AKSHIT MITTAL
10/14/23

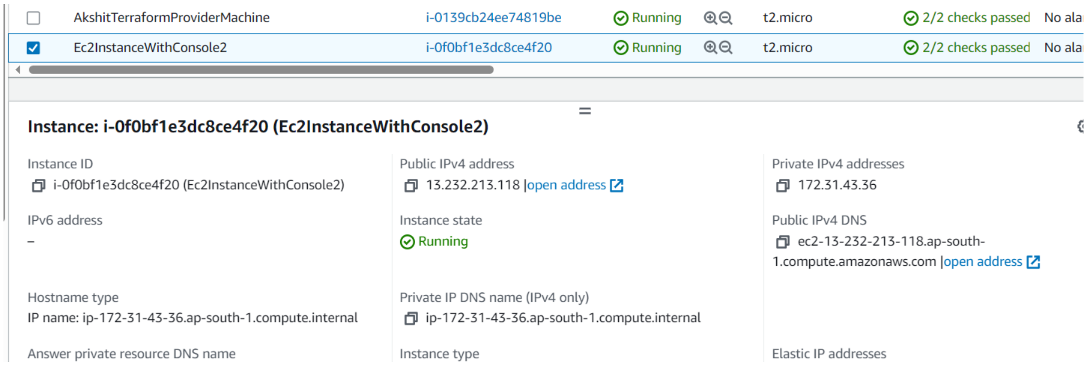
CLOUD & DEVOPS

EIP Association, Variables, Multiple Instances in AWS via Terraform

As we have created the EC2 instances virtual servers through terraform in the previous Terraform #2 module. Now, we will attach some other attributes like Elastic IP address to our machine and explore some other features of Terraform.

Link:- [aws_eip | Resources | hashicorp/aws | Terraform | Terraform Registry]
[aws_eip_association | Resources | hashicorp/aws | Terraform | Terraform Registry]

1. In the previous created instances, we can see the Elastic IP address is not present in the EC2 machine. In the Elastic IPs section, if there are no EIP present then we will create one.



□	AkshitTerraformProviderMachine	i-0139cb24ee74819be	✓ Running		t2.micro	✓ 2/2 checks passed	No ala
☑	Ec2InstanceWithConsole2	i-0f0bf1e3dc8ce4f20	✓ Running		t2.micro	✓ 2/2 checks passed	No ala

Instance: i-0f0bf1e3dc8ce4f20 (Ec2InstanceWithConsole2)

Instance ID
i-0f0bf1e3dc8ce4f20 (Ec2InstanceWithConsole2)

Public IPv4 address
13.232.213.118 |open address

Private IPv4 addresses
172.31.43.36

IPv6 address
–

Instance state
✓ Running

Public IPv4 DNS
ec2-13-232-213-118.ap-south-1.compute.amazonaws.com |open address

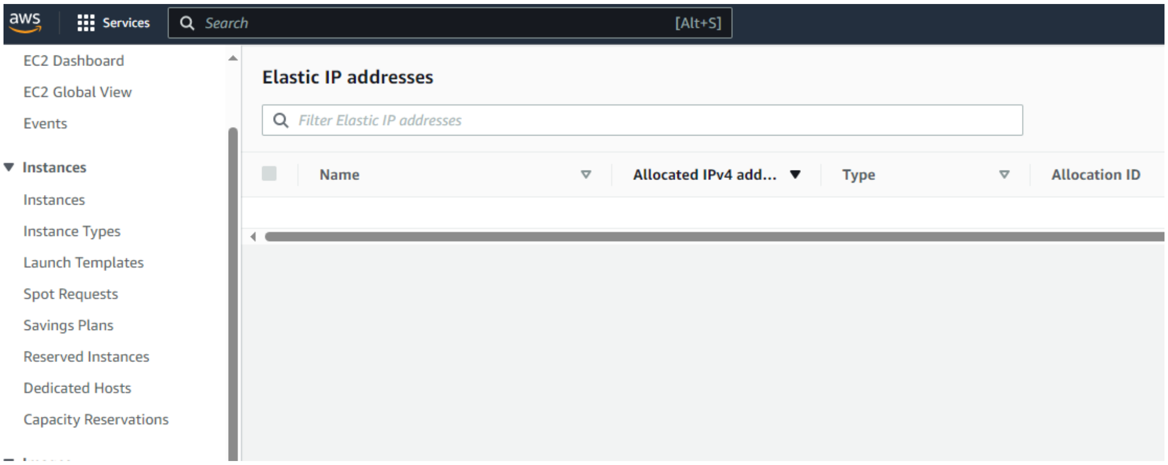
Hostname type
IP name: ip-172-31-43-36.ap-south-1.compute.internal

