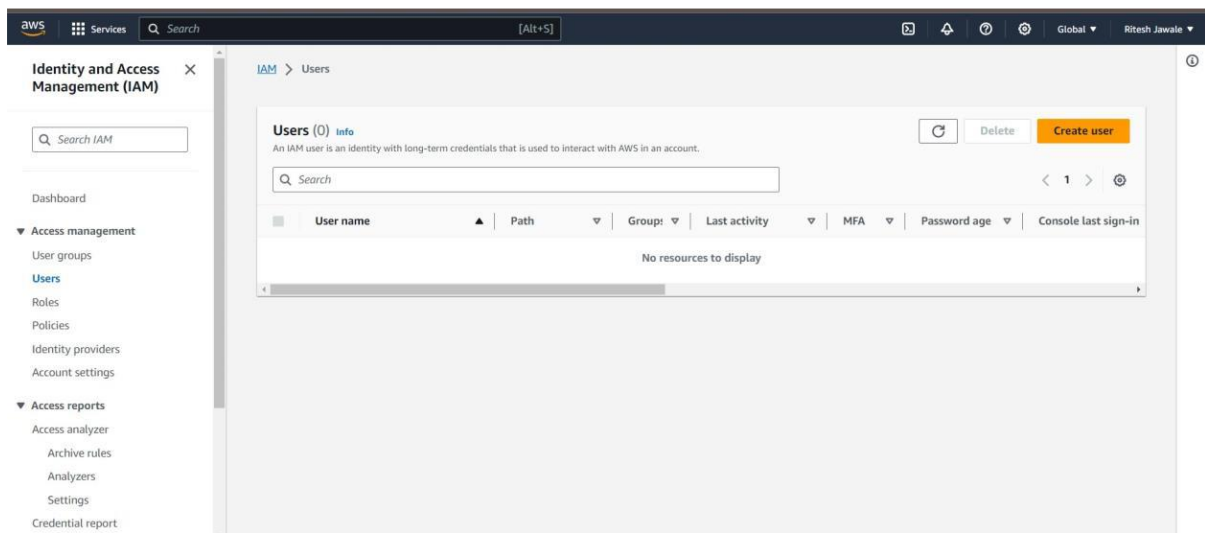
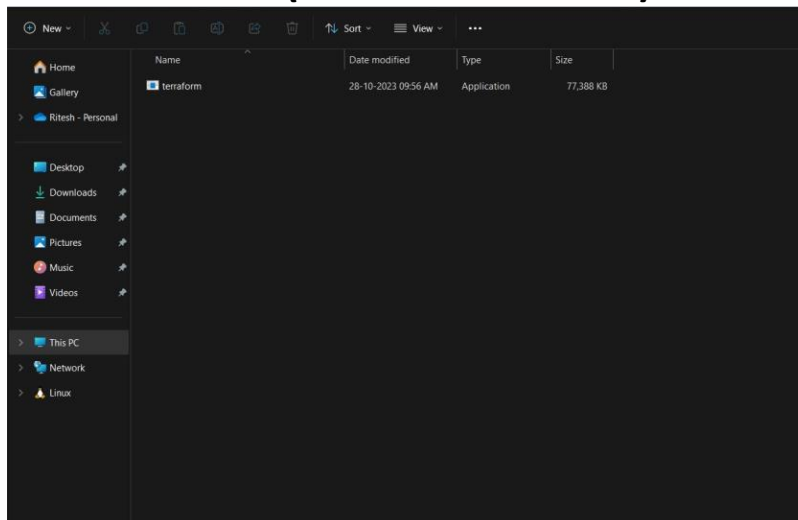
Academic Year: 2023-2024

Subject Name & Code: Cloud Computing & Analytics: ADUA31203

Title of Assignment: Write IAC using terraform to create EC2 machine on AWS.

