

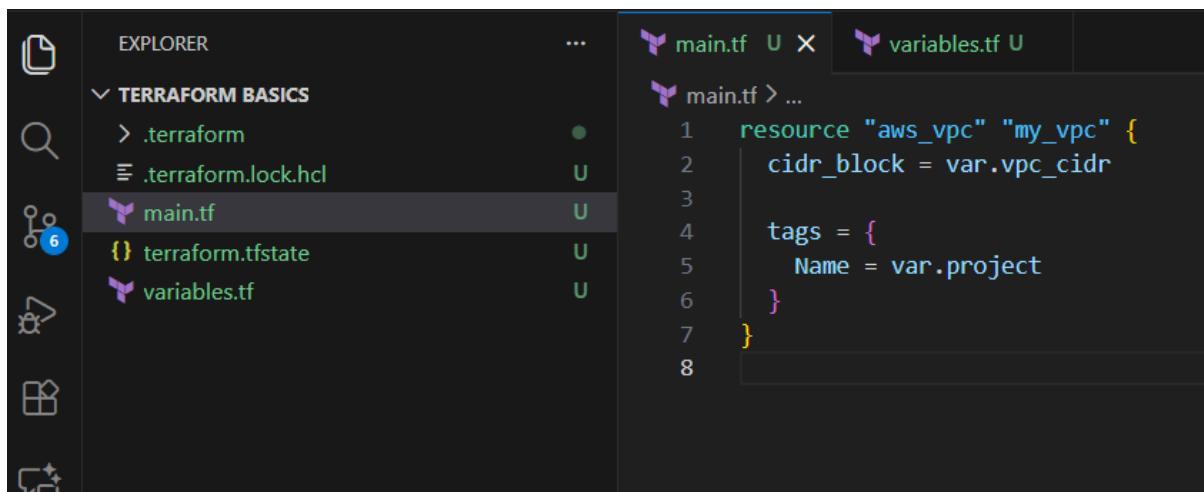
1. Create VPC

Give main.tf as this code.

Main.tf:

```
variable "project" {  
  default = "demo-vpc"  
}
```

```
variable "vpc_cidr" {  
  default = "10.0.0.0/16"  
}
```



```
resource "aws_vpc" "my_vpc" {  
  cidr_block = var.vpc_cidr  
  
  tags = {  
    Name = var.project  
  }  
}
```

Write the code in variables.tf as

Variables.tf:

```
variable "project" {  
  default = "demo-vpc"  
}
```

```

variable "vpc_cidr" {
  default = "10.0.0.0/16"
}

```

```

variables.tf > ...
1 variable "project" {
2   default = "demo-vpc"
3 }
4
5 variable "vpc_cidr" {
6   default = "10.0.0.0/16"
7 }
8

```

- `terraform init`
- `terraform apply`

```

PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.21.0...
- Installed hashicorp/aws v6.21.0 (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,

```

PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
+ create

Terraform will perform the following actions:

# aws_vpc.my_vpc will be created
+ resource "aws_vpc" "my_vpc" {
    + arn
    + cidr_block
    + default_network_acl_id
    + default_route_table_id
    + default_security_group_id
    + dhcp_options_id
    + enable_dns_hostnames
    + enable_dns_support
    + enable_network_address_usage_metrics = (known after apply)
}

```

The screenshot shows the VS Code interface with the terminal tab active. The terminal output displays the Terraform plan and execution process:

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

        + "Name" = "demo-vpc"
    }
+ tags_all
    + "Name" = "demo-vpc"
}
}

Plan: 1 to add, 0 to change, 0 to destroy.

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_vpc.my_vpc: Creating...
aws_vpc.my_vpc: Creation complete after 5s [id=vpc-0753d381cd7a55866]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

```

A vpc has been created.

The screenshot shows the AWS VPC dashboard. A red arrow points to the 'demo-vpc' entry in the 'Your VPCs' table. The table lists two VPCs:

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
default	vpc-06cf45eaab13624fe	Available	Off	172.31.0.0/16	-
demo-vpc	vpc-0753d381cd7a55866	Available	Off	10.0.0.0/16	-