Private IP DNS name (IPv4 only)
ip-172-31-43-36.ap-south-1.compute.internal

Answer private resource DNS name

Instance type

Elastic IP addresses

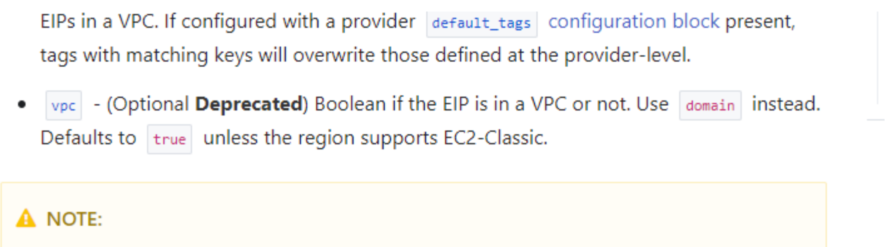


Elastic IP addresses

Filter Elastic IP addresses

□	Name	Allocated IPv4 add...	Type	Allocation ID

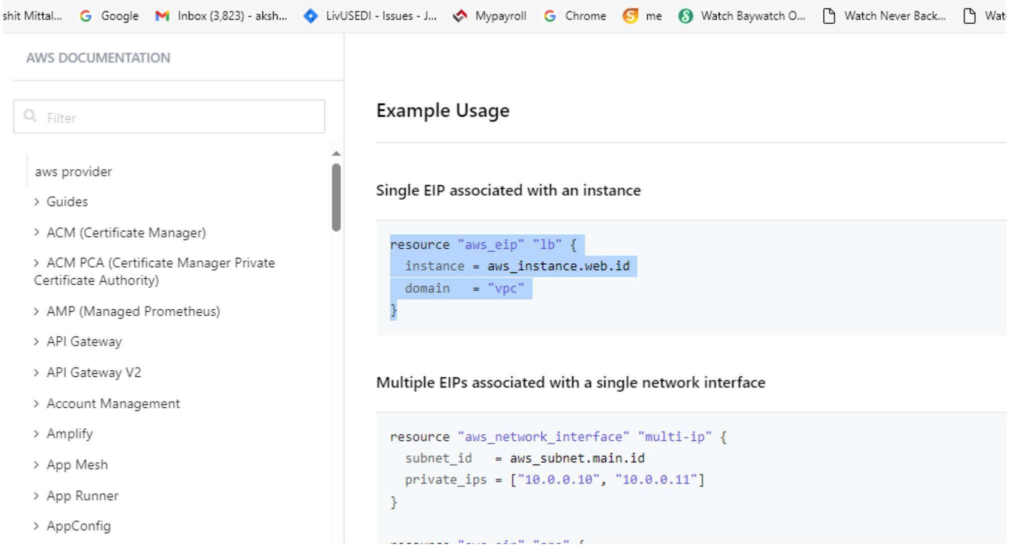
2. Lets create an EIP with terraform. You will find the code syntax in the Hashi-Corp EIP documentation. The Resource “aws_eip” will be remain the same as it is the resource name plugin. The “lb” is the name of your elastic ip you can provide. Instance field is connecting the AWS Instance ID to the EIP ID which will be used later in the association function. The domain tab will be defining the VPC (Virtual Private cloud). You can use the same argument- domain = “vpc” or you can set vpc = true. Run the Apply terraform command again, that will create the EIP address.



EIPs in a VPC. If configured with a provider `default_tags` configuration block present, tags with matching keys will overwrite those defined at the provider-level.

- `vpc` - (Optional **Deprecated**) Boolean if the EIP is in a VPC or not. Use `domain` instead. Defaults to `true` unless the region supports EC2-Classic.

⚠ NOTE:



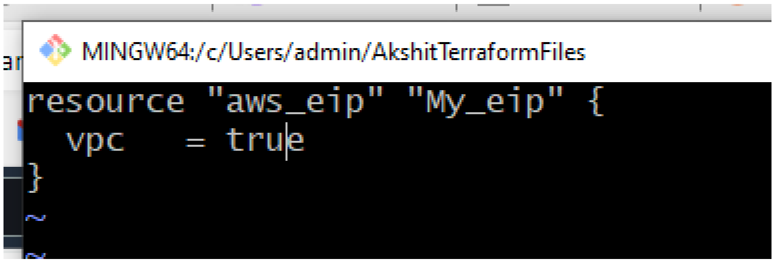
Example Usage

Single EIP associated with an instance

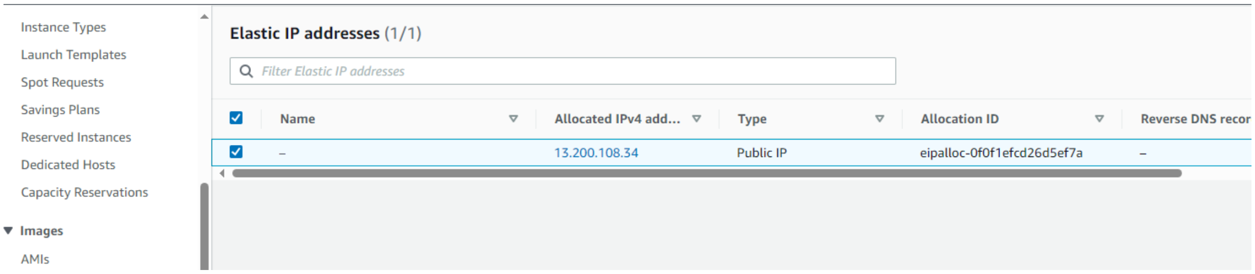
```
resource "aws_eip" "lb" {
  instance = aws_instance.web.id
  domain   = "vpc"
}
```

Multiple EIPs associated with a single network interface

```
resource "aws_network_interface" "multi-ip" {
  subnet_id   = aws_subnet.main.id
  private_ips = ["10.0.0.10", "10.0.0.11"]
}
```