## ASSIGNMENT NO. 5

# Terraform file(before execution):



# CODE:

```
main.tf
C:\Terraform> main.tf provider "aws"
1 provider "aws" {
2   region = "ap-south-1"
3   access_key = "AKIAVQ334G76NZRMYGNE"
4   secret_key = "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"
5 }
6 #1. Create vpc
7
8 resource "aws_vpc" "prod-vpc" {
9   cidr_block = "10.0.0.0/16"
10  tags = {
11    Name = "production"
12  }
13 }
14
15 #2. Create Internet Gateway
16
17 resource "aws_internet_gateway" "gw" {
18   vpc_id = aws_vpc.prod-vpc.id
19 }
20
21 #3. Create Custom Route Table
22
23 resource "aws_route_table" "prod-route-table" {
24   vpc_id = aws_vpc.prod-vpc.id
25
26   route {
27     cidr_block = "0.0.0.0/0"
28     gateway_id = aws_internet_gateway.gw.id
29   }
30
31   route {
32     ipv6_cidr_block = ":::/0"
33   }
34 }
```

```
main.tf
C:\Terraform> main.tf provider "aws"
27   route {
28     cidr_block = "0.0.0.0/0"
29     gateway_id = aws_internet_gateway.gw.id
30   }
31
32   route {
33     ipv6_cidr_block = ":::/0"
34     gateway_id = aws_internet_gateway.gw.id
35   }
36
37   tags = {
38     Name = "Prod"
39   }
40 }
41
42 #4. Create a Subnet
43
44 resource "aws_subnet" "subnet-1" {
45   vpc_id = aws_vpc.prod-vpc.id
46   cidr_block = "10.0.1.0/24"
47   availability_zone = "ap-south-1b"
48
49   tags = {
50     Name = "prod-subnet"
51   }
52 }
53
54 #5. Associate subnet with Route Table
55 resource "aws_route_table_association" "a" {
56   subnet_id = aws_subnet.subnet-1.id
57   route_table_id = aws_route_table.prod-route-table.id
58 }
59 }
```

```
File Edit Selection View Go Run ... Search
main.tf x
C:\> Terraform > main.tf > provider "aws"

54 #5. Associate subnet with Route Table
55 resource "aws_route_table_association" "a" {
56     subnet_id      = aws_subnet.subnet-1.id
57     route_table_id = aws_route_table.prod-route-table.id
58 }
59
60 #6. Create Security Group to allow port 22,80,443
61 resource "aws_security_group" "allow_web" {
62     name            = "allow_web_traffic"
63     description     = "Allow Web inbound traffic"
64     vpc_id          = aws_vpc.prod-vpc.id
65
66     ingress {
67         description = "HTTPS"
68         from_port   = 443
69         to_port     = 443
70         protocol    = "tcp"
71         cidr_blocks = ["0.0.0.0/0"]
72     }
73     ingress {
74         description = "HTTP"
75         from_port   = 80
76         to_port     = 80
77         protocol    = "tcp"
78         cidr_blocks = ["0.0.0.0/0"]
79     }
80     ingress {
81         description = "SSH"
82         from_port   = 22
83         to_port     = 22
84         protocol    = "tcp"
85         cidr_blocks = ["0.0.0.0/0"]
86     }
87 }
88
89 #7. Create a network interface with an ip in the subnet that was created in step 4
90 resource "aws_network_interface" "web-server-nic" {
91     subnet_id      = aws_subnet.subnet-1.id
92     private_ips    = ["10.0.1.50"]
93     security_groups = [aws_security_group.allow_web.id]
94 }
95
96 #8. Assign an elastic IP to the network interface created in step 7
97 resource "aws_eip" "web-server-eip" {
98     instance_id    = aws_instance.web-server.id
99     network_interface_id = aws_network_interface.web-server-nic.id
100 }
```

```
File Edit Selection View Go Run ... Search
main.tf x
C:\> Terraform > main.tf > provider "aws"

77     protocol      = "tcp"
78     cidr_blocks    = ["0.0.0.0/0"]
