

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.

The screenshot displays the AWS Management Console interface for an EC2 instance. At the top, a table lists two instances: 'AkshitTerraformProviderMachine' and 'Ec2InstanceWithConsole2'. The 'Ec2InstanceWithConsole2' instance is selected, and its details are shown below. The instance ID is 'i-0f0bf1e3dc8ce4f20', and it is in a 'Running' state. The public IPv4 address is '13.232.213.118', and the private IPv4 address is '172.31.43.36'. The instance type is 't2.micro'. The console also shows the instance's DNS name, 'ip-172-31-43-36.ap-south-1.compute.internal', and the public IPv4 DNS name, 'ec2-13-232-213-118.ap-south-1.compute.amazonaws.com'.

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0f0bf1e3dc8ce4f20 (Ec2InstanceWithConsole2)	13.232.213.118 open address	172.31.43.36

The screenshot displays the AWS Management Console interface for the 'Elastic IP addresses' section. The left sidebar shows the 'Instances' menu, and the main content area shows the 'Elastic IP addresses' section. The section includes a search bar and a table with columns for 'Name', 'Allocated IPv4 add...', 'Type', and 'Allocation ID'. The table is currently empty.

Name	Allocated IPv4 add...	Type	Allocation ID
------	-----------------------	------	---------------

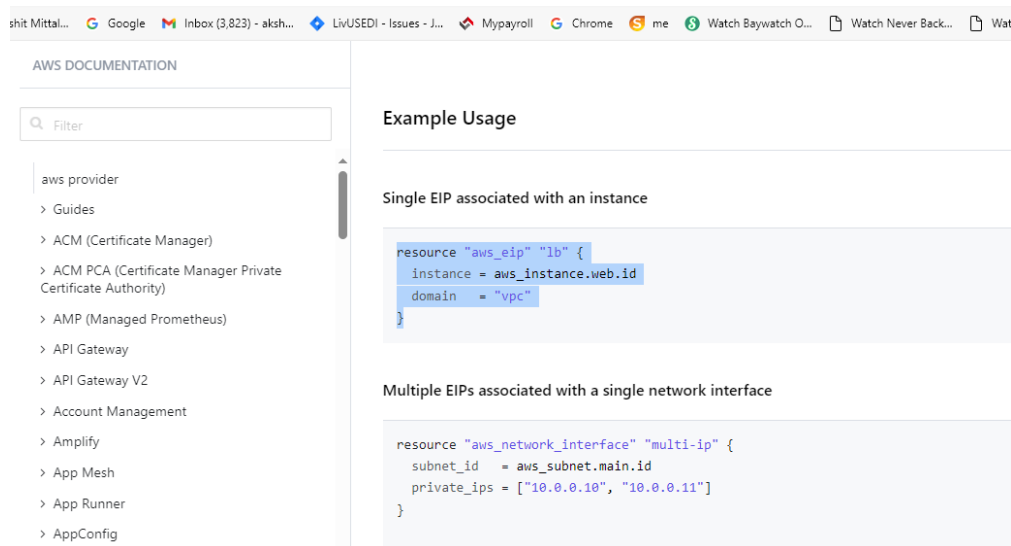
CLOUD & DEVOPS

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.

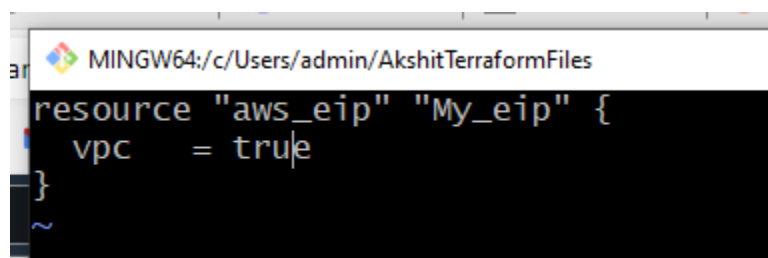
NOTE:



The screenshot shows the AWS Documentation page for the `aws_eip` resource. The left sidebar contains a search bar and a list of navigation links including 'aws provider', 'Guides', and various AWS services. The main content area is titled 'Example Usage' and contains two code blocks. The first block, 'Single EIP associated with an instance', shows a Terraform resource definition for `aws_eip` with attributes `instance` and `domain`. The second block, 'Multiple EIPs associated with a single network interface', shows a Terraform resource definition for `aws_network_interface` with attributes `subnet_id` and `private_ips`.