2. Create Internet gateway

Give the resource in main as

```
resource "aws_internet_gateway" "igw" {
```

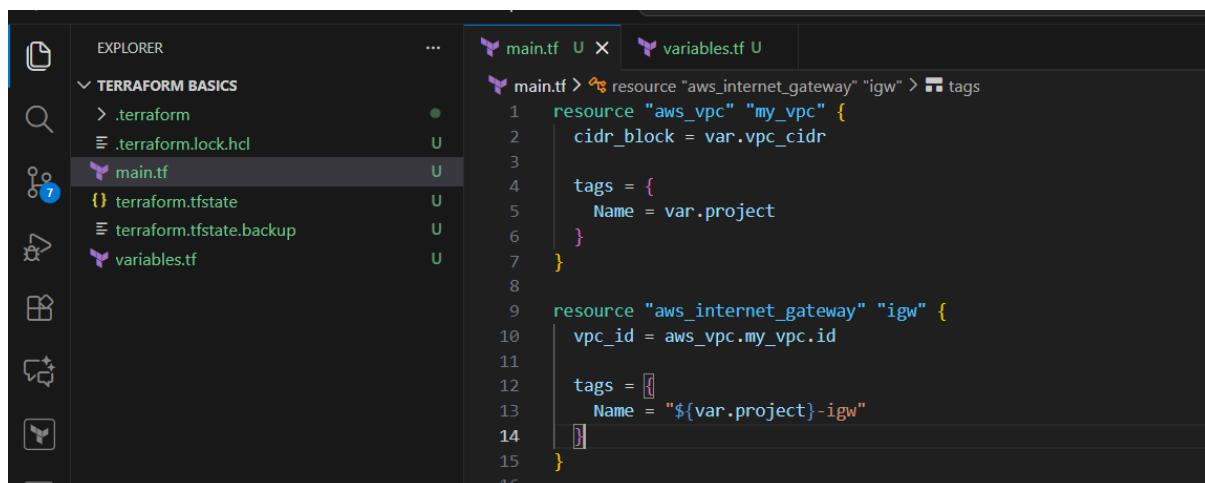
```
    vpc_id = aws_vpc.my_vpc.id
```

```
    tags = {
```

```
        Name = "${var.project}-igw"
```

```
}
```

```
}
```



```
variable "vpc_cidr" { type = "string" }

resource "aws_vpc" "my_vpc" {
  cidr_block = var.vpc_cidr
}

tags = {
  Name = var.project
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.my_vpc.id
}

tags = [
  {
    Name = "${var.project}-igw"
  }
]
```

Keep the variables.tf as same

Give the commands terraform init,terraform apply.

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]

Terraform used the selected providers to generate the following execution plan. Resources:
+ create
```

The screenshot shows a terminal window with several tabs at the top: PROBLEMS (1), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS. The terminal content is as follows:

```
}
```

+ tags_all = {
 + "Name" = "demo-vpc-igw"
}
+ vpc_id = "vpc-0753d381cd7a55866"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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_internet_gateway.igw: Creating...
aws_internet_gateway.igw: Creation complete after 2s [id=igw-079b7b741ab2a1d8e]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

A internet gateway has been created and attached to your created vpc.

The screenshot shows the AWS VPC Internet Gateways page. The left sidebar has sections for VPC dashboard, AWS Global View, Virtual private cloud (Your VPCs, Subnets, Route tables), and Internet gateways (Egress-only Internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers). The main content area shows a table titled "Internet gateways (1/2) info". The table has columns: Name, Internet gateway ID, State, VPC ID, and Owner. It contains two rows: "default-IGW" (id: igw-0e94bad7e15ce8d1, state: Attached, vpc-id: vpc-06cf45aab13624fe, owner: 235351028455) and "demo-vpc-igw" (id: igw-079b7b741ab2a1d8e, state: Attached, vpc-id: vpc-0753d381cd7a55866, owner: 235351028455). A checkbox next to "demo-vpc-igw" is checked. Below the table, a specific internet gateway is selected: "igw-079b7b741ab2a1d8e / demo-vpc-igw". The "Details" tab is selected, showing the following information:

Internet gateway ID igw-079b7b741ab2a1d8e	State Attached	VPC ID vpc-0753d381cd7a55866 demo-vpc	Owner 235351028455
--	-----------------------------------	--	---------------------------------------

3. Create Custom Route Table

Add resource in the main.tf

```
resource "aws_route_table" "custom_rt" {
```

```
  vpc_id = aws_vpc.my_vpc.id
```

```
  tags = {
```

```
    Name = "${var.project}-route-table"
```

```
}
```

```
}
```

```
main.tf
resource "aws_internet_gateway" "igw" {
  tags = {
    Name = "${var.project}-igw"
  }
}

resource "aws_route_table" "custom_rt" {
  vpc_id = aws_vpc.my_vpc.id
  tags = {
    Name = "${var.project}-route-table"
  }
}
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
```

Terraform used the selected providers to generate the following execution plan. Resources

```

}
+ tags_all      = {
  + "Name" = "demo-vpc-route-table"
}
+ vpc_id        = "vpc-0753d381cd7a55866"

Plan: 1 to add, 0 to change, 0 to destroy.