79 }
80 ingress {
81     description = "SSH"
82     from port   = 22
83     to port     = 22
84     protocol    = "tcp"
85     cidr_blocks = ["0.0.0.0/0"]
86 }
87
88 egress {
89     from port   = 0
90     to port     = 0
91     protocol    = "-1"
92     cidr_blocks = ["0.0.0.0/0"]
93 }
94
95 tags = {
96     Name = "allow_web"
97 }
98 }
99
100 #7. Create a network interface with an ip in the subnet that was created in step 4
101 resource "aws_network_interface" "web-server-nic" {
102     subnet_id      = aws_subnet.subnet-1.id
103     private_ips    = ["10.0.1.50"]
104     security_groups = [aws_security_group.allow_web.id]
105 }
106
107 #8. Assign an elastic IP to the network interface created in step 7
108 resource "aws_eip" "web-server-eip" {
109     instance_id    = aws_instance.web-server.id
110     network_interface_id = aws_network_interface.web-server-nic.id
111 }
```

```
File Edit Selection View Go Run ... Search
main.tf x
C:\Terraform> main.tf provider "aws"

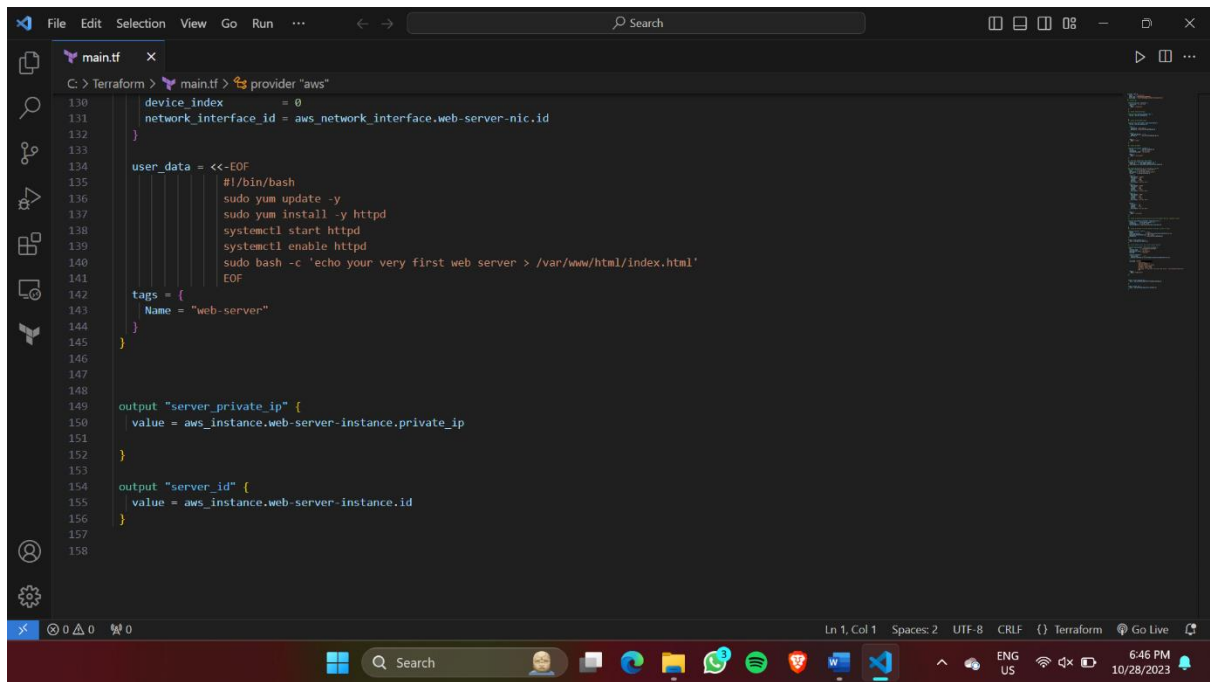
100 #7. Create a network interface with an ip in the subnet that was created in step 4
101
102 resource "aws_network_interface" "web-server-nic" {
103     subnet_id      = aws_subnet.subnet-1.id
104     private_ips    = ["10.0.1.50"]
105     security_groups = [aws_security_group.allow_web.id]
106 }
107
108 #8. Assign an elastic IP to the network interface created in step 7
109
110 resource "aws_eip" "one" {
111     domain            = "vpc"
112     network_interface = aws_network_interface.web-server-nic.id
113     associate_with_private_ip = "10.0.1.50"
114     depends_on        = [aws_internet_gateway.gw]
115 }
116
117 output "server_public_ip" {
118     value = aws_eip.one.public_ip
119 }
120
121 #9. Create Ubuntu server and install/enable apache2
122
123 resource "aws_instance" "web-server-instance" {
124     ami              = "ami-06791f9213cbb608b"
125     instance_type    = "t2.micro"
126     availability_zone = "ap-south-1b"
127     key_name         = "cca_aws5"
128
129     network_interface {
130         device_index = 0
131         network_interface_id = aws_network_interface.web-server-nic.id
132     }
133 }

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} Terraform Go Live
```

```
File Edit Selection View Go Run ... Search
main.tf x
C:\Terraform> main.tf provider "aws"

122
123 resource "aws_instance" "web-server-instance" {
124     ami              = "ami-06791f9213cbb608b"
125     instance_type    = "t2.micro"
126     availability_zone = "ap-south-1b"
127     key_name         = "cca_aws5"
128
129     network_interface {
130         device_index = 0
131         network_interface_id = aws_network_interface.web-server-nic.id
132     }
133
134     user_data = <<-EOF
135     |          #!/bin/bash
136     |          sudo yum update -y
137     |          sudo yum install -y httpd
138     |          systemctl start httpd
139     |          systemctl enable httpd
140     |          sudo bash -c 'echo your very first web server > /var/www/html/index.html'
141     |          EOF
142     tags = {
143         Name = "web-server"
144     }
145 }
146
147
148
149 output "server_private_ip" {
150     value = aws_instance.web-server-instance.private_ip
151 }
152
153
154 output "server_id" {

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} Terraform Go Live
```