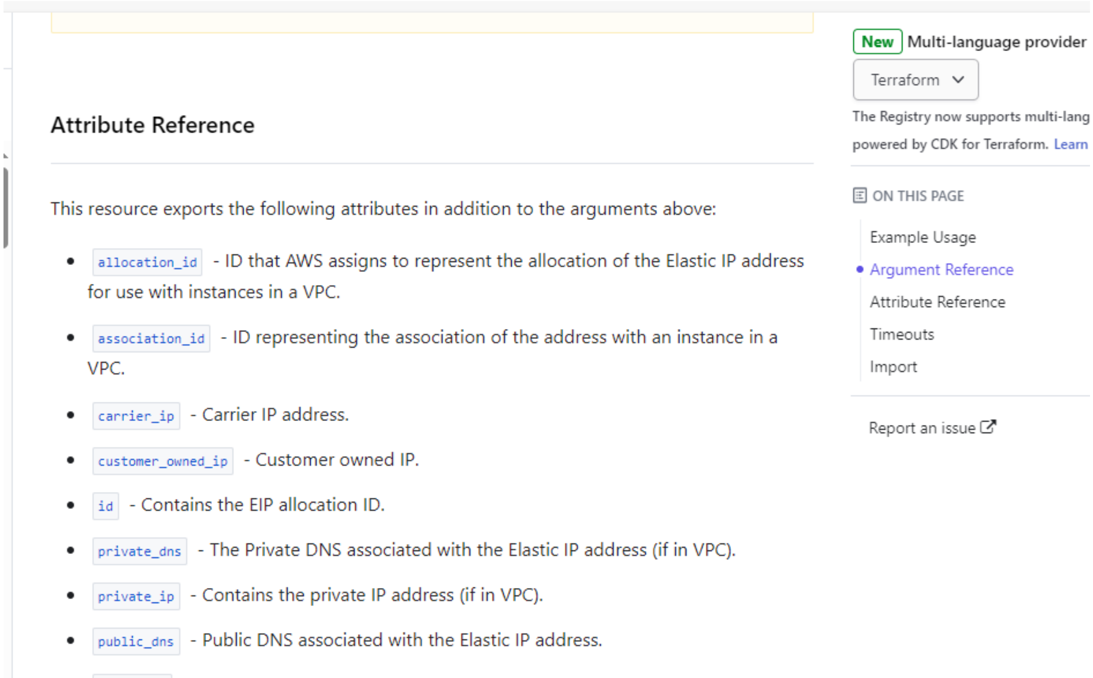
```
resource "aws_eip" "My_eip" {
  vpc    = true
}
~
~
```



Elastic IP addresses (1/1)

☑	Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS recor
☑	–	13.200.108.34	Public IP	eipalloc-0f0f1efcd26d5ef7a	–

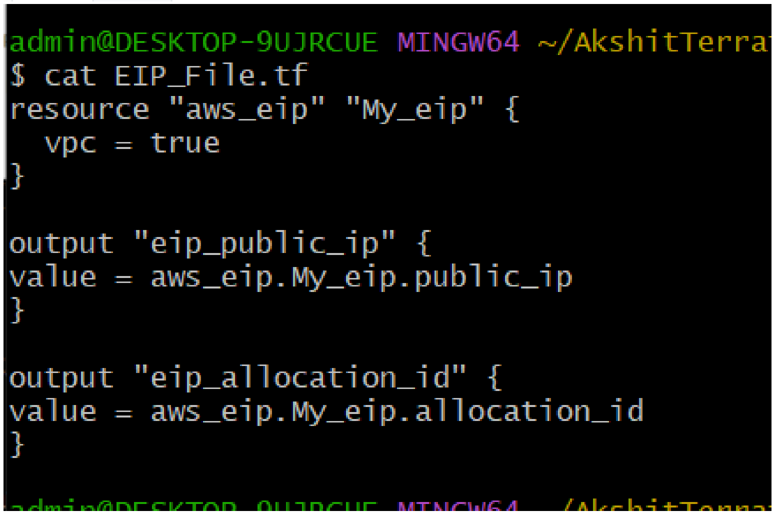
3. You can see the values like allocation_id, carrier_id, private_ip, etc attributes in the console window itself using Output variable. You can find the attribute reference in the documentation.



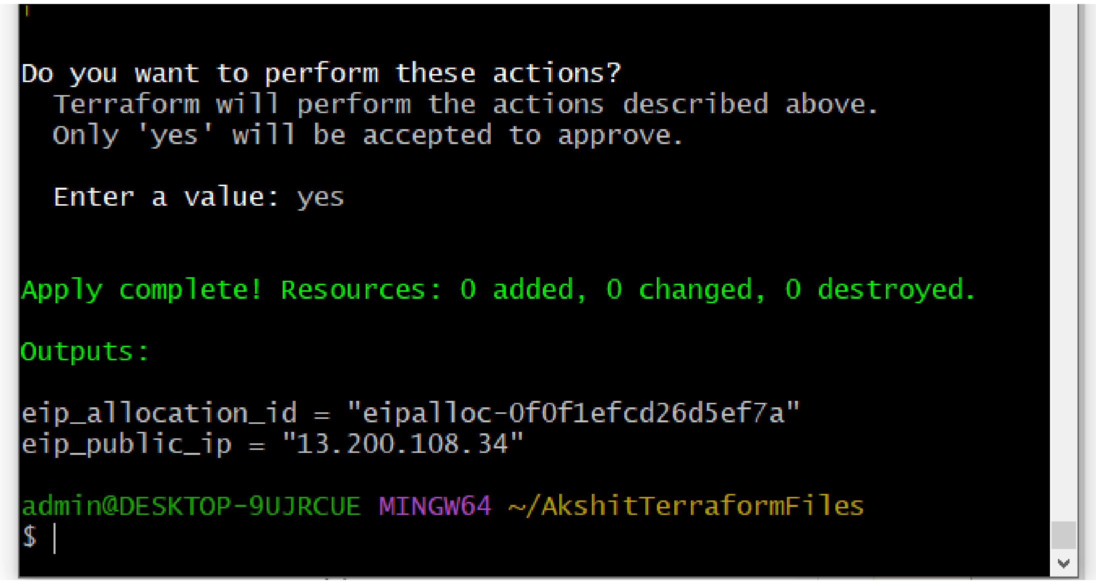
Attribute Reference

This resource exports the following attributes in addition to the arguments above:

- `allocation_id` - ID that AWS assigns to represent the allocation of the Elastic IP address for use with instances in a VPC.
- `association_id` - ID representing the association of the address with an instance in a VPC.
- `carrier_ip` - Carrier IP address.
- `customer_owned_ip` - Customer owned IP.
- `id` - Contains the EIP allocation ID.
- `private_dns` - The Private DNS associated with the Elastic IP address (if in VPC).
- `private_ip` - Contains the private IP address (if in VPC).
- `public_dns` - Public DNS associated with the Elastic IP address.

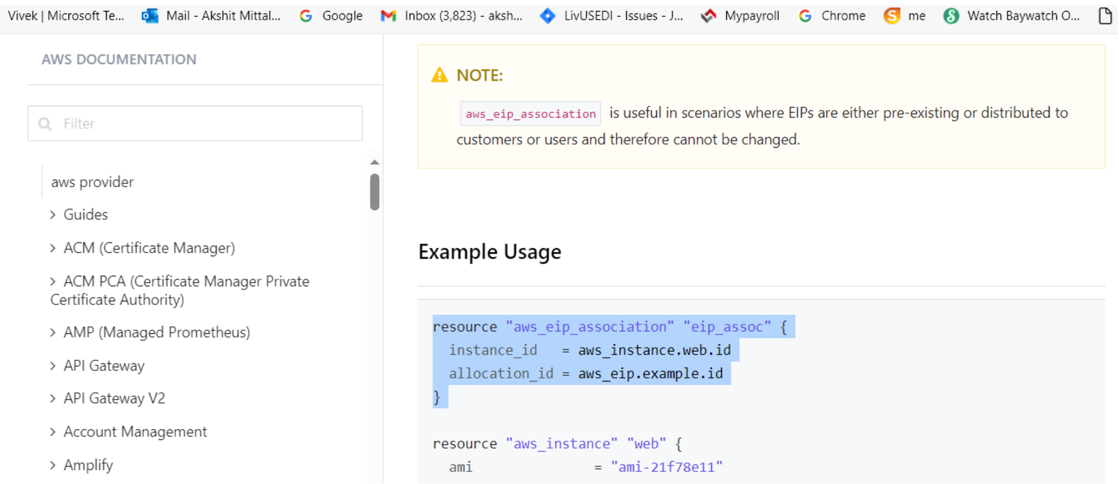


```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerra
$ cat EIP_File.tf
resource "aws_eip" "My_eip" {
  vpc = true
}

output "eip_public_ip" {
value = aws_eip.My_eip.public_ip
}

output "eip_allocation_id" {
value = aws_eip.My_eip.allocation_id
}

admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerra
```



```
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


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

Outputs:

eip_allocation_id = "eipalloc-0f0f1efcd26d5ef7a"
eip_public_ip = "13.200.108.34"

admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ |
```

4. Now we will attach the EIP address to the Instance configuration using the association module in terraform. Syntax you may find on the documentation. Write the syntax code in the EIP allocation file and run terraform apply command to successfully create the ec2 instance containing the EIP address associated. You will be able to see the EIP address in the EC2 instance attributes successfully

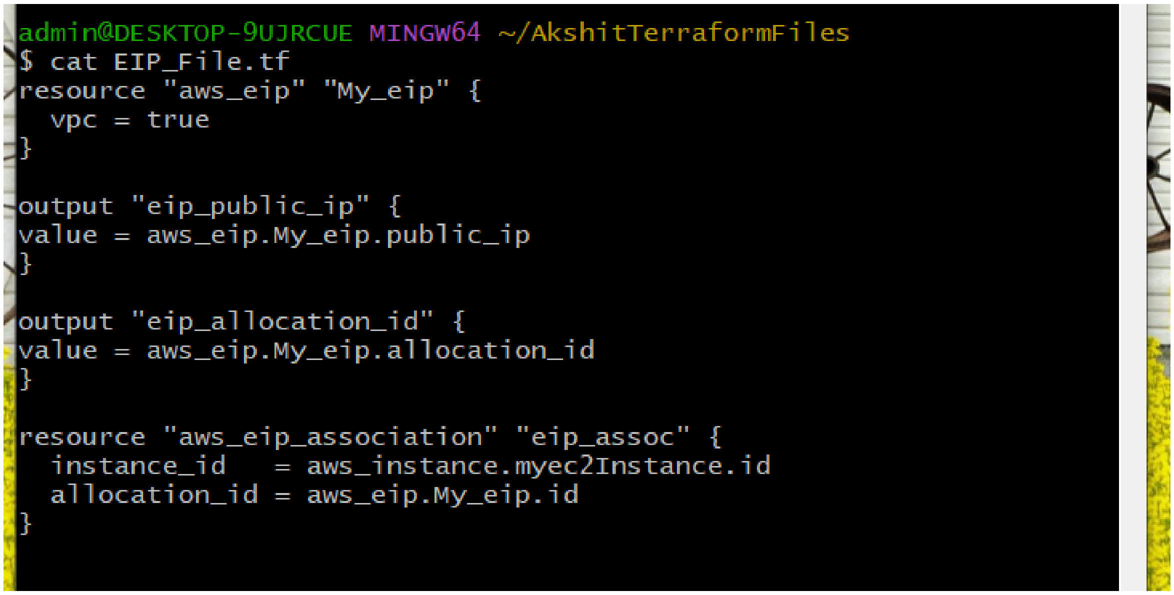


NOTE:

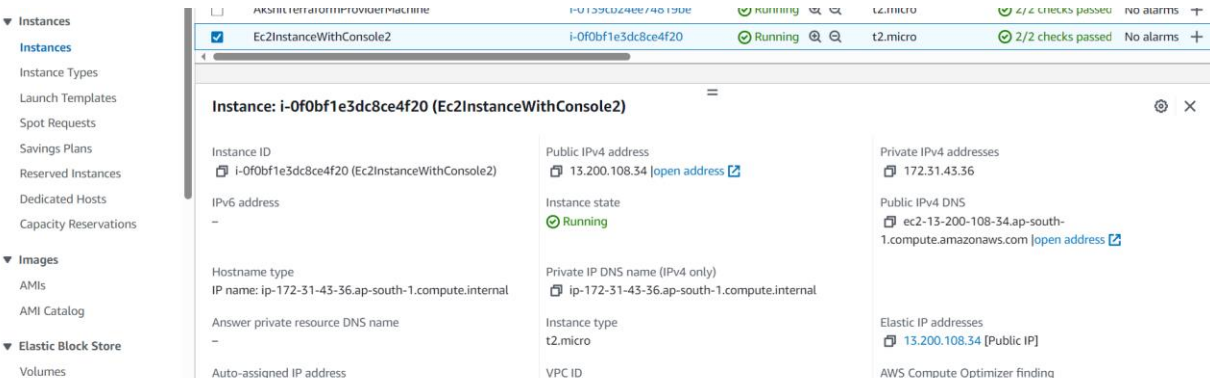
`aws_eip_association` is useful in scenarios where EIPs are either pre-existing or distributed to customers or users and therefore cannot be changed.

Example Usage

```
resource "aws_eip_association" "eip_assoc" {
  instance_id   = aws_instance.web.id
  allocation_id = aws_eip.example.id
}

resource "aws_instance" "web" {
  ami               = "ami-21f78e11"
```



```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat EIP_File.tf
resource "aws_eip" "My_eip" {
  vpc = true
}

output "eip_public_ip" {
value = aws_eip.My_eip.public_ip
}

output "eip_allocation_id" {
value = aws_eip.My_eip.allocation_id
}

resource "aws_eip_association" "eip_assoc" {
  instance_id   = aws_instance.myec2Instance.id
  allocation_id = aws_eip.My_eip.id
}
```

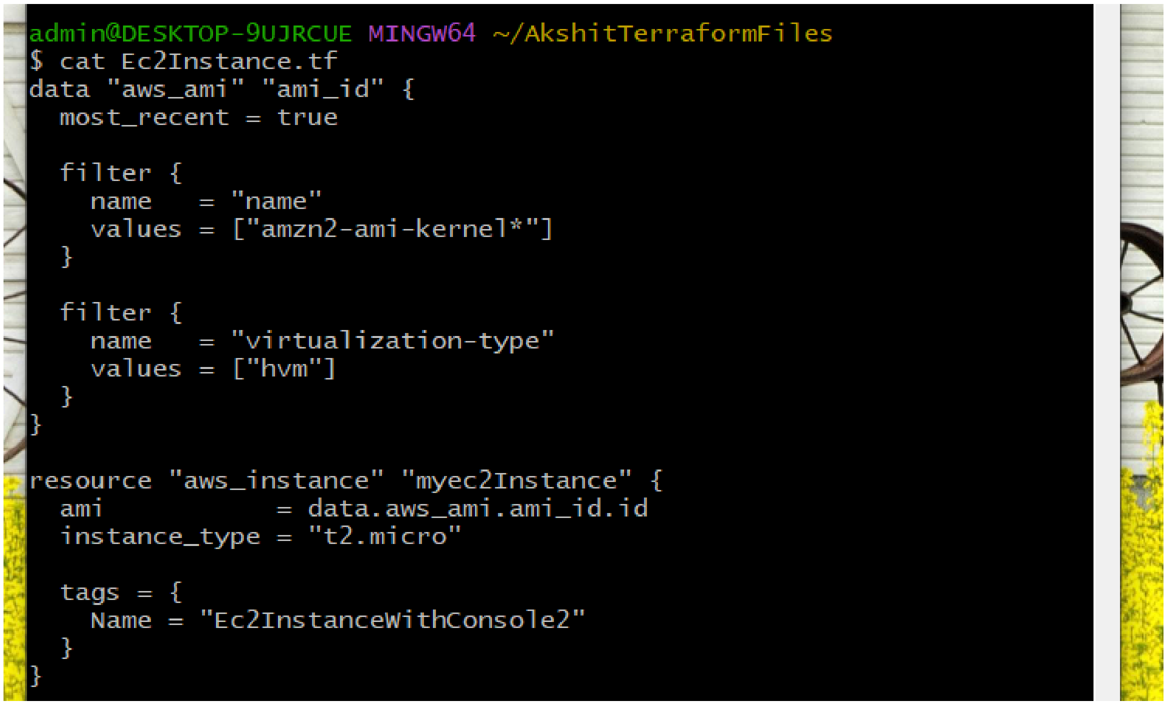


```
☐ AkshitTerraformProviderMachine  i-0139cb24ee74819be  Running  t2.micro  2/2 checks passed  No alarms
☑ Ec2InstanceWithConsole2         i-0f0bf1e3dc8ce4f20  Running  t2.micro  2/2 checks passed  No alarms
```