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_route_table.custom_rt: Creating...
aws_route_table.custom_rt: Creation complete after 2s [id=rtb-0c60d4a3229f73727]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

A routetable has been created for your vpc

The screenshot shows the AWS VPC Route Tables page. A red arrow points to the first row in the table, which is selected. The table has columns for Name, Route table ID, Explicit subnet associations, Edge associations, Main, and VPC. The selected row is ' - ' with Route table ID 'rtb-0d05926c91bc66071'. The other two rows are 'default-private-routetable' and 'default-public-route table'.

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC
-	rtb-0d05926c91bc66071	-	-	Yes	vpc-0753d381cd7a55866 dem.
default-private-routetable	rtb-064fa8fd4ee3109e9	subnet-04f12a817188fca...	-	No	vpc-06cf45eaab13624fe defaul
default-public-route table	rtb-01a414b0e734ca840	subnet-0a192382de0e2b...	-	Yes	vpc-06cf45eaab13624fe defaul

4. Create Subnet

Add this resource block in main.tf

```
resource "aws_subnet" "my_subnet" {  
    vpc_id          = aws_vpc.my_vpc.id  
    cidr_block      = var.subnet_cidr  
    availability_zone = var.az  
  
    tags = {  
        Name = "${var.project}-subnet"  
    }  
}
```

```
main.tf
16 resource "aws_route_table" "custom_rt" {
17   vpc_id           = aws_vpc.my_vpc.id
18   cidr_block       = var.subnet_cidr
19   availability_zone = var.az
20
21   tags = {
22     Name = "${var.project}-subnet"
23   }
24 }
```