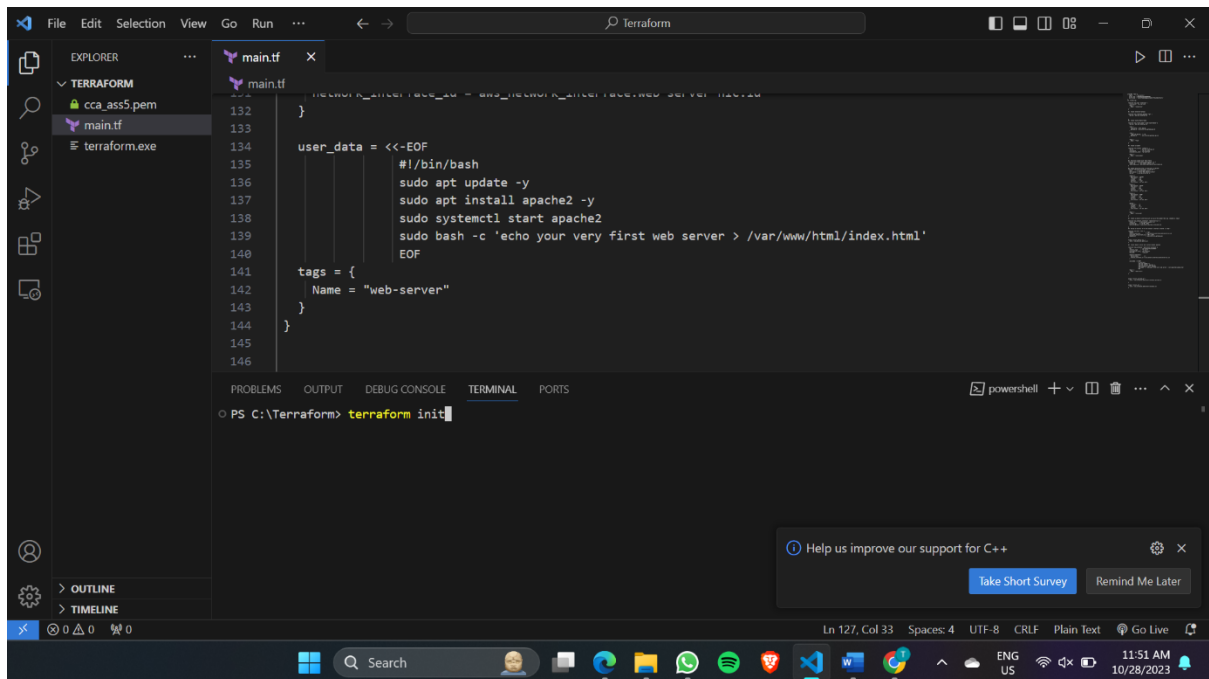


```
130     device_index      = 0
131     network_interface_id = aws_network_interface.web-server-nic.id
132   }
133 }
134
135   user_data = <<-EOF
136   |         #!/bin/bash
137   |         sudo yum update -y
138   |         sudo yum install -y httpd
139   |         systemctl start httpd
140   |         systemctl enable httpd
141   |         sudo bash -c 'echo your very first web server > /var/www/html/index.html'
142   |     EOF
143
144   tags = {
145     Name = "web-server"
146   }
147
148
149   output "server_private_ip" {
150     value = aws_instance.web-server-instance.private_ip
151   }
152
153
154   output "server_id" {
155     value = aws_instance.web-server-instance.id
156   }
157
158
```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} Terraform Go Live