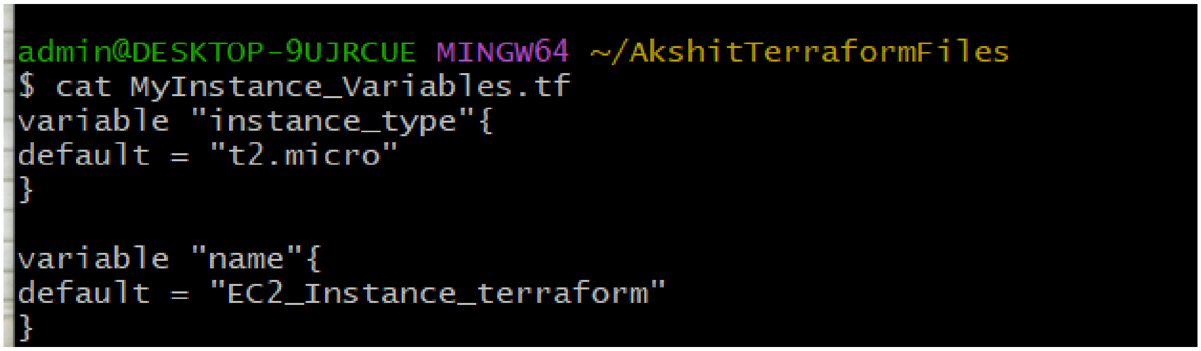
Instance: i-0f0bf1e3dc8ce4f20 (Ec2InstanceWithConsole2)

Instance ID i-0f0bf1e3dc8ce4f20 (Ec2InstanceWithConsole2)	Public IPv4 address 13.200.108.34 |open address	Private IPv4 addresses 172.31.43.36
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-13-200-108-34.ap-south-1.compute.amazonaws.com |open address
Hostname type IP name: ip-172-31-43-36.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-43-36.ap-south-1.compute.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses 13.200.108.34 [Public IP]
Auto-assigned IP address	VPC ID	AWS Compute Optimizer finding

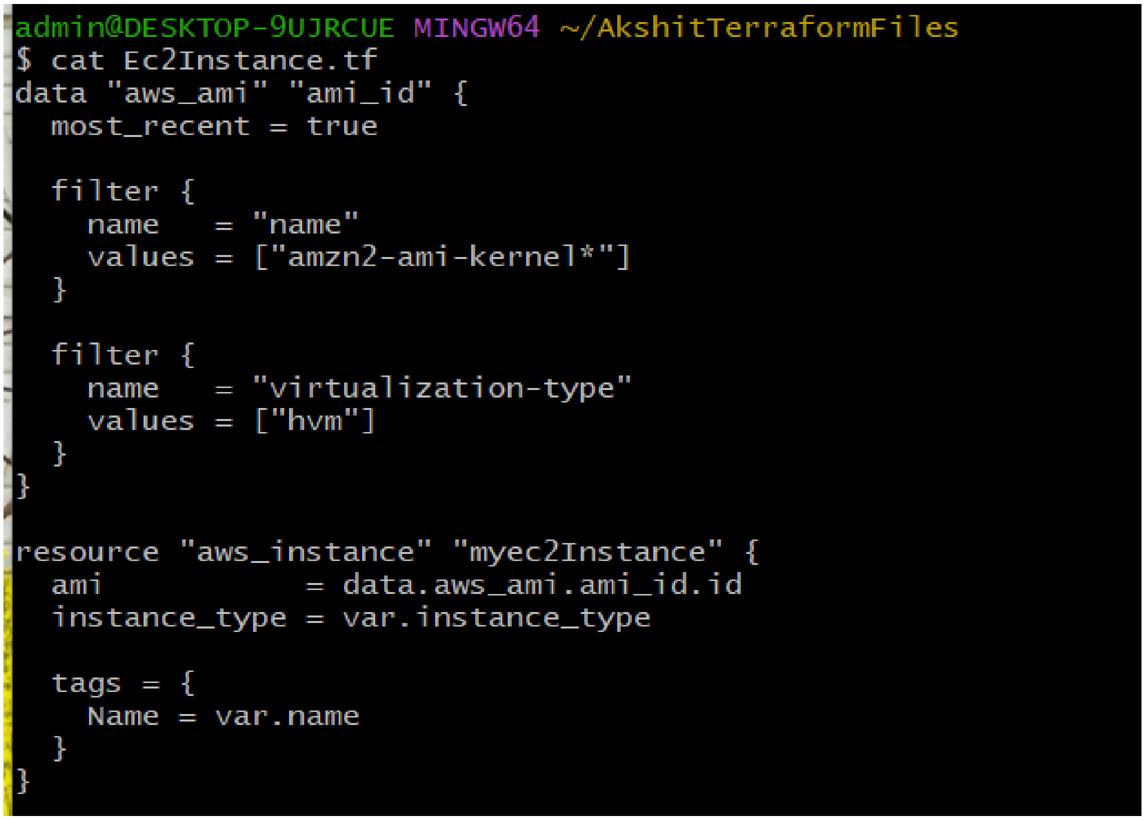
5. Now lets explore the variable function in the terraform. The EC2 instance configuration file, defines the instance type and name for out EC2 machine. We can store those values as default variables in the separate file and call the variable reference in this file, and apply terraform, to create the instance successfully.