Add the data in the variables.tf

```
variable "subnet_cidr" {
  default  = "10.0.1.0/24"
}
```

```
variable "az" {
  default  = "us-east-1a"
}
```

```
variables.tf
6   default = "10.0.0.0/16
7 }
8 variable "subnet_cidr" [
9   default  = "10.0.1.0/24"
10 ]
11
12 variable "az" {
13   default  = "us-east-1a"
14 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

}
+ tags_all
  + "Name" = "demo-vpc-subnet"
}
+ vpc_id
}

Plan: 1 to add, 0 to change, 0 to destroy.

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_subnet.my_subnet: Creating...
aws_subnet.my_subnet: Creation complete after 3s [id=subnet-0b48bd42570b2da32]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

A subnet has been created for our vpc.

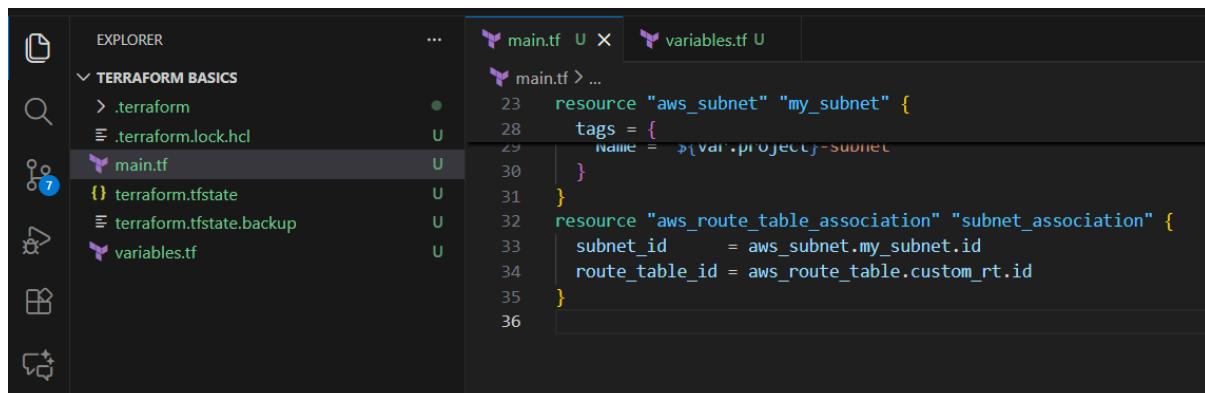
The screenshot shows the AWS VPC Subnets page. A red arrow points from the 'Subnets' link in the left sidebar to the table below. The table lists three subnets:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
<input checked="" type="checkbox"/> demo-vpc-subnet	subnet-0b48bd42570b2da32	Available	vpc-0753d381cd7a55866 dem...	Off	10.0.10.0/24
<input type="checkbox"/> default-private-subnet	subnet-04f12a817188fcad	Available	vpc-06cf45eaab13624fe default	Off	172.31.128.0/17
<input type="checkbox"/> default-public-subnet	subnet-0a192382de0e2bf6a	Available	vpc-06cf45eaab13624fe default	Off	172.31.0.0/17

5. Associate subnet with Route Table

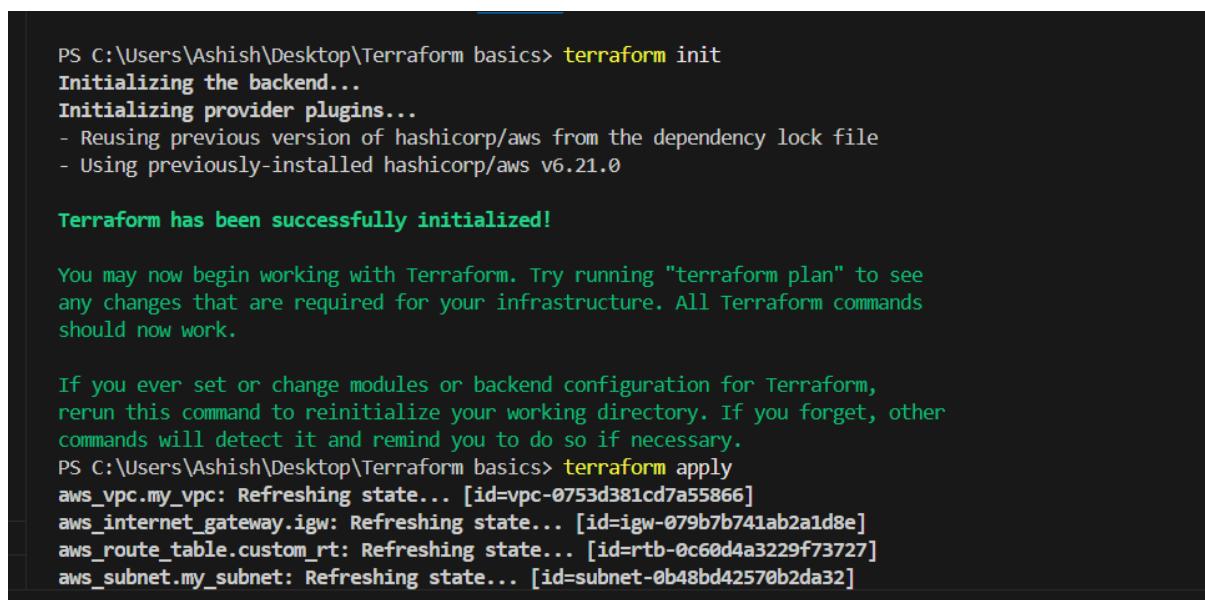
Add the resource in main.tf to associate subnet with route table.

```
resource "aws_route_table_association"  
"subnet_association" {  
  
    subnet_id      = aws_subnet.my_subnet.id  
  
    route_table_id = aws_route_table.custom_rt.id  
  
}
```



The screenshot shows the Visual Studio Code interface. On the left is the 'EXPLORER' sidebar with icons for files, folders, and other workspace items. Under 'TERRAFORM BASICS', there are entries for '.terraform', '.terraform.lock.hcl', 'main.tf' (which is selected), 'terraform.tfstate', 'terraform.tfstate.backup', and 'variables.tf'. The main editor area displays the 'main.tf' file with the following code:

```
resource "aws_subnet" "my_subnet" {  
    tags = {  
        Name = "var.project-subnet"  
    }  
  
resource "aws_route_table_association" "subnet_association" {  
    subnet_id      = aws_subnet.my_subnet.id  
    route_table_id = aws_route_table.custom_rt.id  
}
```



```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init  
Initializing the backend...  
Initializing provider plugins...  
- Reusing previous version of hashicorp/aws from the dependency lock file  
- Using previously-installed hashicorp/aws v6.21.0  
  
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:\Users\Ashish\Desktop\Terraform basics> terraform apply  
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]  
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]  
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]  
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
```