# Terraform Commands:

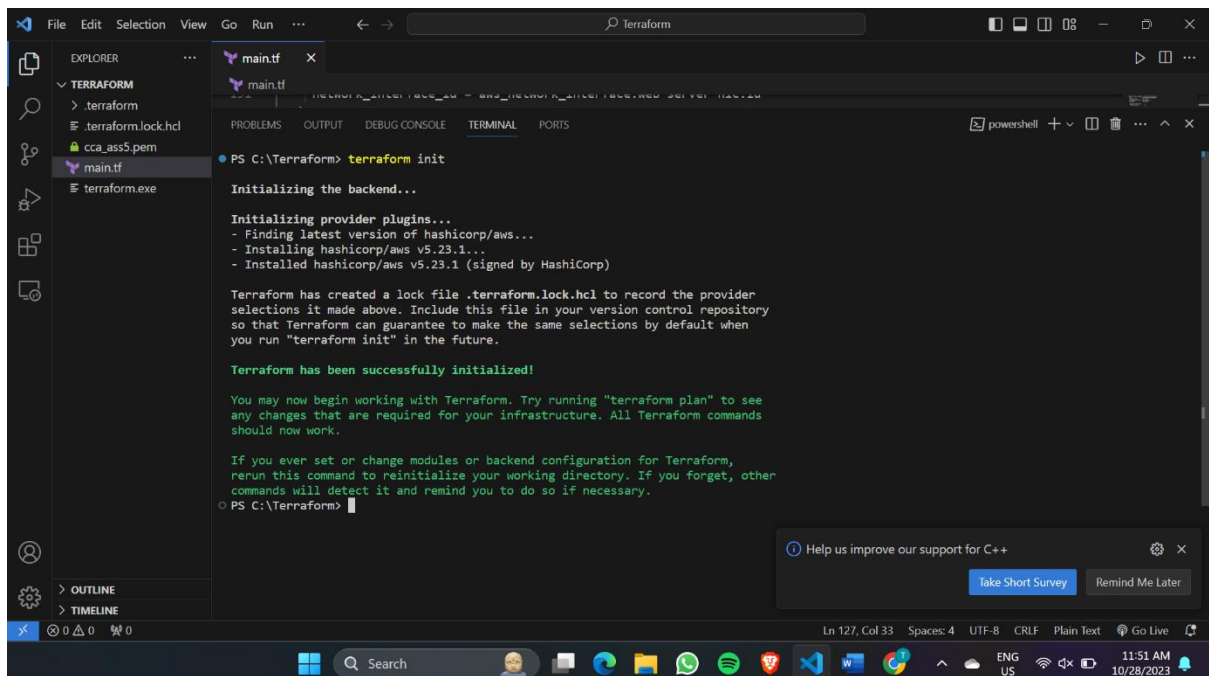
- Terraform init:



The screenshot shows the Visual Studio Code editor with a Terraform configuration file named `main.tf` open. The file contains a configuration for a web server using the `aws_instance` resource. The terminal window at the bottom shows the command `terraform init` being executed in a PowerShell session.

```
main.tf
132 }
133
134 user_data = <<-EOF
135     #!/bin/bash
136     sudo apt update -y
137     sudo apt install apache2 -y
138     sudo systemctl start apache2
139     sudo bash -c 'echo your very first web server > /var/www/html/index.html'
140 EOF
141
142 tags = {
143     Name = "web-server"
144 }
145
146
```

```
PS C:\Terraform> terraform init
```