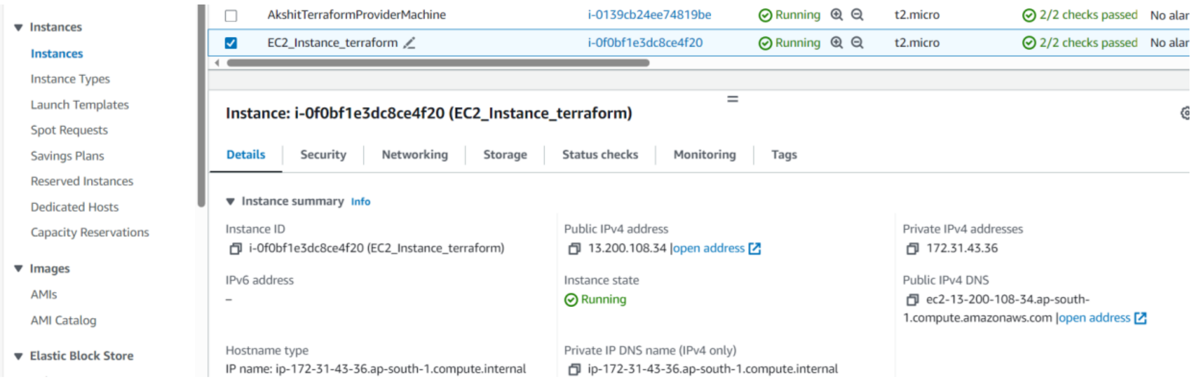
```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat Ec2Instance.tf
data "aws_ami" "ami_id" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-kernel*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

resource "aws_instance" "myec2Instance" {
  ami           = data.aws_ami.ami_id.id
  instance_type = "t2.micro"

  tags = {
    Name = "Ec2InstanceWithConsole2"
  }
}
```



```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat MyInstance_Variables.tf
variable "instance_type"{
default = "t2.micro"
}

variable "name"{
default = "EC2_Instance_terraform"
}
```



```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat Ec2Instance.tf
data "aws_ami" "ami_id" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-kernel*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

resource "aws_instance" "myec2Instance" {
  ami           = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = var.name
  }
}
```



☐	AkshitTerraformProviderMachine	i-0139cb24ee74819be	Running	t2.micro	2/2 checks passed	No alar
☑	EC2_Instance_terraform	i-0f0bf1e3dc8ce4f20	Running	t2.micro	2/2 checks passed	No alar

Instance: i-0f0bf1e3dc8ce4f20 (EC2_Instance_terraform)

▼ Instance summary Info

Instance ID: i-0f0bf1e3dc8ce4f20 (EC2_Instance_terraform)

Public IPv4 address: 13.200.108.34 |open address

Private IPv4 addresses: 172.31.43.36

IPv6 address: –

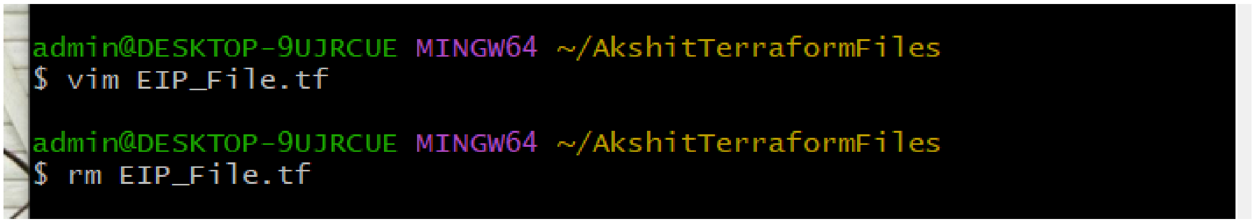
Instance state: Running

Public IPv4 DNS: ec2-13-200-108-34.ap-south-1.compute.amazonaws.com |open address