```

+ resource "aws_route_table_association" "subnet_association" {
  + id          = (known after apply)
  + region      = "us-east-1"
  + route_table_id = "rtb-0c60d4a3229f73727"
  + subnet_id    = "subnet-0b48bd42570b2da32"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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_route_table_association.subnet_association: Creating...
aws_route_table_association.subnet_association: Creation complete after 7s [id=rtbassoc-0d5fb1390a1079cb7]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>

```

An instance has been created with the routetable subnet association.

Name	Route table ID	Explicit subnet associ...	Main	VPC
<input checked="" type="checkbox"/> demo-vpc-route-table	rtb-0c60d4a3229f73727	subnet-0b48bd42570b2da32	-	vpc-0753d381cd7a55866 de
<input type="checkbox"/> -	rtb-0d05926c91bc66071	-	-	yes vpc-0753d381cd7a55866 de
<input type="checkbox"/> default-private-routetable	rtb-064fa8fd4ee3109e9	subnet-04f12a817188fca...	-	vpc-06cf45eaab13624fe def
<input type="checkbox"/> default-public-route table	rtb-01a414b0e734ca840	subnet-0a192382de0e2b...	-	vpc-06cf45eaab13624fe def

6. Create Security Group to allow port 22,80,443

Give the script in main.tf as

```
resource "aws_security_group" "web_sg" {

  name      = "${var.project}-sg"
  vpc_id    = aws_vpc.my_vpc.id
```

```
ingress {  
    description = "SSH"  
    from_port  = 22  
    to_port    = 22  
    protocol   = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
}
```

```
ingress {  
    description = "HTTP"  
    from_port  = 80  
    to_port    = 80  
    protocol   = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
}
```

```
ingress {  
    description = "HTTPS"  
    from_port  = 443  
    to_port    = 443
```

```

protocol  = "tcp"
cidr_blocks = ["0.0.0.0/0"]

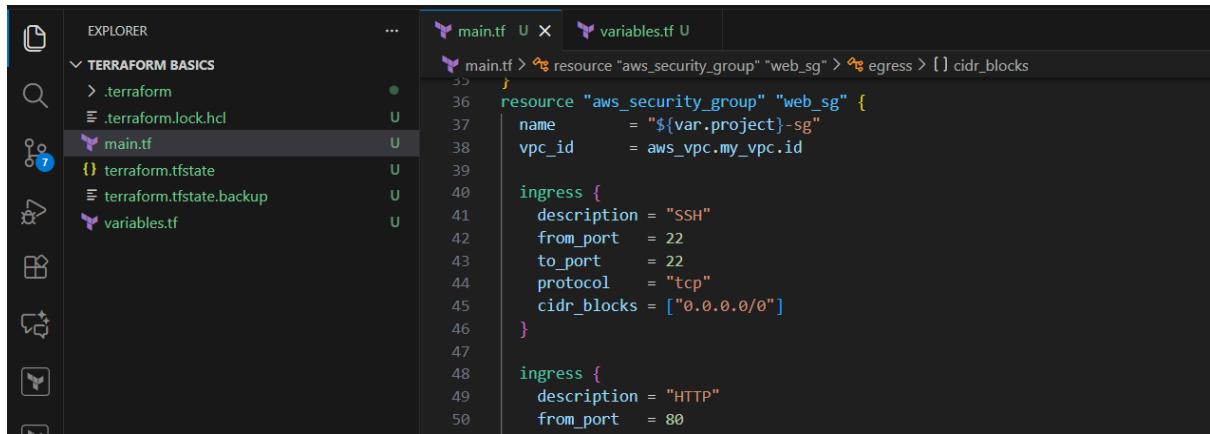
}

egress {
  description = "Allow all outbound"
  from_port  = 0
  to_port    = 0
  protocol   = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}

tags = {
  Name = "${var.project}-sg"
}

}

```



```

main.tf  U x  variables.tf  U
main.tf > resource "aws_security_group" "web_sg" > egress > [] cidr_blocks
35
36   resource "aws_security_group" "web_sg" {
37     name      = "${var.project}-sg"
38     vpc_id    = aws_vpc.my_vpc.id
39
40     ingress {
41       description = "SSH"
42       from_port  = 22
43       to_port    = 22
44       protocol   = "tcp"
45       cidr_blocks = ["0.0.0.0/0"]
46     }
47
48     ingress {
49       description = "HTTP"
50       from_port  = 80