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

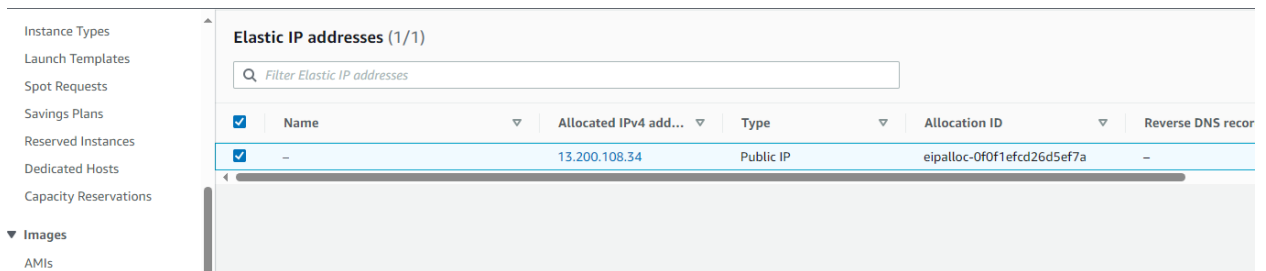
```
resource "aws_network_interface" "multi-ip" {
  subnet_id = aws_subnet.main.id
  private_ips = ["10.0.0.10", "10.0.0.11"]
}
```



The screenshot shows a terminal window with the command prompt `MINGW64:/c/Users/admin/AkshitTerraformFiles`. The terminal displays the following Terraform code snippet:

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

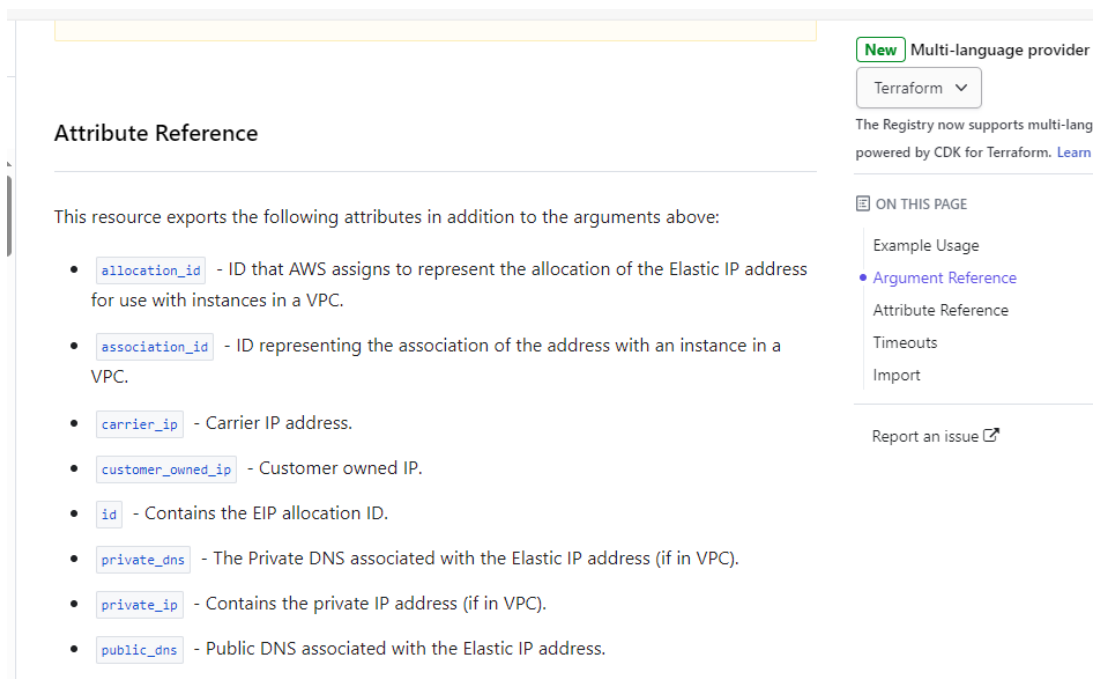
CLOUD & DEVOPS



The screenshot shows the AWS Elastic IP addresses console. On the left is a navigation menu with options like Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, and AMIs. The main panel is titled 'Elastic IP addresses (1/1)' and contains a search bar and a table. The table has columns for Name, Allocated IPv4 address, Type, Allocation ID, and Reverse DNS record. One entry is visible with a Name of '-', an Allocated IPv4 address of 13.200.108.34, a Type of Public IP, an Allocation ID of eipalloc-0f0f1efcd26d5ef7a, and a Reverse DNS record of -.

<input checked="" type="checkbox"/>	Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS recor
<input checked="" type="checkbox"/>	-	13.200.108.34	Public IP	eipalloc-0f0f1efcd26d5ef7a	-

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



The screenshot shows the AWS CLI documentation for the `aws_eip` resource. It includes a 'New' badge for 'Multi-language provider', a dropdown menu for 'Terraform', and a note that 'The Registry now supports multi-lang powered by CDK for Terraform'. On the right, there is a sidebar with 'ON THIS PAGE' links: 'Example Usage', 'Argument Reference' (highlighted), 'Attribute Reference', 'Timeouts', and 'Import'. Below the sidebar is a 'Report an issue' link. The main content area is titled 'Attribute Reference' and states: 'This resource exports the following attributes in addition to the arguments above:'. It lists several attributes with their descriptions:

- `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
}
```

```
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

Vivek | Microsoft Te... Mail - Akshit Mittal... Google Inbox (3,823) - aksh... LivUSEDI - Issues - J... Mypayroll Chrome me Watch Baywatch O...