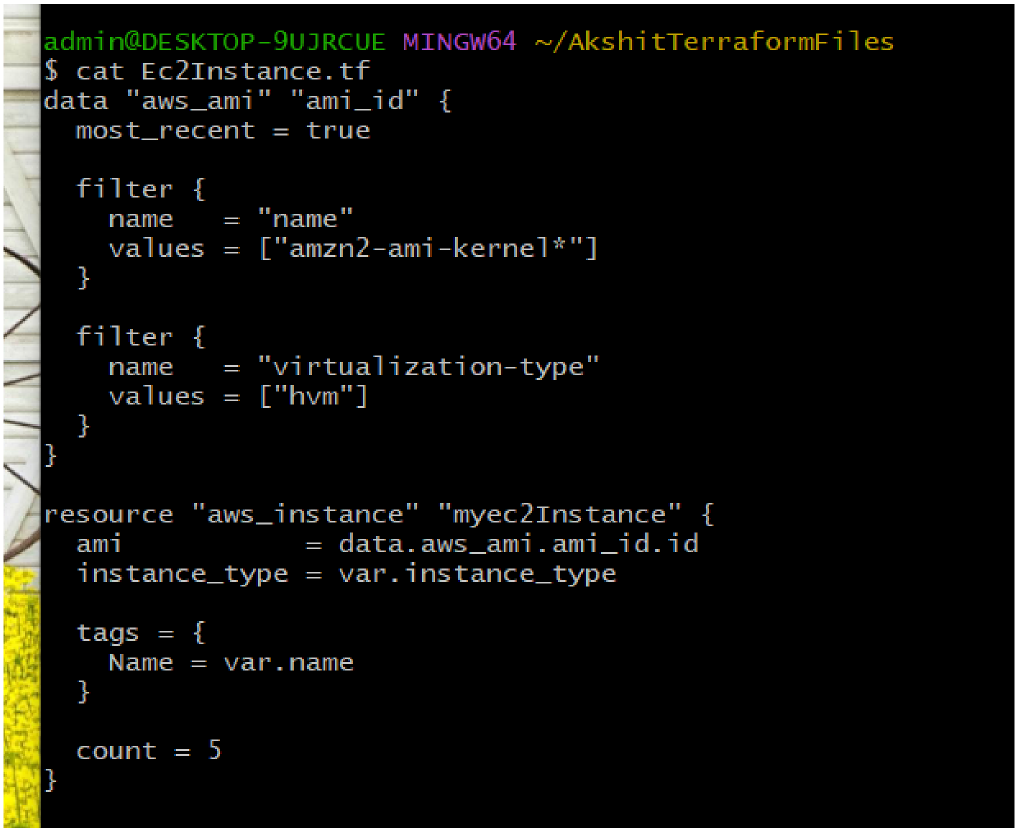
Hostname type: IP name: ip-172-31-43-36.ap-south-1.compute.internal

Private IP DNS name (IPv4 only): ip-172-31-43-36.ap-south-1.compute.internal

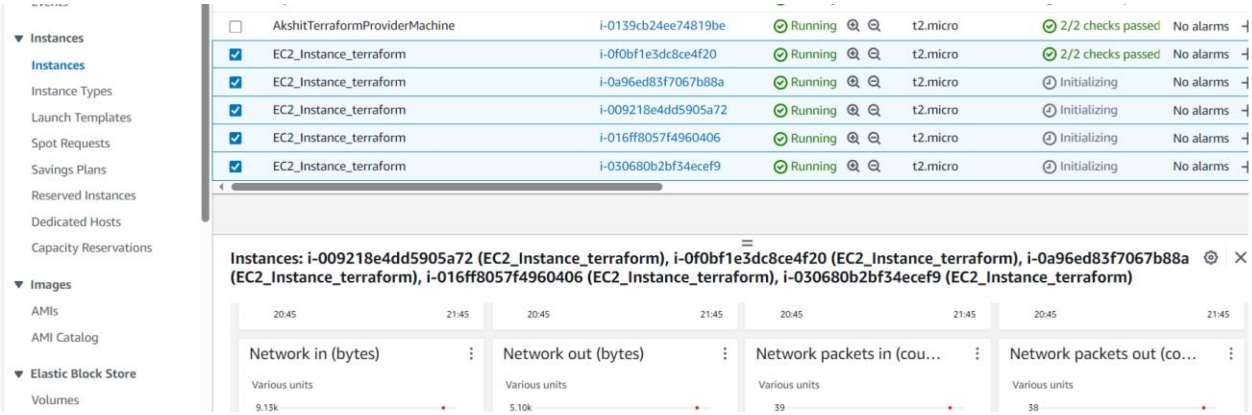
6. Lets explore the Count function in the terraform. You can create multiple instances at the same time by the count function in the terraform. But prior to that you have to delete the EIP address config file, because AWS cannot allocate the same EIP to the multiple instances. Give variable count = “ ” - any number of instance you want to create in the EC2 config file and execute apply terraform to create multiple instances at once.
   This is the main real time application of the Terraform tool in creating the multiple infrastructures at once, instead of creating them manually.



```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ vim EIP_File.tf

admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ rm EIP_File.tf
```



```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshitTerraformFiles
$ cat Ec2Instance.tf
data "aws_ami" "ami_id" {
  most_recent = true

  filter {
    name   = "name"
    values = ["amzn2-ami-kernel*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

resource "aws_instance" "myec2Instance" {
  ami           = data.aws_ami.ami_id.id
  instance_type = var.instance_type

  tags = {
    Name = var.name
  }

  count = 5
}
```



□	AkshitTerraformProviderMachine	i-0139cb24ee74819be	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms
☑	EC2_Instance_terraform	i-0f0bf1e3dc8ce4f20	✓ Running	t2.micro	✓ 2/2 checks passed	No alarms
☑	EC2_Instance_terraform	i-0a96ed83f7067b88a	✓ Running	t2.micro	Initializing	No alarms
☑	EC2_Instance_terraform	i-009218e4dd5905a72	✓ Running	t2.micro	Initializing	No alarms
☑	EC2_Instance_terraform	i-016ff8057f4960406	✓ Running	t2.micro	Initializing	No alarms
☑	EC2_Instance_terraform	i-030680b2bf34ecef9	✓ Running	t2.micro	Initializing	No alarms

Instances: i-009218e4dd5905a72 (EC2_Instance_terraform), i-0f0bf1e3dc8ce4f20 (EC2_Instance_terraform), i-0a96ed83f7067b88a (EC2_Instance_terraform), i-016ff8057f4960406 (EC2_Instance_terraform), i-030680b2bf34ecef9 (EC2_Instance_terraform)