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
+ tags_all          = {
  + "Name" = "demo-vpc-sg"
}
+ vpc_id           = "vpc-0753d381cd7a55866"
```

Plan: 1 to add, 0 to change, 0 to destroy.

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_sg: Creating...
aws_security_group.web_sg: Still creating... [00m10s elapsed]
aws_security_group.web_sg: Creation complete after 18s [id=sg-05399f4a8449bea38]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

A security group has been created with the 3 protocols.

The screenshot shows the AWS VPC dashboard with the 'Security Groups' section selected. A table lists three security groups: 'default', 'sg-0880f310383562b69', and 'demo-vpc-sg'. The 'demo-vpc-sg' row is selected, showing its details: Name (sg-05399f4a8449bea38), Security group ID (sg-05399f4a8449bea38), Security group name (demo-vpc-sg), VPC ID (vpc-0753d381cd7a55866), and Description (Managed by Terraform). Below the table, a detailed view for 'sg-05399f4a8449bea38 - demo-vpc-sg' is shown with tabs for Details, Inbound rules, Outbound rules, Sharing - new, VPC associations - new, and Tags. The 'Details' tab is active, displaying the same information as the table.

7. Create a network interface with an ip in the subnet that was created in step 4

Give the script in main.tf as

```
resource "aws_network_interface" "my_eni" {
    subnet_id      = aws_subnet.my_subnet.id
    private_ips    = [var.eni_private_ip]
    security_groups = [aws_security_group.web_sg.id]
```

```
tags = {
    Name = "${var.project}-eni"
}
}
```

```
main.tf
resource "aws_security_group" "web_sg" {
  tags = {
    Name = "${var.project}-sg"
  }
}

resource "aws_network_interface" "my_eni" {
  subnet_id      = aws_subnet.my_subnet.id
  private_ips    = [var.eni_private_ip]
  security_groups = [aws_security_group.web_sg.id]
}
```

Give the ip in variables.tf

```
variable "eni_private_ip" {
  default  = "10.0.1.10"
}
```

```
variables.tf
variable "eni_private_ip" {
  default  = "10.0.1.10"
}
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows files in the current workspace, including `.terraform`, `.terraform.lock.hcl`, `main.tf`, `terraform.tfstate`, `terraform.tfstate.backup`, and `variables.tf`.
- EDITOR**: Two tabs are open: `main.tf` and `variables.tf`.
 - `main.tf` contains a provider block for AWS.
 - `variables.tf` contains variable definitions for `az` and `eni_private_ip`.
- TERMINAL**: Displays the output of the `terraform init` command, which initializes the backend and provider plugins, and successfully initializes the Terraform environment.

The screenshot shows the terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ tags_all = {
    + "Name" = "demo-vpc-eni"
}

+ attachment (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

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_network_interface.my_eni: Creating...
aws_network_interface.my_eni: Creation complete after 5s [id=eni-074605fdc46607064]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

An network interface will be created with this.

The screenshot shows the AWS EC2 Network interfaces page. The left sidebar navigation includes: Capacity Manager, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), and Auto Scaling. The main content area displays a table titled 'Network interfaces (1/1)' with one item: 'demo-vpc-eni' (eni-074605fdc46607064). The interface is associated with subnet 'subnet-0b48bd42570b2da32', VPC 'vpc-0755d381cd7a55866', availability zone 'us-east-1a', and security group 'demo-vpc-sg'. Below the table, a detailed view for 'Network interface: eni-074605fdc46607064 (demo-vpc-eni)' is shown, with tabs for Details, Flow logs, and Tags. The 'Details' tab shows the following information:

Network interface ID	Name	Description
eni-074605fdc46607064	demo-vpc-eni	-
Network interface status	Interface type	Security groups
Available	Elastic network interface	sg-05399f4a8449bea38 (demo-vpc-sg)