The screenshot shows the Visual Studio Code editor with the same Terraform configuration file `main.tf` open. The terminal window now displays the output of the `terraform init` command, showing the initialization of the backend and provider plugins.

```
PS C:\Terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.23.1...
- Installed hashicorp/aws v5.23.1 (signed by HashiCorp)

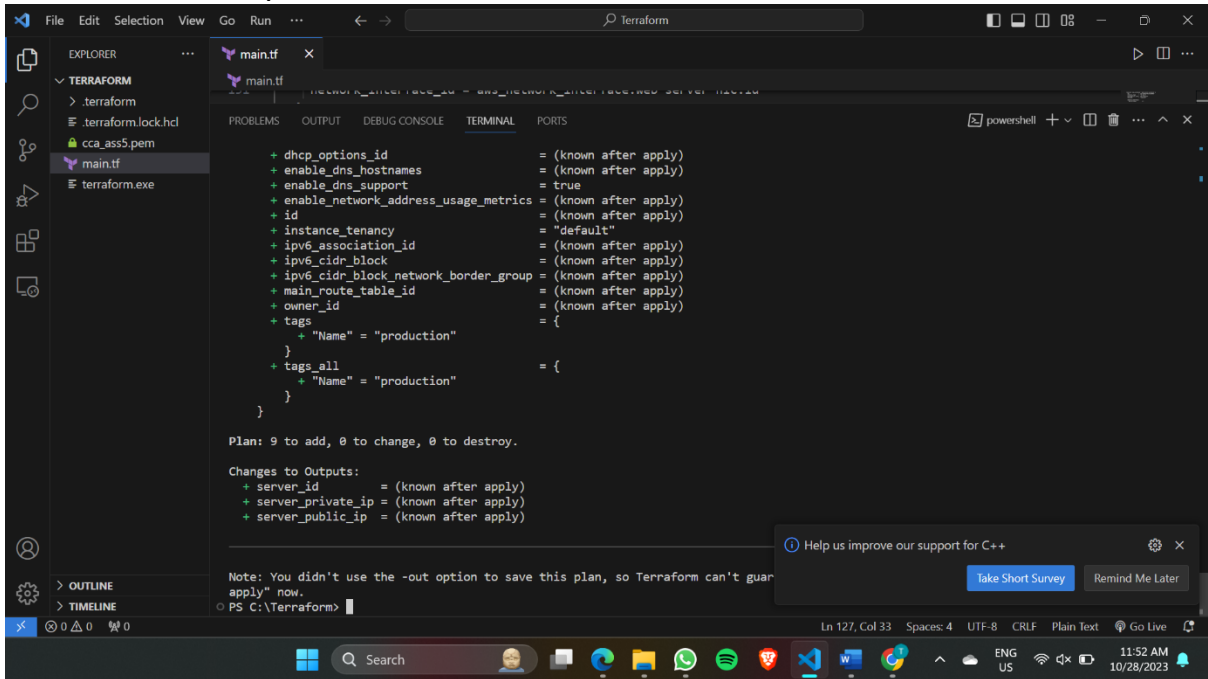
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

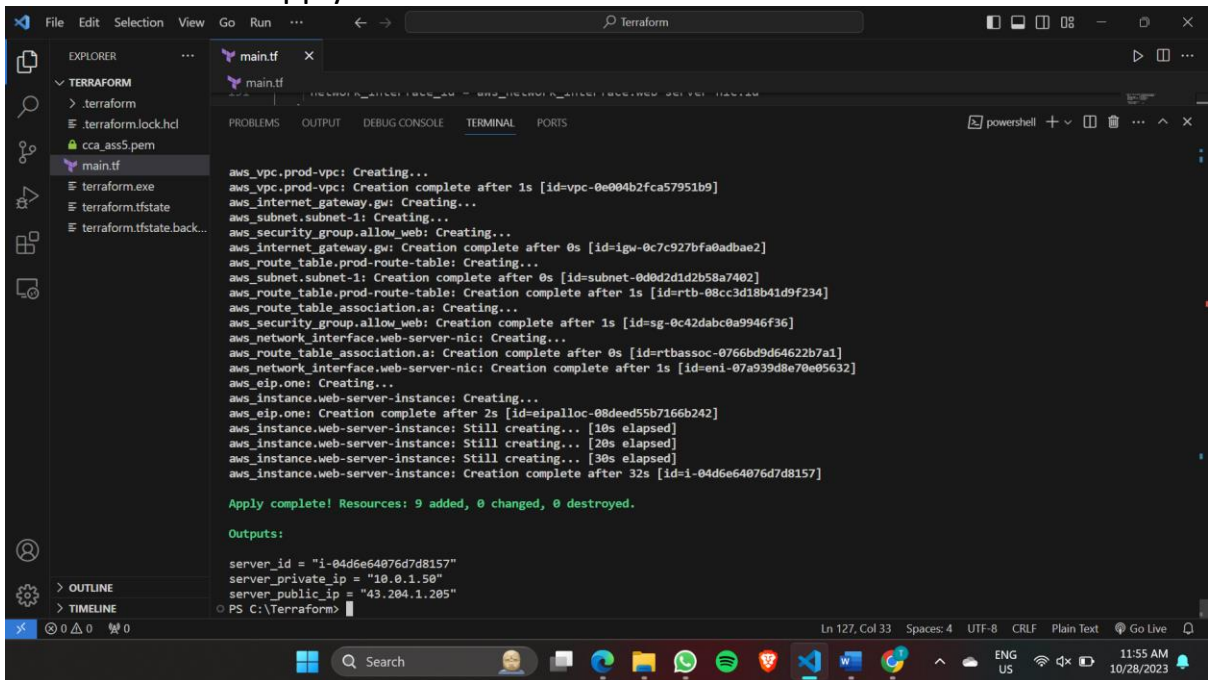
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Terraform>
```