AWS DOCUMENTATION

Filter

- aws provider
 - > Guides
 - > ACM (Certificate Manager)
 - > ACM PCA (Certificate Manager Private Certificate Authority)
 - > AMP (Managed Prometheus)
 - > API Gateway
 - > API Gateway V2
 - > Account Management
 - > Amplify

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
}

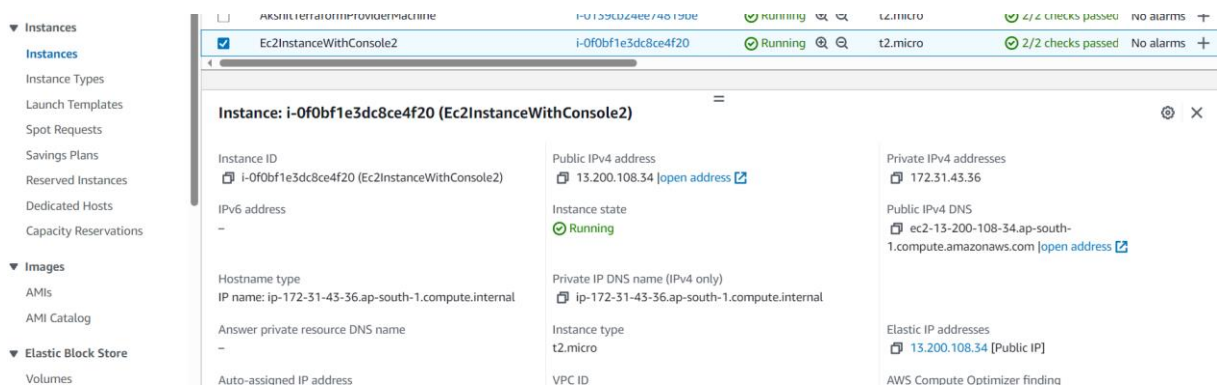
resource "aws_instance" "web" {
  ami = "ami-21f78e11"
```

```
admin@DESKTOP-9UJRCUE MINGW64 ~/AkshiTerraformFiles
$ 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
}
```



- Now let's explore the variable function in the terraform. The EC2 instance configuration file, defines the instance type and name for our EC2 machine. We can store those values as default variables in a 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 ~/AkshiTerraformFiles
$ 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
  }
}
```

The screenshot displays the AWS Management Console interface. On the left, a navigation menu includes 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images', 'AMIs', 'AMI Catalog', and 'Elastic Block Store'. The main content area shows a list of instances at the top, with 'EC2_Instance_terraform' (ID: i-0f0bf1e3dc8ce4f20) selected. Below this, the 'Instance: i-0f0bf1e3dc8ce4f20 (EC2_Instance_terraform)' details are shown. The 'Instance summary' section includes the following information:

Property	Value
Instance ID	i-0f0bf1e3dc8ce4f20 (EC2_Instance_terraform)
Public IPv4 address	13.200.108.34 open address
Private IPv4 addresses	172.31.43.36
Instance state	Running
Public IPv4 DNS	ec2-13-200-108-34.ap-south-1.compute.amazonaws.com open address
IPV6 address	-
Private IP DNS name (IPv4 only)	ip-172-31-43-36.ap-south-1.compute.internal
Hostname type	IP name: ip-172-31-43-36.ap-south-1.compute.internal

6. Let's 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.

CLOUD & DEVOPS

```
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
}
```

Instances

- Instances
 - Instance Types
 - Launch Templates
 - Spot Requests
 - Savings Plans
 - Reserved Instances
 - Dedicated Hosts
 - Capacity Reservations
- Images
 - AMIs
 - AMI Catalog
- Elastic Block Store
 - Volumes

Instance ID	Instance Type	Status	Progress	Alarms
AkshitTerraformProviderMachine	t2.micro	Running	2/2 checks passed	No alarms
EC2_Instance_terraform	t2.micro	Running	2/2 checks passed	No alarms
EC2_Instance_terraform	t2.micro	Running	2/2 checks passed	No alarms
EC2_Instance_terraform	t2.micro	Initializing	2/2 checks passed	No alarms
EC2_Instance_terraform	t2.micro	Initializing	2/2 checks passed	No alarms
EC2_Instance_terraform	t2.micro	Initializing	2/2 checks passed	No alarms
EC2_Instance_terraform	t2.micro	Initializing	2/2 checks passed	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)

Network in (bytes)	Network out (bytes)	Network packets in (count)	Network packets out (count)
9.13k	5.10k	39	38