8. Assign an elastic IP to the network interface created in step 7

Add this script in resource block in main.tf

```
resource "aws_eip" "eni_eip" {
    domain = "vpc"
```

```
network_interface = aws_network_interface.my_eni.id
```

```
associate_with_private_ip = var.eni_private_ip
```

```
tags = {
```

```
Name = "${var.project}-eni-eip"
```

```
}
```

```
}
```

```
main.tf > resource "aws_eip" "eni_eip" > tags > Name
76   resource "aws_network_interface" "my_eni" {
83     }
84   }
85   resource "aws_eip" "eni_eip" {
86     domain = "vpc"
87
88     network_interface = aws_network_interface.my_eni.id
89     associate_with_private_ip = var.eni_private_ip
90
91     tags = {
92       Name = "${var.project}-eni-eip"
93     }
94   }
95 }
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0

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:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
```

```
+ "Name" = "demo-vpc-eni-eip"
}
+ tags_all = {
  + "Name" = "demo-vpc-eni-eip"
}
}

Plan: 1 to add, 0 to change, 0 to destroy.

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_eip.eni_eip: Creating...
aws_eip.eni_eip: Creation complete after 4s [id=eipalloc-03c2ac0893ec73e3f]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

Elastic ip has been assigned to your network interface.

The image shows two screenshots of the AWS EC2 console. The top screenshot displays the 'Network interfaces' page with one interface listed: 'demo-vpc-eni' (eni-074605fdc46607064). A red arrow points from the 'Interface type' field ('Elastic network interface') to the 'Details' tab. The bottom screenshot shows the 'Elastic IP addresses' page with one address listed: 'demo-vpc-eni-eip' (3.208.232.169), which is associated with the 'demo-vpc-eni' interface.

9. Create Ubuntu server and install/enable apache2

Add the script in resource block in main.tf

```
resource "aws_instance" "ubuntu_server" {  
    ami      = "ami-0c02fb55956c7d316"  
    instance_type = "t3.micro"  
    key_name = "red"
```

```
network_interface {
```

```
    network_interface_id = aws_network_interface.my_eni.id
```

```

device_index      = 0

}

user_data = <<-EOF

#!/bin/bash

apt-get update -y

apt-get install apache2 -y

systemctl enable apache2

systemctl start apache2

EOF

```

```

tags = {

  Name = "${var.project}-ubuntu-server"

}

```

```

main.tf 1, U
variables.tf U

main.tf > resource "aws_instance" "ubuntu_server" > user_data
94 }
95 resource "aws_instance" "ubuntu_server" {
96   ami           = "ami-0c02fb55956c7d316"
97   instance_type = "t3.micro"
98
99   network_interface {
100     network_interface_id = aws_network_interface.my_eni.id
101     device_index         = 0
102
103
104   user_data = <<-EOF
105     #!/bin/bash
106     apt-get update -y
107     apt-get install apache2 -y
108     systemctl enable apache2
109     systemctl start apache2

```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0
```

```
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:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_vpc.my_vpc: Refreshing state... [id=vpc-0753d381cd7a55866]
aws_internet_gateway.igw: Refreshing state... [id=igw-079b7b741ab2a1d8e]
aws_subnet.my_subnet: Refreshing state... [id=subnet-0b48bd42570b2da32]
aws_route_table.custom_rt: Refreshing state... [id=rtb-0c60d4a3229f73727]
```

Only 'yes' will be accepted to approve.

Enter a value: yes

```
aws_instance.ubuntu_server: Creating...
aws_instance.ubuntu_server: Still creating... [00m10s elapsed]
aws_instance.ubuntu_server: Creation complete after 18s [id=i-0885d0bb79f526fe0]
```

Warning: Argument is deprecated

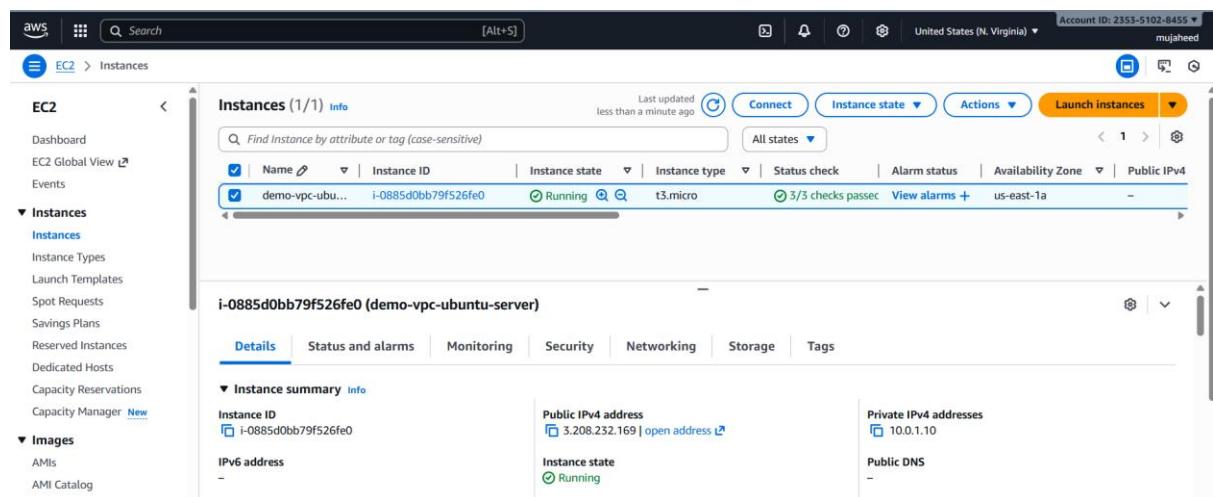
```
with aws_instance.ubuntu_server,
on main.tf line 95, in resource "aws_instance" "ubuntu_server":
95: resource "aws_instance" "ubuntu_server" {
```