- Terraform plan:



- Terraform apply:





EC2 Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

VPC dashboard

EC2 Global View New

Filter by VPC:

Select a VPC

Virtual private cloud

Your VPCs New

Subnets

Route tables

Internet gateways

Egress-only internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

VPC dashboard

EC2 Global View New

Filter by VPC:

Select a VPC

Virtual private cloud

Your VPCs New

Subnets

Route tables

Internet gateways

Egress-only internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

Instances (1) Info

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Clear filters

< 1 >

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	web-server	i-04d6e64076d7d8157	Running	t2.micro	Initializing	No alarms

Select an instance

Your VPCs (2) Info

Actions

Create VPC

Search

< 1 >

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR
<input type="checkbox"/>	-	vpc-036c563bb70544647	Available	172.31.0.0/16
<input type="checkbox"/>	production	vpc-0e004b2fca57951b9	Available	10.0.0.0/16

Select a VPC above

Subnets (4) Info

Actions

Create subnet

Find resources by attribute or tag

< 1 >

<input type="checkbox"/>	Name	Subnet ID	State	VPC
<input type="checkbox"/>	-	subnet-03245a3643c3b84db	Available	vpc-036c563bb70544647
<input type="checkbox"/>	prod-subnet	subnet-0d0d2d1d2b58a7402	Available	vpc-0e004b2fca57951b9
<input type="checkbox"/>	-	subnet-0406fbadc663356f8	Available	vpc-036c563bb70544647
<input type="checkbox"/>	-	subnet-06e8375a544763957	Available	vpc-036c563bb70544647

Select a subnet

VPC dashboard

EC2 Global View

Filter by VPC:

Virtual private cloud

Subnets

Route tables

Internet gateways

Egress-only internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

VPC dashboard

EC2 Global View

Filter by VPC:

Virtual private cloud

Subnets

Route tables

Internet gateways

Egress-only internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

your very first web server

Route tables (3)

Find resources by attribute or tag

Actions

Create route table

	Name	Route table ID	Explicit subnet associati...	Edge associations
<input type="checkbox"/>	-	rtb-08b2c573c35584b24	-	-
<input type="checkbox"/>	-	rtb-0ec81898563774ca0	-	-
<input type="checkbox"/>	Prod	rtb-08cc3d18b41d9f234	subnet-0d0d2d1d2b58a7...	-

Select a route table

Internet gateways (2)

Search

Actions

Create internet gateway

	Name	Internet gateway ID	State	VPC ID
<input type="checkbox"/>	-	igw-09063ce4f52ada5c8	Attached	vpc-036c563bb70
<input type="checkbox"/>	-	igw-0c7c927bfa0adbae2	Attached	vpc-0e004b2fca57

Select an internet gateway above

- Terraform destroy:

```

main.tf
141     tags = {
142       Name = "web-server"
143     }
144   }
145 }
146
147 output "server private ip" {
148   value = aws_instance.web-server-instance.private_ip
149 }

```

```

PS C:\Terraform> terraform destroy
aws_vpc.prod-vpc: Refreshing state... [id=vpc-0e004b2fca57951b9]
aws_internet_gateway.gw: Refreshing state... [id=igw-0c7c927bfa0adbae2]
aws_subnet.subnet-1: Refreshing state... [id=subnet-0d0d2d1d2b58a7402]
aws_security_group.allow_web: Refreshing state... [id=sg-0c42dabc0a9946f36]
aws_route_table.prod-route-table: Refreshing state... [id=rtb-08cc3d18b41d9f22a]
aws_network_interface.web-server-nic: Refreshing state... [id=eni-07a939d8e70e05632]
aws_route_table_association.a: Refreshing state... [id=rtbassoc-0766bd9d64622b7a1]
aws_eip.one: Refreshing state... [id=eipalloc-08deed55b7166b242]
aws_instance.web-server-instance: Refreshing state... [id=i-04d6e64076d7d8157]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_eip.one will be destroyed
- resource "aws_eip" "one" {
  - allocation_id           = "eipalloc-08deed55b7166b242" -> null
  - associate_with_private_ip = "10.0.1.50" -> null
  - association_id          = "eipassoc-05ef4fae731b24fd2" -> null
  - domain                  = "vpc" -> null
}