network_interface is deprecated. To specify the primary network interface, use primary_network_interface instead. To attach additional network interfaces, use the aws_network_interface_attachment resource.

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

```
PS C:\Users\Ashish\Desktop\Terraform basics>
```

An ubuntu instance has been created.



If you copy your ip and search in browser with 80 port number you will see like.



main.tf:

```
resource "aws_vpc" "my_vpc" {
```

```
cidr_block = var.vpc_cidr
```

```
tags = {
```

```
  Name = var.project
```

```
}
```

```
}
```

```
resource "aws_internet_gateway" "igw" {
```

```
  vpc_id = aws_vpc.my_vpc.id
```

```
tags = {  
    Name = "${var.project}-igw"  
}  
}  
  
resource "aws_route_table" "custom_rt" {  
    vpc_id = aws_vpc.my_vpc.id
```

```
tags = {  
    Name = "${var.project}-route-table"  
}  
}
```

```
resource "aws_route" "internet_access" {  
    route_table_id      = aws_route_table.custom_rt.id  
    destination_cidr_block = "0.0.0.0/0"  
    gateway_id          = aws_internet_gateway.igw.id  
}
```

```
resource "aws_subnet" "my_subnet" {  
    vpc_id           = aws_vpc.my_vpc.id
```

```
cidr_block      = var.subnet_cidr
availability_zone = var.az

tags = {
    Name = "${var.project}-subnet"
}

resource "aws_route_table_association" "subnet_association" {
    subnet_id    = aws_subnet.my_subnet.id
    route_table_id = aws_route_table.custom_rt.id
}

resource "aws_security_group" "web_sg" {
    name        = "${var.project}-sg"
    vpc_id     = aws_vpc.my_vpc.id

ingress {
    description = "SSH"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
```

```
    cidr_blocks = ["0.0.0.0/0"]  
}  
  
  
```

```
ingress {  
    description = "HTTP"  
    from_port  = 80  
    to_port    = 80  
    protocol   = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
}  
  
  
```

```
ingress {  
    description = "HTTPS"  
    from_port  = 443  
    to_port    = 443  
    protocol   = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
}  
  
  
```

```
egress {  
    description = "Allow all outbound"  
}
```

```
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]

}

tags = {
    Name = "${var.project}-sg"
}
}

resource "aws_network_interface" "my_eni" {
    subnet_id      = aws_subnet.my_subnet.id
    private_ips    = [var.eni_private_ip]
    security_groups = [aws_security_group.web_sg.id]

}

tags = {
    Name = "${var.project}-eni"
}
}

resource "aws_eip" "eni_eip" {
    domain = "vpc"
```

```
network_interface = aws_network_interface.my_eni.id
associate_with_private_ip = var.eni_private_ip

tags = {
    Name = "${var.project}-eni-eip"
}
}

resource "aws_instance" "ubuntu_server" {
    ami      = "ami-0c02fb55956c7d316"
    instance_type = "t3.micro"
    key_name = "red"

    network_interface {
        network_interface_id = aws_network_interface.my_eni.id
        device_index      = 0
    }
}

user_data = <<-EOF
#!/bin/bash

apt-get update -y
```

```
apt-get install apache2 -y
systemctl enable apache2
systemctl start apache2
EOF
```

```
tags = {
    Name = "${var.project}-ubuntu-server"
}
}
```

Variables.tf:

```
variable "project" {
    default = "demo-vpc"
}
```

```
variable "vpc_cidr" {
    default = "10.0.0.0/16"
}
variable "subnet_cidr" {
    default = "10.0.1.0/24"
}
```

}

```
variable "az" {
  default    = "us-east-1a"
}

variable "eni_private_ip" {
  default    = "10.0.1.10"
}
```