```

```

main.tf
141     tags = {
142       Name = "web-server"
143     }
144   }
145 }
146
147 output "server private ip" {
148   value = aws_instance.web-server-instance.private_ip
149 }

```

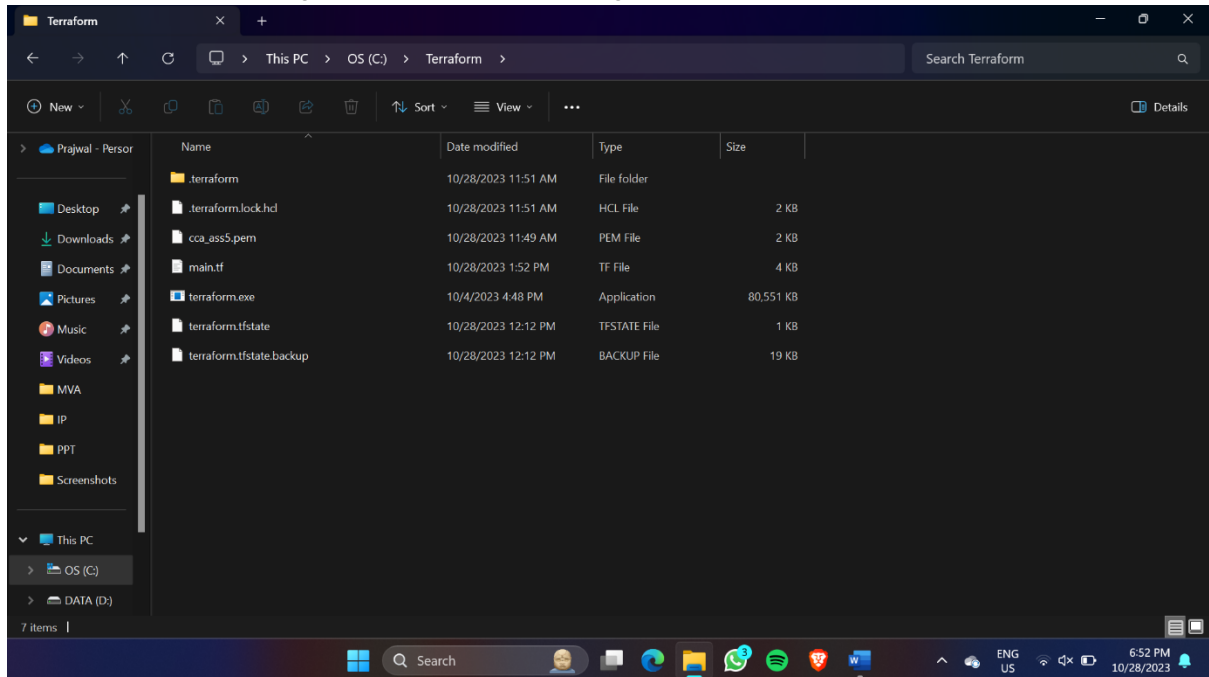
```

aws_eip.one: Destruction complete after 2s
aws_internet_gateway.gw: Destroying... [id=igw-0c7c927bfa0adbae2]
aws_internet_gateway.gw: Destruction complete after 0s
aws_instance.web-server-instance: Still destroying... [id=i-04d6e64076d7d8157, 10s elapsed]
aws_instance.web-server-instance: Still destroying... [id=i-04d6e64076d7d8157, 20s elapsed]
aws_instance.web-server-instance: Still destroying... [id=i-04d6e64076d7d8157, 30s elapsed]
aws_instance.web-server-instance: Destruction complete after 31s
aws_network_interface.web-server-nic: Destroying... [id=eni-07a939d8e70e05632]
aws_network_interface.web-server-nic: Destruction complete after 1s
aws_subnet.subnet-1: Destroying... [id=subnet-0d0d2d1d2b58a7402]
aws_security_group.allow_web: Destroying... [id=sg-0c42dabc0a9946f36]
aws_subnet.subnet-1: Destruction complete after 0s
aws_security_group.allow_web: Destruction complete after 1s
aws_vpc.prod-vpc: Destroying... [id=vpc-0e004b2fca57951b9]
aws_vpc.prod-vpc: Destruction complete after 1s

Destroy complete! Resources: 9 destroyed.
PS C:\Terraform>

```

## Terraform file (after execution):



Conclusion: Thus, we have successfully Write IAC using terraform to create EC2 machine on AWS.